

GeoGebra Manual

The official manual of GeoGebra.

Introduction

Choose your favorite GeoGebra Math App!

GeoGebra provides several Math Apps^[1] for learning and teaching at all levels. This manual covers the commands and tools of our GeoGebra Classic App^[2].

You may also be interested in our other apps:

GeoGebra Graphing Calculator^[3] and GeoGebra Graphing Calculator Tutorials^[4]

GeoGebra Geometry App^[5] and GeoGebra Geometry Tutorials^[6]

GeoGebra 3D Graphing App^[7] and GeoGebra 3D Graphing Tutorials^[8]

GeoGebra Classic User Interface

The following information is about our GeoGebra Classic App^[2] which you can use online^[9] and also download^[10] as an offline version.

Views and Perspectives

GeoGebra Classic provides different *Views* for mathematical objects:

Algebra View *Graphics View* *Spreadsheet View*

CAS View *3D Graphics View* *Probability Calculator View*

Each View offers its own *Toolbar* that contains a selection of *Tools* and range of *Commands* as well as Predefined Functions and Operators that allow you to create dynamic constructions with different representations of mathematical objects.

Depending on the mathematics you want to use GeoGebra Classic for, you can select one of the default *Perspectives* (e.g. *Algebra Perspective*, *Geometry Perspective*). Each *Perspective* displays those *Views* and other interface components most relevant for the corresponding field of mathematics.

Other Components of the User Interface

You may also customize GeoGebra Classic's user interface to match your personal needs by changing the default *Perspectives* and adding other components:

- *Menubar*
- *Input Bar*
- *Style Bar*
- *Navigation Bar*
- *Context Menu*
- *Virtual Keyboard*

GeoGebra Classic's user interface also provides a variety of dialogs. Different accessibility features as well as keyboard shortcuts allow you to access many features of GeoGebra Classic more conveniently.

Getting Started

If you need additional information about using the GeoGebra Classic App have a look at our GeoGebra Classic App Tutorials ^[11]. You will get step-by-step instructions and learn how to use GeoGebra Classic for different constructions. You can also visit the Geogebra User Forum ^[12] to get help. In addition you can get further information on our GeoGebra YouTube Channel ^[13].

Hints for Advanced Users

- **Explore further GeoGebra Maths Apps and the corresponding Tutorials**
 - GeoGebra Graphing Calculator ^[3] and GeoGebra Graphing Calculator Tutorials ^[4]
 - GeoGebra Geometry App ^[5] and GeoGebra Geometry Tutorials ^[6]
 - GeoGebra 3D Graphing App ^[7] and GeoGebra 3D Graphing Tutorials ^[8]
- **Publish your Work**
 - Create a Dynamic Activity ^[14] by using our online Activity Editor ^[15] on GeoGebra ^[1].
 - Create a GeoGebra Book ^[16] by using our online GeoGebra Book Editor ^[17] on GeoGebra ^[1].
 - Upload your GeoGebra Classic files ^[18] on your GeoGebra Account and share your Dynamic Activities ^[19] online on GeoGebra ^[20].
 - Create or join a GeoGebra Group ^[21] and share your resources with members of your GeoGebra Groups ^[22].
- **Tutorials for Experts**
- **Tutorials for Administrators**
- **References for Programmers**

Troubleshooting

- The Installation Guide helps you with installation questions about our GeoGebra Maths Apps on different platforms
- The Compatibility page explains small differences between GeoGebra versions
- Visit our GeoGebra User Forum ^[23] if you have any questions or suggestions

References

- [1] <https://www.geogebra.org/>
- [2] <https://www.geogebra.org/classic>
- [3] <https://www.geogebra.org/graphing>
- [4] <https://www.geogebra.org/m/vd6UC685>
- [5] <https://www.geogebra.org/geometry>
- [6] <https://www.geogebra.org/m/DmVNbn2V>
- [7] <https://www.geogebra.org/3d>
- [8] <https://www.geogebra.org/m/aWhYSpvy>
- [9] <http://www.geogebra.org/classic>
- [10] <http://www.geogebra.org/download>
- [11] <https://www.geogebra.org/m/XUv5mXTm>
- [12] <https://help.geogebra.org/>
- [13] <https://www.youtube.com/user/GeoGebraChannel>
- [14] <https://www.geogebra.org/m/e9Z6UDu4>
- [15] <https://www.geogebra.org/worksheet/new>
- [16] <https://www.geogebra.org/m/P5Zrj0Su>
- [17] <https://www.geogebra.org/book/create>
- [18] <https://www.geogebra.org/m/e9Z6UDu4#material/Y5TI773i>
- [19] <https://www.geogebra.org/m/e9Z6UDu4#material/fS4RooWB>
- [20] <http://www.geogebra.org/>

- [21] <https://www.geogebra.org/m/rQrbooeq>
- [22] <https://www.geogebra.org/groups>
- [23] <http://www.geogebra.org/forum>

Compatibility

GeoGebra is backward compatible in sense that files created with older version should open in later versions. Sometime we make changes which will cause files to behave differently.

Differences from GeoGebra 4.4 to 5.0

- The winding rule for self-intersecting polygons has changed so they will look different.

Differences from GeoGebra 3.2 to 4.0

- lists of angles, integrals, barcharts, histograms etc. are now visible
- lists {Segment[A,B], Segment[B,C]} are now draggable
- circle with given radius (e.g. Circle[(1,1),2]) draggable
- Distance[Point, Segment] gives distance to the Segment (was to the extrapolated line in 3.2)
- Angle[A,B,C] now resizes if B is too close to A or C
- Integral[function f,function g,a,b] is now transcribed to IntegralBetween[function f,function g,a,b].
- Objects that are a translation by a free vector are now draggable, e.g. Translate[A, Vector[(1,1)]]
- Points on paths may behave differently when the path is changed, e.g. point on conic.

LaTeX Equations

The LaTeX rendering is now nicer, but some errors in LaTeX syntax which were ignored in 3.2 will cause missing texts in 4.0.

- Make sure that each \left\{ has corresponding \right. or \right\}
- See here for more tips about getting LaTeX working: <http://forum.geogebra.org/viewtopic.php?f=20&t=33449>

Installation Guide

GeoGebra Installation

- Installation
- Mass Installation
- FAQ

GeoGebra Portable Versions (Windows)

Download Windows Portable ^[1] and after unzipping run

- GeoGebraCalculator.exe for GeoGebra Calculator Suite
- GeoGebraCAS.exe for GeoGebra CAS Calculator
- GeoGebraGeometry.exe for GeoGebra Geometry Calculator
- GeoGebraGraphing.exe for GeoGebra Graphing Calculator
- GeoGebra.exe for GeoGebra Classic 6

GeoGebra Scientific Calculator

- Android Phones & Tablets: GeoGebra Scientific Calculator on Google Play Store ^[2] (recommended), APK ^[3]
- iPhone & iPad: GeoGebra Scientific Calculator on App Store ^[4]

GeoGebra Calculator Suite

- Android Phones & Tablets: GeoGebra Calculator Suite on Google Play Store ^[5] (recommended), APK ^[6]
- iPhone & iPad: GeoGebra Calculator Suite on App Store ^[7]
- Windows desktop: GeoGebra Calculator Suite Offline Installer ^[8] – updates automatically
- Mac Portable: GeoGebra Calculator Suite for OSX 10.9 or later ^[9]

GeoGebra Graphing Calculator

- Android Phones & Tablets: GeoGebra Graphing Calculator on Google Play Store ^[10] (recommended), APK ^[11]
- iPhone & iPad: GeoGebra Graphing Calculator on App Store ^[12]
- Windows desktop: GeoGebra Calculator Suite Offline Installer ^[13] – updates automatically
- Mac Store: GeoGebra Graphing Calculator in the Mac App Store ^[14] (recommended, updates automatically)
- Mac Portable: GeoGebra Graphing Calculator for OSX 10.9 or later ^[15]
- Chromebooks: GeoGebra Graphing Calculator for Chrome ^[16]

GeoGebra Geometry

- Android Phones & Tablets: GeoGebra Geometry on Google Play Store ^[17] (recommended), APK ^[18]
- iPhone & iPad: GeoGebra Geometry on App Store ^[19]
- Windows desktop: GeoGebra Geometry Offline Installer ^[20] – updates automatically
- Mac Store: GeoGebra Geometry in the Mac App Store ^[21] (recommended, updates automatically)
- Mac Portable: GeoGebra Geometry for OSX 10.9 or later ^[22]

GeoGebra CAS Calculator

- Android Phones & Tablets: GeoGebra CAS Calculator on Google Play Store ^[23] (recommended), APK ^[24]
- iPhone & iPad: GeoGebra CAS Calculator on App Store ^[25]
- Windows desktop: GeoGebra CAS Calculator Offline Installer ^[26] – updates automatically
- Mac Store: GeoGebra CAS Calculator in the Mac App Store ^[27] (recommended, updates automatically)
- Mac Portable: GeoGebra CAS Calculator for OSX 10.9 or later ^[28]

GeoGebra 3D Graphing Calculator

- Android Phones & Tablets: GeoGebra 3D Grapher on Google Play Store ^[29] (recommended), APK ^[30]
- iPhone & iPad: GeoGebra 3D Graphing Calculator on App Store ^[31]

GeoGebra Classic 6

The following offline versions of GeoGebra Classic 6 are available for tablets, laptops and desktops and include the same user interface as www.geogebra.org/classic ^[2]. This version includes: Graphing, CAS, Geometry, 3D Graphing, Spreadsheet, Probability Calculator and Exam mode.

Windows

- **GeoGebra Classic 6 Installer for Windows** ^[32] (offline installer file, recommended for all Windows versions, updates automatically)
- GeoGebra Classic 6 Portable for Windows ^[1] (runs from USB memory sticks for example, does NOT update automatically)

Mac

- **GeoGebra Classic 6 in the Mac App Store** ^[33]
- GeoGebra Classic 6 Portable for Mac ^[34] (runs from USB memory sticks for example, does NOT update automatically)
- On older Macs (before OS 10.9), please use www.geogebra.org/classic ^[2] or GeoGebra Classic 5

Other GeoGebra Classic 6 versions

- iPad: GeoGebra Classic in the App Store ^[35]
- Chrome Store: GeoGebra Classic (with Exam Mode) in the Chrome Web Store ^[36]
- Linux (deb): 64 bit ^[37] installers for .deb based systems (Mint \geq 18, Debian \geq 8, Ubuntu \geq 14.10)
- Linux (rpm): 64 bit ^[38] installers for .rpm based systems (openSUSE \geq 42.1, Fedora \geq 22, Mageia \geq 5)
- Linux Portable: 64 bit ^[39] (runs from USB memory sticks for example)
- Raspberry Pi 3: Raspbian jessie/stretch ^[40]

Note:

- Ad Linux: the .deb and .rpm installers will automatically add the official GeoGebra repository to the package management system on the workstation. This will enable automatic update of GeoGebra every time a new version is released. If you want to include GeoGebra in your custom Linux distribution with GeoGebra included, the best way is to add the official GeoGebra repository (<http://www.geogebra.net/linux/>) to your package management system. The GPG key of the repository is at <https://static.geogebra.org/linux/office@geogebra.org.gpg.key> - the name of the package is **geogebra-classic**.
- Installing GeoGebra directly from the command line: for Ubuntu, these two commands eliminate the need for downloading .deb files:

```
$ sudo apt-add-repository -u 'deb http://www.geogebra.net/linux/ stable main'
$ sudo apt install geogebra
```

GeoGebra in Exams

We believe that students benefit from using the exact same GeoGebra app in class, for homework and during tests. This way, they will get the most practice with the app and therefore be able to make the best use of our technology in all situations. That's why we have created our GeoGebra Exam Mode ^[41] and added it within each one of our apps. The exam mode locks down mobile devices so students cannot communicate or use any other apps than GeoGebra during a test. This approach has already been field-tested and approved by several ministries of education in several regions. Read more about how to use GeoGebra in exams ^[41].

GeoGebra Classic 5 for Desktop

The good old GeoGebra 5 desktop application with its classic user interface.

- Windows Installer: GeoGebra Classic 5 Installer for Windows ^[42] (recommended, updates automatically)
- Windows Portable: GeoGebra Classic 5 Portable for Windows ^[43] (runs from USB memory sticks for example)
- Mac Portable: GeoGebra Classic 5 Portable for OSX 10.8 or later ^[44] (**doesn't automatically update**)
- Mac Portable (OSX 10.7 or below): GeoGebra Classic 5 Portable for OSX 10.6 and 10.7 ^[45].
- Linux (deb): 64 bit ^[46] installers for .deb based systems (Debian, Mint, Ubuntu)
- Linux (rpm): 64 bit ^[47] installers for .rpm based systems (Red Hat, Fedora, openSUSE)
- Linux Portable: Portable Linux ^[48] bundle for 64-bit Linux systems

Ad Linux: the .deb and .rpm installers will automatically add the official GeoGebra repository to the package management system on the workstation. This will enable automatic update of GeoGebra every time a new version is released. Note that the portable version will not automatically update. If you want to include GeoGebra in your custom Linux distribution with GeoGebra included, the best way is to add the official GeoGebra repository (<http://www.geogebra.net/linux/>) to your package management system. The GPG key of the repository is at <https://static.geogebra.org/linux/office@geogebra.org.gpg.key> - the name of the package is **geogebra5**. This will conflict with the earlier versions (4.0, 4.2 and 4.4), which are named **geogebra** (and **geogebra44** for 4.4) and should be deleted first.

Supported devices and Troubleshooting

Please check the Supported devices ^[49] page for further information about running GeoGebra on different devices, have a look at the FAQ ^[50] for more information and visit the forum ^[51] for support.

References

- [1] <https://download.geogebra.org/package/win-port6>
- [2] <https://play.google.com/store/apps/details?id=org.geogebra.android.scicalc>
- [3] <https://download.geogebra.org/package/android-scientific>
- [4] <https://itunes.apple.com/us/app/geogebra-scientific-calculator/id1412748754>
- [5] <https://play.google.com/store/apps/details?id=org.geogebra.android.calculator.suite>
- [6] <https://download.geogebra.org/package/android-graphing>
- [7] <https://apps.apple.com/us/app/geogebra-calculator-suite/id1504416652>
- [8] <https://download.geogebra.org/package/win-suite>
- [9] <https://download.geogebra.org/package/mac-suite-portable>
- [10] <https://play.google.com/store/apps/details?id=org.geogebra.android>
- [11] <https://download.geogebra.org/package/android-suite>
- [12] <https://itunes.apple.com/us/app/geogebra-graphing-calculator/id1146717204>
- [13] <https://download.geogebra.org/package/windows-graphing>
- [14] <https://itunes.apple.com/app/geogebra-graphing-calculator/id1294018688>

- [15] <https://download.geogebra.org/package/mac-graphing>
 - [16] <https://chrome.google.com/webstore/detail/geogebra-graphing-calculation/ajjjpokmmmljpdacmnejdkbjppjekj>
 - [17] <https://play.google.com/store/apps/details?id=org.geogebra.android.geometry>
 - [18] <https://download.geogebra.org/package/android-geometry>
 - [19] <https://itunes.apple.com/us/app/geogebra-geometry-calculator/id1232591335>
 - [20] <https://download.geogebra.org/package/windows-geometry>
 - [21] <https://itunes.apple.com/app/geogebra-graphing-calculator/id1294020062>
 - [22] <https://download.geogebra.org/package/mac-geometry>
 - [23] <https://play.google.com/store/apps/details?id=org.geogebra.android.cascalc>
 - [24] <https://download.geogebra.org/package/android-cas>
 - [25] <https://itunes.apple.com/us/app/geogebra-cas-calculator/id1436278267>
 - [26] <https://download.geogebra.org/package/windows-cas>
 - [27] <https://itunes.apple.com/app/geogebra-cas-calculator/id1489962804>
 - [28] <https://download.geogebra.org/package/mac-cas>
 - [29] <https://play.google.com/store/apps/details?id=org.geogebra.android.g3d>
 - [30] <https://download.geogebra.org/package/android-3dgrapher>
 - [31] <https://itunes.apple.com/us/app/geogebra-3d-graphing-calc/id1445871976>
 - [32] <https://download.geogebra.org/package/win-autoupdate>
 - [33] <https://itunes.apple.com/us/app/geogebra-math-apps/id1182481622>
 - [34] <https://download.geogebra.org/package/mac-port>
 - [35] <https://itunes.apple.com/gb/app/geogebra/id687678494?mt=8>
 - [36] <https://chrome.google.com/webstore/detail/geogebra/bnbaboihhkjoaolnfoablhlahjnee>
 - [37] <https://www.geogebra.org/download/deb.php?arch=amd64&ver=6>
 - [38] <https://www.geogebra.org/download/rpm.php?arch=amd64&ver=6>
 - [39] <https://download.geogebra.org/package/linux-port6>
 - [40] <https://archive.raspberrypi.org/debian/pool/main/g/geogebra-classic/>
 - [41] <https://www.geogebra.org/m/y3aufmy8>
 - [42] <https://download.geogebra.org/package/win>
 - [43] <https://download.geogebra.org/package/win-port>
 - [44] <https://download.geogebra.org/package/mac>
 - [45] <https://download.geogebra.org/package/mac-old>
 - [46] <https://www.geogebra.org/download/deb.php?arch=amd64>
 - [47] <https://www.geogebra.org/download/rpm.php?arch=amd64>
 - [48] <https://download.geogebra.org/package/linux-port>
 - [49] https://wiki.geogebra.org/en/Reference:Supported_Devices
 - [50] https://wiki.geogebra.org/en/Reference:GeoGebra_Installation_FAQ
 - [51] <https://www.reddit.com/r/geogebra/>
-

Objects

Free, Dependent and Auxiliary Objects

There are two types of objects in GeoGebra: free and dependent. Some of them can be defined to be auxiliary.

Free Objects

Free Objects are objects whose position or value doesn't depend on any other objects. They are created by direct input or e.g. Point Tool. They can be moved, unless they are fixed.

Dependent Objects

Dependent Objects are objects that depend on some other objects. They are created using Tools and Commands.

Auxiliary Objects

Auxiliary Objects are either objects which are defined to be auxiliary by user, or objects which were created by specific tools, e.g. Regular Polygon Tool. Spreadsheet cells are also considered to be auxiliary. They have their separate place in Algebra View.

Geometric Objects

Types of Geometric Objects

GeoGebra works with many types of geometric objects

- Points and Vectors
- Lines and Axes
- Conic sections and Arcs
- Functions
- Curves
- Inequalities
- Intervals

Paths

Lists of points, loci, and some of the above mentioned objects (lines, conic sections, arcs, polygons, functions, single variable inequalities, intervals) are referred to as *paths*. One can define a point to belong to a path using the Point Command. Each point on a path has a path parameter, which is a number ranging from 0 to 1. To get this parameter, you can use the PathParameter Command.

Note: Lists of other paths are also paths.

Regions

You can also restrict points to a *region* (polygon, conic, arc, two variable inequality) using the PointIn Command or Point on Object Tool.

Note: See also Attach / Detach Point Tool.

Points and Vectors

Points and vectors may be entered via Input Bar in Cartesian or polar coordinates (see Numbers and Angles). Points can also be created using Point tools and vectors can be created using the Vector from Point Tool or the Vector Tool and a variety of commands.

Note: Upper case labels denote points, whereas lower case labels refer to vectors. This convention is not mandatory.

Examples: To enter a point P or a vector v in 2D in Cartesian coordinates you may use $P = (1, 0)$ or $v = (0, 5)$. To enter a point P or a vector v in 3D in Cartesian coordinates you may use $P = (1, 0, 2)$ or $v = (0, 5, -1)$. To enter a point P in 2D in polar coordinates, you may use $P = (1; 0^\circ)$ or $v = (5; 90^\circ)$. To enter a point P in 3D in spherical coordinates, enter three coordinates of the type (ρ, θ, φ) like e.g. $P = (1; 60^\circ; 30^\circ)$. To enter a point in the Spreadsheet View, name it using its cell address, e.g.: $A2 = (1, 0)$

Notes:

- You need to use a semicolon to separate polar coordinates. If you don't type the degree symbol, GeoGebra will treat the angle as if entered in radians
- Coordinates of points and vectors can be accessed using predefined functions $x()$ and $y()$ (and $z()$ for 3D points).
- Polar coordinates of point Q can be obtained using $\text{abs}(Q)$ and $\text{arg}(Q)$ (and also $\text{alt}(Q)$ for 3D points).

Examples:

- If $P = (1, 2)$ is a point and $v = (3, 4)$ is a vector, $x(P)$ returns 1 and $y(v)$ returns 4.
- $\text{abs}(P)$ returns 2.24 and $\text{arg}(P)$ returns 26.57° .

Calculations

In GeoGebra, you can also do calculations with points and vectors.

Example: You can create the midpoint M of two points A and B by entering $M = (A + B) / 2$ into the Input Bar. You may calculate the length of a vector v using $\text{length} = \text{sqrt}(v * v)$ or $\text{length} = \text{Length}(v)$. You can get the coordinates of the starting and terminal point of a vector v using the commands $\text{Point}(v, 0)$ and $\text{Point}(v, 1)$ respectively. If $A = (a, b)$, then $A + 1$ returns $(a + 1, b + 1)$. If A is a Complex Numberscomplex number $a+bi$, then $A+1$ returns $a + 1 + bi$.

Vector Product

Let (a, b) and (c, d) be two points or vectors. Then $(a, b) \otimes (c, d)$ returns the z-coordinate of vector product $(a, b, 0) \parallel (c, d, 0)$ as single number.

Similar syntax is valid for lists, but the result in such case is a list.

Example: $\{1, 2\} \otimes \{4, 5\}$ returns $\{0, 0, -3\}$. $\{1, 2, 3\} \otimes \{4, 5, 6\}$ returns $\{3, 6, -3\}$.

Lines and Axes

Lines

You can enter a line as a linear equation in x and y or in parametric form into the Input Bar. In both cases previously defined variables (e.g. numbers, points, vectors) can be used within the equation.

Note: You can enter a line's name at the beginning of the input followed by a colon.

Example: 2D Type in $g: 3x + 4y = 2$ to enter line g as a linear equation. You can enter a line in parametric form thus: $g: X = (-5, 5) + t (4, -3)$ Define the parameters $m = 2$ and $b = -1$. Then, you can enter the equation $h: y = m*x + b$ to get a line h in y -intercept-form.

Example: 3D You can enter a line in parametric form thus: $g: X = (1, 6, 3) + \lambda (7, -4, 4)$; or via $g: \text{Line}[(1, 6, 3), \text{Vector}[(7, -4, 4)]]$ You can enter a line as an intersection of 2 planes, by one of the following 3 equivalent input : $\text{IntersectPath}[4x+7y=46, y+z=9](4x + 7y = 46, y + z = 9)7y = 46 - 4x = 7(9 - z)$

Axes

The two coordinate axes are available in commands using the names $xAxis$ and $yAxis$.

Example: The command $\text{PerpendicularLine}[A, \text{xAxis}]$ constructs the perpendicular line to the x -axis through a given point A .

Conic sections

You may enter a conic section as a quadratic equation in x and y . Prior defined variables (e.g. numbers, points, vectors) can be used within the conic's equation.

Note: The conic section's name can be entered at the beginning of the input, followed by a colon.

Examples

Conic section	Input
Ellipse ell	ell: 9 $x^2 + 16 y^2 = 144$
Hyperbola hyp	hyp: 9 $x^2 - 16 y^2 = 144$
Parabola par	par: $y^2 = 4 x$
Circle c1	c1: $x^2 + y^2 = 25$
Circle c2	c2: $(x - 5)^2 + (y + 2)^2 = 25$

Note: If you define two parameters $a = 4$ and $b = 3$ in advance, you may enter for example an ellipse as $\text{ell}: b^2 x^2 + a^2 y^2 = a^2 b^2$.

Functions

To enter a function you can use previously defined variables (e.g. numbers, points, vectors) as well as other functions.

Example:

- Function f: $f(x) = 3x^3 - x^2$
- Function g: $g(x) = \tan(f(x))$
- Nameless function: $\sin(3x) + \tan(x)$

Note: All available predefined functions (e.g. sin, cos, tan) are described in section Predefined Functions and Operators.

In GeoGebra you can also use commands to get for example, the integral and derivative of a function. You can use If Command to get Conditional Functions.

Note: You can also use the commands $f(x)$ or $f'(x)$, ... in order to get the derivatives of a previously defined function $f(x)$.

Example: Define function f as $f(x) = 3x^3 - x^2$. Then, you can type in $g(x) = \cos(f'(x + 2))$ in order to get function g .

Furthermore, functions can be translated by a vector (see Translate Command) and a free function can be moved by using the Move Tool. Other Transformation Commands can be also applied to functions, but in most cases the result is not a function but a curve.

Limit Function to Interval

In order to limit a function to an interval $[a, b]$, you need to use the Function Command or the If Command.

Example: `If[3<=x<=5, x^2]` and `Function[x^2, 3, 5]` both define a function x^2 restricted to interval $[3, 5]$

Curves

GeoGebra supports the following types of curves:

Parametric curves

Parametric curves of the form $a(t) = (f(t), g(t))$ where t is real parameter within a certain range can be created:

- using the Curve Command or
- by typing their expression directly in the *input bar*, e.g. (t^2, t^3) .

Parametric curves can be used as arguments in the following commands: Tangent, Point, Intersect, Derivative, Length, Curvature, CurvatureVector and OsculatingCircle.

Note: Parametric curves can be used with pre-defined functions and arithmetic operations. For example, input $c(3)$ returns the point at parameter position 3 on curve c . You can also place a point on a curve using tool Point ToolPoint or command Point CommandPoint. Since the endpoints a and b are dynamic you can use slider variables as well (see tool Slider ToolSlider).

Creating a parametric curve through some given points is not possible. You can however try e.g. FitPoly Command to get a function going through these points.

Polar curves

In order to draw a curve defined using polar coordinates, it is possible to use one of the following (equivalent) syntaxes:

Example: $\rho=\sin(2\theta)$, or $\sin(2\theta)$, or $f(t)=(\sin(2*t); t)$, or $(\sin(2*t); t)$, or $f(t)=(\sin(2*t); t)$, $0 < t < \pi$, or $(\sin(2*t); t)$, $0 < t < \pi$, or $\text{Curve}[(\sin(2*t); t), t, 0, 2\pi]$.

Implicit curves

Implicit curves are polynomials in variables x and y . They can be entered directly using the Input Bar.

The ImplicitCurve command generates an implicit curve through a list of points.

Example: $x^4 + y^3 = 2xy$

Inequalities

GeoGebra supports inequalities in one or two variables. There are no limitations for inequalities to appear in Algebra View, but only specific inequalities can be drawn in Graphics View:

- polynomial inequalities in one variable, e.g. $x^3 > x + 1$ or $y^2 > y$,
- quadratic inequalities in two variables, e.g. $x^2 + y^2 + x \cdot y \leq 4$,
- inequalities linear in one variable, e.g. $2x > \sin(y)$ or $y < \sqrt{x}$.

For inequality sign you can use $<$, $>$, \leq , \geq . The Symbols \leq and \geq also valid. Conjunction and disjunction are also supported for inequalities, e.g. $(x > y) \&\& (x + y < 3)$ can be drawn.

In order to show the solution of an inequality as one or more intervals on the x-axis, select the *Show on x-axis* option in the *Style* tab of the Properties dialog of the inequality. (This does not work with every inequality.)

Inequalities are similar to functions. You can test whether x and y satisfy inequality a by typing $a(x, y)$ in the Input Bar, also when A is a point, syntax $a(A)$ is valid. A point can be restricted to the region given by inequality using PointIn Command. For inequality b in one variable, e.g. in x , Point (b) yields a point restricted to the part of x-axis which satisfies inequality b .

Intervals

An interval is a set of numbers between upper and lower bound. To create an interval, type e.g. $2 < x < 3$ in Input Bar. Interval in previous example is open. You can also define closed ($2 \leq x \leq 3$) and semi-closed ($2 \leq x < 3$) intervals.

Note: See also Boolean values.

To determine whether number c belongs to interval r type $r(c)$ into the Input Bar, the result will be a Boolean value. Generalization of intervals are Inequalities.

Commands for intervals

- Min, Max, Midpoint for an interval with lower bound a and upper bound b return numbers a, b and respectively. The result doesn't depend on whether the interval is open, closed or semi-closed.
- Point returns a moveable point whose x-coordinate belongs to the interval and y-coordinate is 0.
- PointIn returns a moveable point whose x-coordinate belongs to the interval and y-coordinate may be changed arbitrarily.

General Objects

Besides Geometric Objects GeoGebra can also handle

- Numbers and Angles
- Complex Numbers
- Boolean values
- Lists
- Matrices
- Texts
- Images

Numbers and Angles

Numbers

You can create numbers by using the Input Bar. If you only type in a number (e.g. 3), GeoGebra assigns a lower case letter as the name of the number. If you want to give your number a specific name, you can type in the name followed by an equal sign and the number (e.g. create a decimal r by typing in $r = 5.32$).

Note: In GeoGebra, numbers and angles use a period (.) as a decimal point.

You can also use the constant π and the Euler constant e for expressions and calculations by selecting them from the drop down list next to the *Input Bar* or by using Keyboard Shortcuts.

Note: If the variable e is not used as a name of an existing object yet, GeoGebra will recognize it as the Euler constant if you use it in new expressions.

Angles

Angles can be entered in degrees ($^\circ$) or radians (rad). The constant π is useful for radian values, and can also be entered as pi .

Note: You can enter a degree symbol ($^\circ$) or the pi symbol (π) by using the following keyboard shortcuts: Alt + O (Mac OS: Ctrl + O) for the degree symbol $^\circ$ Alt + P (Mac OS: Ctrl + P) for the pi symbol π

Examples: Angle α in degree $\alpha = 60^\circ$, or using the Degrees/Minutes/Seconds syntax: $\alpha = 1^\circ 23' 45''$ Angle α in radians $\alpha = pi/3$.

Note: GeoGebra does all internal calculations in radians. The degree symbol ($^\circ$) is nothing but the constant $\pi/180$ used to convert degree into radians.

Examples: If $a = 30$ is a number, then $\alpha = a^\circ$ converts number a to an angle $\alpha = 30^\circ$, without changing its value. If you type in $b = \alpha / ^\circ$, the angle α is converted back to the number $b = 30$, without changing its value.

Note: For dependent angles you can specify whether they may become reflex or not on tab *Basic* of the Properties Dialog.

Free Numbers and Angles

Free numbers and angles can be displayed as sliders in the Graphics View (see Slider Tool). Using the arrow keys, you may change the value of numbers and angles in the Algebra View too (see Manual Animation section).

Limit Value to Interval

Free numbers and angles may be limited to an interval [min, max] by using tab *Slider* of the Properties Dialog (see also Slider Tool).

Texts

Text objects can be easily created using Text Command or Insert Text Tool, or by dragging an object from the Algebra View to the Graphics View. Another way **for advanced users** (described below) is typing into Input Bar directly.

Terms

Static text

does not depend on any mathematical objects and is usually not affected by changes of the construction.

Dynamic text

contains values of objects that automatically adapt to changes made to these objects.

Mixed text

is a combination of static and dynamic text. In order to create a mixed text you may enter the static part of the text using the keyboard (e.g. Point A =). Then, click on the object whose value you want to display in the dynamic part of the text.

Input via Input Bar

Texts can also be created using the Input Bar. In this case the syntax, which separates the dynamic and static parts, is to be considered.

Note: GeoGebra automatically adds the syntax ("Point A = " + A) necessary to create your mixed text: quotation marks around the static part of the text and a plus (+) symbol to connect the different parts of the text.

Input	Description
This is static text	Static text
A	Dynamic text (if point A exists)
"Point A = " + A	Two-part mixed text using the value of point A
"a = " + a + "cm"	Three-part mixed text using the value of number a

Note: If an object with the name xx already exists and you want to create a static text using the object's name, you need to enter it with quotation marks ("xx"). Otherwise, GeoGebra will automatically create a dynamic text that gives you the value of object xx instead of its name. However, you can type any text that doesn't match any existing object's name without the quotation marks.

Note: Within a mixed text, the static part needs to be in between a pair of quotation marks. Different parts of a text (e.g. static and dynamic parts) can be connected using plus (+) symbols. Since 4.0, the + symbols are not mandatory.

Note: Text objects can also use LaTeX for typesetting math.

Boolean values

You can use the Boolean variables *true* and *false* in GeoGebra. Just type, for example, `a = true` or `b = false` into the Input Bar and press the Enter key. With Boolean variables you can e.g. define the conditional visibility of an object.

Check Box and Arrow Keys

Free Boolean variables can be displayed as check boxes in the Graphics View (see tool Check Box Tool). After selecting a Boolean variable in the Algebra View you can use the arrow keys to change the value of the Boolean variable (see Manual Animation).

Note: You may also use Boolean variables like numbers (value 0 or 1). This allows you to use a checkbox as the dynamic speed of an animated slider allowing you to start and stop the animation. In this case, the animation button is only shown in the *Graphics View* if there is also an animated slider with static (i.e. non-dynamic) speed.

Operations

You can use the following operations for Boolean variables and conditions in GeoGebra by either selecting them from the list next to the Input Bar or by entering them using the keyboard:

Operation	List	Keyboard	Example	Object types
Equal	\equiv	<code>==</code>	<code>a ≡ b</code> or <code>a == b</code>	numbers, points, lines, conics, functions (needs CAS), planes, a, b
Not equal	\neq	<code>!=</code>	<code>a ≠ b</code> or <code>a != b</code>	numbers, points, lines, conics, functions (needs CAS), planes a, b
Less than	$<$	<code><</code>	<code>a < b</code>	numbers a, b
Greater than	$>$	<code>></code>	<code>a > b</code>	numbers a, b
Less or equal than	\leq	<code><=</code>	<code>a ≤ b</code> or <code>a <= b</code>	numbers a, b
Greater or equal than	\geq	<code>>=</code>	<code>a ≥ b</code> or <code>a >= b</code>	numbers a, b
And	\wedge	<code>&&</code>	<code>a ∧ b</code> or <code>a && b</code>	booleans a, b
Or	\vee	<code> </code>	<code>a ∨ b</code> or <code>a b</code>	booleans a, b
Not	\neg	<code>!</code>	<code>¬a</code> or <code>!a</code>	boolean a
Exclusive or		<code>Alt+</code>	<code>a ⊕ b</code>	booleans a, b
Implication	\rightarrow	<code>-></code>	<code>a → b</code>	booleans a, b
Parallel	\parallel		<code>a ∥ b</code>	lines a, b
Perpendicular	\perp		<code>a ⊥ b</code>	lines a, b
Belongs to	\in		<code>a ∈ list1</code>	number a, list of numbers list1
Subset	\subseteq		<code>list1 ⊆ list2</code>	lists list1 and list2
Proper subset	\subset		<code>list1 ⊂ list2</code>	lists list1 and list2

Complex Numbers

GeoGebra does not support complex numbers directly, but you may use points to simulate operations with complex numbers.

Example: If you enter the complex number $3 + 4i$ into the Input Bar, you get the point $(3, 4)$ in the Graphics View. This point's coordinates are shown as $3 + 4i$ in the Algebra View.

Note: You can display any point as a complex number in the *Algebra View*. Open the Properties Dialog for the point and select *Complex Number* from the list of Coordinates formats on tab *Algebra*.

The imaginary unit i can be chosen from the symbol box in the Input Bar or written using Alt + i. Unless you are typing the input in CAS View or you defined variable i previously, variable i is recognized as the ordered pair $i = (0, 1)$ or the complex number $0 + 1i$. This also means, that you can use this variable i in order to type complex numbers into the *Input Bar* (e.g. $q = 3 + 4i$), but not in the CAS.

Examples: Addition and subtraction: $(2 + 1i) + (1 - 2i)$ gives you the complex number $3 - 1i$. $(2 + 1i) - (1 - 2i)$ gives you the complex number $1 + 3i$.

Examples: Multiplication and division: $(2 + 1i) * (1 - 2i)$ gives you the complex number $4 - 3i$. $(2 + 1i) / (1 - 2i)$ gives you the complex number $0 + 1i$.

Note: The usual multiplication $(2, 1) * (1, -2)$ gives you the scalar product of the two vectors.

The following commands and predefined operators can also be used:

- `x(w)` or `real(w)` return the real part of the complex number w
- `y(w)` or `imaginary(w)` return the imaginary part of the complex number w
- `abs(w)` or `Length[w]` return the absolute value of the complex number w
- `arg(w)` or `Angle[w]` return the argument of the complex number w

Note: `arg(w)` is a number between -180° and 180° , while `Angle[w]` returns values between 0° and 360° .

- `conjugate(w)` or `Reflect[w, xAxis]` return the conjugate of the complex number w

GeoGebra also recognizes expressions involving real and complex numbers.

Examples: $3 + (4 + 5i)$ gives you the complex number $7 + 5i$. $3 - (4 + 5i)$ gives you the complex number $-1 - 5i$. $3 / (0 + 1i)$ gives you the complex number $0 - 3i$. $3 * (1 + 2i)$ gives you the complex number $3 + 6i$.

Lists

Using curly braces you can create a *list* of several objects (e.g. points, segments, circles).

Example: `L = {A, B, C}` gives you a list consisting of three prior defined points A, B, and C. `L = {(0, 0), (1, 1), (2, 2)}` produces a list that consists of the entered points and also creates these nameless points. The short syntax .. creates a list of successive integers: e.g. `-5..5` creates the list `{-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5}`.

Notes:

- By default, the elements of this list are not shown in the Graphics View.
- Lists can also be used as arguments in list operations (mentioned further in this article) or List Commands.

Accessing Elements of Lists

To access particular elements of a list you can use the Element Command or the simplified syntax shown in the example below:

Example: Let `list = {1, 2, 3, 4, 5}`, then:

- `list(1)` returns the first element of the list: 1
- `list(2)` returns the second element of the list: 2
- ...
- `list(-1)` returns the last element of the list: 5
- `list(-5)` returns the first element of the list: 1
- `list(0)` returns *undefined*, as well as `list(k)` for $k > 5$ or $k < -5$

Comparing Lists of Objects

You can compare two lists of objects by using the following syntaxes and commands:

- `List1 == List2`: checks if the two lists are equal as ordered tuples, and yields *true* or *false*.
- `List1 != List2`: checks if the two lists are not equal as ordered tuples, and yields *true* or *false*.
- `Unique[list1] == Unique[list2]` or `list1 \ list2 == {}`: checks if the two lists are equal as sets (i.e. all repeated elements are ignored, as well as the elements order) and yields *true* or *false*.
- `Sort[list1] == Sort[list2]`: checks if the two lists are equal as multisets (i.e. the elements order is ignored) and yields *true* or *false*.

See also Unique and Sort commands.

List Operators

`<Object> ∈ <List>`: returns *true* if *Object* is an element of *List*

`<List1> ⊂ <List2>`: returns *true* if *List1* is subset of *List2*

`<List1> ⊊ <List2>`: returns *true* if *List1* is a strict subset of *List2*

`<List1> \ <List2>`: creates the set difference of *List1* and *List2*

Apply Predefined Operations and Functions to Lists

If you apply Predefined Functions and Operators to lists, you will always get a new list as a result.

Addition and subtraction

- `List1 + List2`: adds the corresponding elements of two lists. **Note:** The two lists need to be of the same length.
- `List + Number`: adds *Number* to every element of *List*.
- `List1 - List2`: subtracts the elements of *List2* from corresponding elements of *List1*. **Note:** The lists need to be of the same length.
- `List - Number`: subtracts *Number* from every element of *List*.

Multiplication and division

- `List1 * List2`: multiplies the corresponding elements of two lists. **Note:** The lists need to be of the same length. If the two lists are compatible matrices, matrix multiplication is used.
- `List * Number`: multiplies every *List* element by the given *Number*.
- `List1 / List2`: divides the elements of *List1* by the corresponding elements of *List2*. **Note:** The two lists need to be of the same length.
- `List / Number`: divides every *List* element by *Number*.
- `Number / List`: divides *Number* by every element of *List*.

Note: See also Vector product.

Other examples

- `List ^ 2`: squares every element of *List*.
- `2 ^ List`: creates a list of powers of two, using the *List* elements as exponents.
- `List1 ^ List2`: creates a list containing a^b , where a and b are corresponding elements of *List1* and *List2*.
- `sin(List)`: applies the sine function to every element of *List*.

User defined functions can be applied the same way as well.

Matrices

GeoGebra supports real matrices, which are represented as a list of lists that contain the rows of the matrix.

Example: In GeoGebra, `{ {1, 2, 3}, {4, 5, 6}, {7, 8, 9} }` represents the 3x3 matrix

To display a matrix using LaTeX formatting in the Graphics View, use the `FormulaText` command or drag and drop the matrix definition from *Algebra View* to *Graphics View*.

Example: In the Input Bar type `FormulaText[{ {1, 2, 3}, {4, 5, 6}, {7, 8, 9} }]` to display the matrix using LaTeX formatting.

Accessing Elements of Matrices

To access particular elements of a matrix you can use the `Element` Command or the simplified syntax shown in the example below:

Example: Let `matrix = { {1, 2}, {3, 4} }`, then:

- `matrix(1, 1)` returns the first element in the first line: *1*
- `matrix(2, 2), matrix(-1, 2), matrix(2, -1)` and `matrix(-1, -1)` all return the second element of the second line: *4*.
- In general, `matrix(i, j)`, where *i* and *j* are integers, returns the element of the matrix that occupies the *i-th* row and the *j-th* column.

Matrix Operations

Matrix operations are *operations with lists*, so the following syntaxes produce the described results.

Note: Some syntaxes can represent operations which are not defined in the same way in the matrices set.

Addition and subtraction

- `Matrix1 + Matrix2`: adds the corresponding elements of two compatible matrices.
- `Matrix1 - Matrix2`: subtracts the corresponding elements of two compatible matrices.

Multiplication and division

- `Matrix * Number`: multiplies each element of *Matrix* by the given *Number*.
- `Matrix1 * Matrix2`: uses matrix multiplication to calculate the resulting matrix.

Note: The rows of the first and columns of the second matrix need to have the same number of elements.

Example: `{ {1, 2}, {3, 4}, {5, 6} } * { {1, 2, 3}, {4, 5, 6} }` yields the matrix `{ {9, 12, 15}, {19, 26, 33}, {29, 40, 51} }`.

- `2x2 Matrix * Point (or Vector)`: multiplies the *Matrix* by the given *Point / Vector* and yields a point.

Example: `{ {1, 2}, {3, 4} } * (3, 4)` yields the point $A = (11, 25)$.

- `3x3 Matrix * Point (or Vector)`: multiplies the *Matrix* by the given *Point / Vector* and yields a point.

Example: `{ {1, 2, 3}, {4, 5, 6}, {0, 0, 1} } * (1, 2)` gives you the point $A = (8, 20)$.

Note: This is a special case for affine transformations where homogeneous coordinates are used: $(x, y, 1)$ for a point and $(x, y, 0)$ for a vector. This example is therefore equivalent to: `{ {1, 2, 3}, {4, 5, 6}, {0, 0, 1} } * { 1, 2, 1 }`.

- `Matrix1 / Matrix2`: Divides each element of *Matrix1* by the corresponding element in *Matrix2*.

Note: However, GeoGebra supports the syntax `Matrix1 * Matrix2 ^ (-1)`.

Other operations

The section Matrix Commands contains the list of all available commands related to matrices, such as:

- Determinant[Matrix]: calculates the determinant for the given matrix.
- Invert[Matrix]: inverts the given matrix
- Transpose[Matrix]: transposes the given matrix
- ApplyMatrix[Matrix, Object]: apply affine transform given by matrix on object.
- ReducedRowEchelonForm[Matrix]: converts the matrix to a reduced row-echelon form

Action Objects

For interactive worksheets with scripting Action Objects may come handy.

There are four types of them:

Checkboxes

Checkboxes are graphical representations of Boolean values. See Check Box Tool for details. Checkboxes can be created using the aforementioned tool or the Checkbox Command.

Input Boxes

Input Boxes work as text inputs for scripts. The script is triggered by changing text in the Input Box and either pressing enter or leaving the Input Box. The inserted value may be accessed using the %0 variable.

Example: Input Box with $a=a+ \%0$ in its *OnClick* script will increase number a by the entered value. Works only if a is free.

If you want the Input Box to change value of a free object (or redefine dependent object), you may define that object as linked. This way you don't have to insert any script.

Input Boxes can be created using the Input Box Tool or the InputBox Command.

Buttons

Buttons are meant to trigger scripts by being clicked. Although scripts can be triggered by clicking any other object (e.g. an image), using buttons for this makes your worksheet more intuitive.

Buttons can be created using the Button Tool or the Button Command.

Drop-down lists

If you want to show the contents of a list, organized in a drop-down list:

- in the Graphics View: check the *Draw as drop-down list* box in the *Basics* tab of the Properties Dialog of the list.
- in the Spreadsheet View: enter a list into the spreadsheet, then check the *Use Buttons and Checkboxes* box in the *Spreadsheet Options*.

The selected element of a drop-down list may be also obtained using SelectedIndex and SelectedElement commands.

Example: To create a drop-down list with three functions $x+1$, x^2 , \sqrt{x} , first create a list by typing $L=\{x+1, x^2, \sqrt{x}\}$ in the input bar. Then right-click the list in *Algebra View* and select *Object Properties....* Select the *Draw as drop-down list* option in the appearing dialog window. You can also enter a caption that describes the drop-down list. In order to plot in *Graphics View* the currently selected function, enter the command SelectedElement (L) in the input bar.

Selecting objects

Selecting one Object

Activate the Move Tool and click (tap) on an object in order to select it.

Selecting several Objects

You can select several objects at the same time in the following ways:

- Hold the Ctrl-key (Mac OS: Cmd-key) while clicking on different objects.
- Use the Move Tool in order to create a **selection rectangle**. **Note:** All objects within the selection rectangle are selected.

How to create a selection rectangle using the mouse

1. Right-click on the position of the first corner of your desired selection rectangle and hold the mouse button.
2. Drag the pointer to the position of the diagonally opposite corner of your desired selection rectangle.
3. Release the mouse button in order to draw the selection rectangle.

How to create a selection rectangle using a touch screen

1. Long-tap (or double-tap) on the position of the first corner of your desired selection rectangle and keep the finger on the screen.
2. Drag your finger to the position of the diagonally opposite corner of your desired selection rectangle.
3. Remove your finger from the screen in order to draw the selection rectangle.

Change Values

There are two ways of manipulating a free object's value:

- Change the value of the object by entering its name and the new value in the Input Bar.

Example: If you want to change the value of an existing number $a = 3$, type $a = 5$ in the Input Bar and press Enter.

- Edit the algebraic representation: Activate Move Tool and double click on the object in the Algebra View. This opens a text box where you can edit the object's value. Press Enter to apply your changes.

Note: While free objects' values can be changed directly, the values of dependent objects can only be influenced by changing their parent objects or by redefining the dependent object.

Naming Objects

You can assign a certain name to an object when you create it using the Input Bar:

- Points: In GeoGebra, points are always named using upper case letters. Just type in the name (e.g. A, P) and an equal sign in front of the coordinates or commands.

Example: $C = (2, 4)$, $P = (1; 180^\circ)$

- Vectors: In order to distinguish between points and vectors, vectors need to have a lower case name in GeoGebra. Again, type in the name (e.g. v, u) and an equal sign in front of the coordinates or commands.

Example: $v = (1, 3)$, $u = (3; 90^\circ)$

- Lines, circles, and conic sections: These objects can be named by typing in the name and a colon in front of their equations or commands.

Example: $g: y = x + 3$, $c: (x-1)^2 + (y - 2)^2 = 4$, $hyp: x^2 - y^2 = 2$

- Functions: You can name functions by typing, for example, $f(x) =$ or $g(x) =$ in front of the function's equation or commands.

Example: $h(x) = 2x + 4$, $q(x) = x^2$, $\text{trig}(x) = \sin(x)$

Notes: If you don't manually assign a name to an object, GeoGebra assigns the names of new objects in alphabetical order. You can create indices within the names of objects by using an underscore. For example A1 is entered as A_1 and sAB is entered as s_{AB} into the Input Bar.

Reserved labels

These are the labels you can't use for objects: x, y, z, xAxis, yAxis, zAxis, abs, sgn, sqrt, exp, log, ln, ld, lg, cos, sin, tan, acos, arcos, arccos, asin, arcsin, atan, arctan, cosh, sinh, tanh, acosh, arcosh, arccosh, asinh, arcsinh, atanh, arctanh, atan2, erf, floor, ceil, round, random, conjugate, arg, gamma, gammaRegularized, beta, betaRegularized, sec, csc, cosec, cot, sech, csch, coth

In the symbol list of the *Input Bar*, you will find special characters for the following constants:

- π - the circle constant pi, which you can also type with Alt + p
- e - the Euler number, e.g. for the exponential function e^x , which you can also type with Alt + e
- i - the imaginary unit, e.g. for complex numbers like $z = 3 + i$, which you can also type with Alt + i

When the variable names e and i are not used yet, they are automatically read as e and i respectively for convenience.

Renaming objects

The easiest way to change the name of an existing object is to select it, then start typing its new name.

You can also rename an object by then selecting the *Rename* option in the Context Menu of the object or opening the Properties Dialog window of the object and typing the new name in the *Name* box of the *Basic* tab.

Notes: Dependencies are usually automatically resolved: this means that the name of the object is also changed in its definition. Scripting involving objects that were assigned a new name need to be changed manually. If you assign to an object a name that is already in use by another object, the name of this last object will be changed, e.g. if you change the name of point B to A and a point A already exists, the former point A gets the new name A1.

See also Labels and Captions.

Animation

Automatic Animation

GeoGebra allows you to animate not only free numbers and/or angles at the same time, but also dependent points that are constrained on an object (segment, line, function, curve, etc.). In order to be automatically animated, free numbers / angles need to be shown as sliders in the Graphics View.

If you want to animate a free number or angle in GeoGebra, as well as a dependent point, you need to select *Animation On* in the Context Menu of that number, angle or point. In order to stop the animation, you need to un-check *Animation On* in the same context menu.

Note: After animating a free number, a free angle or a dependent point, an animation button appears in the lower left corner of the *Graphics View*. It allows you to either pause or continue an animation.

In the Properties Dialog on tab *Slider* you can change the behavior of the animation:

On the one hand, you may control the *Speed* of the animation.

Note: A speed of 1 means that the animation takes about 10 seconds to run once through the interval of the slider.

On the other hand, you can decide how the animation cycle is repeated:

⇒ *Oscillating*

The animation cycle alternates between Decreasing and Increasing.

⇒ *Increasing*

The slider value is always increasing. After reaching the maximum value of the slider, it jumps back to the minimum value and continues the animation.

⇒ *Decreasing*

The slider value is always decreasing. After reaching the minimum value of the slider, it jumps back to the maximum value and continues the animation.

⇒ *Increasing (Once)*

The slider value is always increasing. After reaching the maximum value of the slider, it stops at this value and ends the animation.

Note: while an automatic animation is activated, GeoGebra remains fully functional. This allows you to make changes to your construction while the animation is playing.

Manual Animation

To manually change a number, angle or point position continuously, select the Move Tool. Then, click on a free number, angle or a dependent point and press either the + or – key or the arrow keys on your keyboard. Keeping one of these keys pressed allows you to produce manual animations.

Example: If the point coordinates depend on a number t like in $P = (2t, t)$, the point will move along a straight line when t is changed continuously.

Note: You can adjust the increment of the slider on tab *Slider* of the *Properties Dialog* of this object.

Keyboard Shortcuts

- Shift + arrow key gives you a step width of 0.1 units
- Ctrl + arrow key gives you a step width of 10 units
- Alt + arrow key gives you a step width of 100 units

Note: A point on a line can also be moved along that line using the + or – key.

Tracing

Objects can leave a trace in the Graphics View when they are moved. Use the Context Menu and select *Trace On*. Alternatively you can go into the Object Properties and activate *Show Trace*. Then, modify the construction so that the object whose trace you turned on changes its position and leaves a trace.

You can turn off the trace of an object by un-checking *Trace On* in the Context Menu. The menu item *Refresh Views* in the View Menu clears all trace.

Note: The trace is not permanent, it disappears e.g. on zoom. Instead of permanent trace of a point you can use Locus.

Tracing to Spreadsheet

When the Spreadsheet View is enabled, it is also possible to trace the changing positions of a point in the Graphics View. To do so, open the Context Menu of a point in the Graphics View and select the *Record to Spreadsheet* option (in the GeoGebra Desktop version it is also possible to trace the point positions in a list, by selecting the *Trace to List* option in the appearing dialog window).

Object Properties

Following properties can be set via Properties Dialog.

Visibility

You may show or hide objects in the Graphics View in different ways.

- You may use tool Show/_/Hide Object Tool to show or hide objects.
- Open the Context Menu and select item *Show Object* to change the visibility status of the selected object.
- In the Algebra View, the icon to the left of every object shows its current visibility state (shown or hidden). You may directly click on the little marble icon in order to change the visibility status of an object.
- You can also use the Check Box Tool in order to show or hide one or several objects.

Note: To make an object "invisible" in the *Algebra View*, make it Auxiliary.

Fixed objects

You can define an object to be fixed via *Properties Dialog*. Fixed objects (both free and dependent) cannot be moved. To stop an object being selected / deleted you can uncheck its "Selection Allowed" option in the Advanced Tab.

Filling

For closed Curves and regions you can specify filling using the *Style tab* of *Object properties dialog*. There are three types of filling:

Standard

Fills the object by color specified in the *Color tab*. The same color is used to draw border of that object. Using *Style tab* you can define opacity -- e.g. Conics have by default opacity 0 which means they are transparent.

Hatching

The object is hatched, angle of hatches and distance between them can be specified. Thickness of hatches equals the thickness of object's border.

Cross-hatching

The object is cross-hatched (available angle for hatches: $0^\circ - 45^\circ$) and the distance between lines can be specified. Thickness of hatches equals the thickness of object's border.

Chessboard

Fills the object using a chessboard pattern (available angle for squares: $0^\circ - 45^\circ$). The squares' heights can be specified.

Dots

Fills the object using a dotted pattern, distance between dots can be specified.

Honeycomb

Fills the object using a honeycomb pattern, distance between the cells can be specified.

Bricks

Fills the object using a brick pattern, angles and bricks' heights can be specified.

Weaving

Fills the object using a weaving pattern, angles and spacing can be specified.

Symbol

Fills the object using a special symbol chosen from a list, distance between symbols can be specified.

Image

You can specify location of image on the local disc. The image is repeated periodically, its size is fixed in pixels and top left corner is aligned to the top left corner of the view.

Note: You can also check the *Inverse filling* box to fill with the selected pattern the whole *Graphics View*, except the selected object.

Advanced properties

Advanced properties such as Dynamic Colors and Conditional Visibility are listed in article Advanced Features.

Labels and Captions

In GeoGebra, each object has its unique **label**. For labeling you can choose one or more letters, possibly with subscript. For details see Naming Objects.

Show and Hide Labels

You can show or hide the objects labels in the Graphics View in different ways:

- Select the Show / Hide Label Tool and select the object whose label you would like to show or hide.
- Open the Context Menu for the desired object and select *Show Label*.
- Open the Properties Dialog for the desired object and check or un-check the checkbox *Show Label* on tab *Basic*.

Name and Value

In GeoGebra, every object has a unique name that can be used to label the object in the *Graphics View*. In addition, an object can also be labeled using its value or its name and value. You can change this label setting in the *Properties Dialog* on tab *Basic* by selecting the corresponding option *Name*, *Value*, or *Name & Value* from the drop down menu next to the checkbox *Show Label*.

Note: The value of a point is its coordinates, while the value of a function is its equation.

Caption

However, sometimes you might want to give several objects the same label, for example, to label the four edges of a square *a*. In this case, GeoGebra offers captions for all objects, in addition to the three labeling options mentioned above. You can set the caption of an object on tab *Basic* of the *Properties Dialog* by entering the desired caption into the text field called "Caption". Afterwards, you can select the labeling option "Caption" from the drop down menu next to the checkbox "Show Label".

You can use following placeholders in captions:

Placeholder	Meaning
%c	The value of the adjacent spreadsheet cell (to the right), which should be an independent text Not dynamic: i.e. the caption won't be updated unless you press F9 or Ctrl + R
%v	Value
%n	Name
%d	Description
%f	Definition
%x	x coordinate (or x coefficient for the line $a x + b y + c = 0$)
%y	y coordinate (or y coefficient for the line $a x + b y + c = 0$)
%z	z coordinate (or constant c' term for the line $a x + b y + c = 0$)

Example: Let A be a point and (1,2) be its coordinates. Setting the caption to "Point %n has coordinates %v" results in caption "Point A has coordinates (1,2)"

LaTeX in Captions

You can also use LaTeX in your labels, enclosing the desired LaTeX command in dollar characters (e.g. $\$ \mathbf{x}^{\{2\}}$ \$), and choose from a list of most commonly used Greek letters and operators, just clicking on the little box displayed at the end of the *Caption* field.

Example: If you want to display a nicely formatted math text, use LaTeX in captions, e.g. to display a fraction like , in the caption field type: $\$ \frac{a}{b} \$$.

Point Capturing

Point Capturing

The *Point Capturing* feature is part of the *Graphics View* and *3D Graphics View Style Bar* and offers four different options for points:

- **Automatic:** turns *Point Capturing On* when the grid or the coordinate axes are shown, and *Off* when both are hidden.
- **Snap to Grid:** when a point is close enough to an intersection of lines of the grid, it will align to it.
- **Fixed to Grid:** a point can be placed only on the intersections of lines of the grid.
- **Off:** no capturing is applied to the point.

Note: *Snap to Grid* and *Fixed to Grid* options, when selected, remain active even though the grid is hidden.

Advanced Features

For geometric objects, following properties can be found on the *Advanced* tab

- Layers
- Conditional Visibility
- Dynamic Colors
- Tooltips
- Object Position

For texts there is a powerful feature: LaTeX. It allows user to create nice looking mathematical formulas.

Object Position

Position of objects can be specified in the Position tab of Properties Dialog.

- For vectors, the position is specified by the start point.
- For images, the position is specified by one, two or three corners (see Image properties for details.)

Position of images and sliders may be fixed with respect to screen. This feature is by default enabled for sliders and disabled for images. To change it, switch *Absolute Position On Screen* in the Position Tab of Properties Dialog. The Action Objects have always absolute position on screen.

Conditional Visibility

Apart from just showing or hiding certain objects you can also have their visibility status depend on a certain condition. For example, you would like an object to appear on screen if you check a checkbox positioned in the Graphics View or if a slider is changed to a certain value.

Conditionally Show or Hide Existing Objects

You can use the Check Box Tool in order to create a checkbox that controls the visibility of one or more existing objects on screen.

Alternatively, you could also create a Boolean variable (e.g. `b = true`) using the Input Bar and make it visible as a checkbox in the *Graphics View* by changing its visibility status (e.g. use Show / Hide Object Tool or the Context Menu). A checkbox therefore is a visible *Boolean variable*. In order to use this Boolean variable as a condition for the visibility of certain objects, you need to follow the steps described below.

Changing the Visibility of Newly Created Objects

In the Properties Dialog, you can enter a condition for the visibility of an object on tab *Advanced*.

Note: You can select the logic operators (e.g. `≠`, `≥`, `^`, `||`) from list to the right of the input field in order to create your conditional statements.

Example: If `a` is a slider, then the conditional statement `a < 2` means that the corresponding object is only shown in the Graphics View if the slider value is less than 2. If `b` is a Boolean variable, you can use `b` as a conditional statement. The corresponding object is shown whenever the value of `b` is true and is hidden when the value of `b` is false. If `g` and `h` are two lines and you would like a text to be shown whenever these lines are parallel, then you could use `g || h` as a conditional statement for the text.

Note: See also SetVisibleInView command.

Dynamic Colors

In GeoGebra, you can change the objects' colour using the *Colour* tab of the Properties Dialog. The colour of an object can also be set to change dynamically: open the *Properties Dialog* of the object whose colour you would like to define, then select the *Advanced* tab. The *Dynamic Colours* sections contains three input boxes which allow you to enter the main colour's components: *Red*, *Green*, and *Blue*.

Note:

- The RGB values of the main colours are displayed to the right of the *Preview* box in the *Colour* tab of the *Properties Dialog*.
- Dynamic colours can also be defined entering a function with range [0,1].

Example: Create three Slider Toolsliders a, b, and c with an interval from 0 to 1. Create a polygon, whose colour will be dynamically related to the sliders values. Open the Properties Dialog for the polygon, then enter the names of the three sliders into the Red, Green and Blue input boxes. Close the Properties Dialog and change the values of the sliders in order to find out how each colour component influences the resulting colour of the polygon.

Note: You may also animate the sliders with different speeds, in order to see the colour of the polygon change automatically.

The *Dynamic Colours* section also contains an input box which allows you to change the *Opacity* of the selected object. You can enter a number ranging in [0,1] (where 0 means transparent, and 1 means 100% opaque), as well as a slider, in order to obtain a dynamic opacity. Other numbers will be ignored.

RGB / HSV / HSL

Some dynamic colouring patterns can be obtained using a different colour model. Besides the default RGB, GeoGebra offers two more models, HSV and HSL, that can be selected in the drop down list displayed at the bottom of the *Dynamic Colours* section of the *Advanced* tab of the *Properties Dialog*.

Example: To make a point A go through all the colours of the rainbow when moved left and right, switch to *HSV* mode, then set *Saturation* and *Value* to 1, and *Hue* to x (A) .

LaTeX

In GeoGebra you can write formulas as well. To do so, check the box LaTeX formula in the dialog window of the Text Tool and enter your formula in LaTeX syntax.

Note: In order to create text that contains a LaTeX formula as well as text you may enter the text inside `\text{ }}, while LaTeX Formula is activated.`

Example: `\text{The length of the diagonal is } \sqrt{2}`

Note: You can also use the **FormulaText** Command to enter your formula within quotes.

Example: `FormulaText ["\text{The length of the diagonal is } \sqrt{2}"]`

Note: You can simply obtain a LaTeX text containing the value of an object listed in the Algebra View by dragging that object in the *Algebra View* and dropping it in a selected location of the Graphics View .

You can find the syntax for common formula symbols from the drop-down menu next to the LaTeX checkbox (GeoGebra Desktop Version) or in the *Advanced* menu below the input field (GeoGebra Web and Tablet Apps Version). This inserts the corresponding LaTeX code into the text field and places the cursor in between a set of curly brackets. The Symbols drop-down menu contains a list of common math symbols, Greek letters and operators. If you would like to create dynamic text within the formula, you need to select the relating objects from the Objects drop-down list, causing GeoGebra to insert their names as well as the syntax for mixed text.

Some important LaTeX commands are explained in following table. Please have a look at any LaTeX documentation for further information.

LaTeX input	Result
a \cdot b	
\frac{a}{b}	
\sqrt{x}	
\sqrt[n]{x}	
\vec{v}	
\overline{AB}	
x^2	
a_1	
\sin\alpha + \cos\beta	
\int_{a}^{b} x dx	
\sum_{i=1}^{n} i^2	

It is also possible to enter nicely formatted chemical formulas, using syntaxes `\ce`, `\mathchoice`, `\mskip`, `\lower`, `\raise`, `\mkern`.

Please refer to this document ^[1] for further information.

References

[1] <https://mhchem.github.io/MathJax-mhchem/>

Layers

Note: In GeoGebra, layers are used to determine which object to select or drag when the user clicks on multiple objects at the same time.

By default, all objects are drawn on layer 0, which is basically the background layer of the Graphics View. A total of 10 layers are available (numbers 0 to 9) and higher numbered layers are drawn on top of lower numbered layers.

Using the *Advanced* tab of the Properties Dialog, you can change the layer for a certain object (layers from 0 to 9 available). Once you change the layer number for at least one object to be different from layer 0 (e.g. to layer 3), all new objects will be drawn on the layer with the highest number that is used for any object.

Note: After selecting any object in the GeoGebra Desktop Version, you can select all objects in the same layer by selecting item *Select Current Layer* (keyboard shortcut: Ctrl + L) from the Edit menu. This menu item is only available if all selected objects lie on the same layer.

Further use of layers

- For SVG export, objects are grouped by layer.
- Layers can be controlled using the JavaScript Interface for GeoGebra applets.

Scripting

Script is a sequence of commands, that are executed one after each other. GeoGebra supports two scripting languages - GGBScript and Javascript. The execution can be triggered by :

- clicking a particular object
- updating a particular object (when value or properties of the object are changed)
- loading the file (in case of JavaScript)
- Javascript listeners (see Reference:JavaScript)

You can set this script via the Tab *Scripting* in the Properties Dialog.

Note: The *Properties* panel needs to be closed for scripts to work.

GGBScript

You can create scripts consisting of GeoGebra commands, like you can use them in the Input Bar. After triggering the script, every command is executed one after each other.

Example:

- a is an integer-valued slider ranging from 1 to 3 (therefore Increment equals 1)
- type in: `list1 = {"red", "green", "blue"}`
- in properties of a , set "On Update" script to `SetColor(a, Element(list1, a))`
- by moving the slider you change its color

Explanation: Every time the slider is moved, there an update occurs. So, for every move the script is called and the value of a is used to get one color from the list and change the color of the slider a .

Note: You can use `#` to start a comment

Hint: There are commands that can be only used for scripting. You can find them in the page [Scripting_Commands](#).

JavaScript

JavaScript is a programming language used by many Internet technologies. Unlike GeoGebra Script, in Javascript the commands don't have to be executed as a simple sequence, but a control flow (`if`, `while`, `for`) can be used. For generic JavaScript you can find a nice tutorial on developer.mozilla.org^[1]. In GeoGebra, you can use special JavaScript methods which allow you to change the construction. These methods belong to `ggbApplet` object, which means that you call them as `ggbApplet.method_name(parameter, ..., parameter)`. For complete list of these methods see Reference:JavaScript.

Example: `for(var i =0;i<10;i++) ggbApplet.evalCommand("A_" +i+"=(random()*10,random()*10)");`

This script creates 10 points A_0 to A_9 at random coordinates.

Note: Scripting with JavaScript is very versatile, but many tasks can also be achieved using the simpler GeoGebraScript.

Global JavaScript

In the *Global JavaScript* part of the *Scripting* tab in the *Properties Dialog* you may define **functions** (not variables) which will be available from the other scripts. You can also define function `ggbOnInit(name, api)`, which is called automatically once the construction is loaded. The `ggbOnInit` function can be used for registering some *listeners*, as shown below.

Example: `function onAdd(name){ alert("Object "+name+" was added."); } function ggbOnInit(name, api){ api.registerAddListener("onAdd"); }` First we defined function `onAdd` that given a string shows a message depending on that string. After that, using the `ggbOnInit` function, we told GeoGebra to call this function whenever a new object is added. Once we reload our construction, function `ggbOnInit` will be called and since then, when user adds a point named e.g. A, message "Object A was added" will appear.

You can also use *listeners* for actions like rename, delete and clear construction. A complete list is available in Reference:JavaScript.

Note: Using any `ggbApplet` methods in *Global JavaScript* outside of `ggbOnInit` will not work as intended since they will be called before the construction is loaded.

References

[1] <https://developer.mozilla.org/en/JavaScript/Guide>

Toolips

Toolips are texts that appear next to your mouse cursor when you hover the cursor over an object in Graphics View. Additionally a tooltip appears, if you hover the cursor over a tool icon in the Toolbar (GeoGebra Desktop Version) or select a tool in the Toolbar (GeoGebra Web and Tablet App Version). In the Advanced tab of Properties Dialog you can specify five tooltip modes:

Automatic

Toolips are shown if Algebra View only is active. Tooltip contains object type and name; in case of dependent objects the tooltip also includes object description.

On

Toolips are shown whether *Algebra View* is shown or not. Content of the Tooltip is the same as for *Automatic*.

Off

No tooltip is shown.

Caption

Caption of the object is used as tooltip. You can set Caption in Basic tab of Properties Dialog.

Next Cell

If the object is a Spreadsheet cell, content of the cell to the right is used as tooltip.

In Advanced tab of Options Dialog you may also specify the language and timeout for tooltips.

Tools

Tools

GeoGebra's *Tools* enable you to produce new Objects using your pointing device. They can be activated by clicking on the corresponding buttons of the *Toolbar*.

Note: All *Tools* have their *Commands equivalents* which are suitable for more complicated constructions.

Different Toolbars for Different Views

Each *View* except the *Algebra View* has its own *Toolbar*, providing *Tools* specific for the *View* you are working with.

- *Graphics View Toolbar*
- *3D Graphics View Toolbar*
- *CAS View Toolbar*
- *Spreadsheet View Toolbar*

Once you start using another *View* within the GeoGebra window, the *Toolbar* changes automatically. If you open another *View* in a separate window, it will have its *Toolbar* attached.

Toolboxes

Each *Toolbar* is divided into *Toolboxes*, containing one or more related *Tools*. You may open a *Toolbox* by clicking

- on one of the default *Tools* in the *Toolbar* (GeoGebra Web and Tablet Apps)
- on the little arrow in the lower right corner of the default *Tools* (GeoGebra Desktop).

You can reorder these *Toolboxes* and save the setting in the GeoGebra Worksheet (*.ggb). See Customizing the *Toolbar* for details.

More Information about Tools

For more information please check our complete list of all available *Tools* as well as the general information about *Toolbars*, creating *Custom Tools* and Customizing the *Toolbar*.

Movement Tools

Movement tools are by default grouped under icon (the first from left) in the toolbar.

Two movement tools are currently available:

- Move
- Move around Point

Move Tool

Move Tool in the Graphics View

Drag and drop Free Objects.

If you select an object by selecting it in *Move* mode, you may...

- ... delete the object by pressing the Delete key
- ... move the object by using the arrow keys (see section Manual Animation)

You can also pan the *Graphics View* by dragging the view's background, while the *Move* tool is activated.

Note: If you are using the GeoGebra Desktop Version, then you can quickly activate the Move Tool by pressing the Esc key of your keyboard. To move a Slider ToolSlider when Move Tool is selected, you need to drag it with your right mouse button. See also Move Graphics View ToolMove Graphics View tool.

Move Tool in the 3D Graphics View

Using the *Move Tool* in the *3D Graphics View* you may drag and drop free points. In order to move a point in the three-dimensional coordinate system, you can switch between two modes by clicking on the point:

- **Mode x-y-plane:** You may move the point parallel to the *x*-*y*-plane without changing the *z*-coordinate.
- **Mode z-axis:** You may move the point parallel to the *z*-axis without changing the *x*- and *y*-coordinates.

Record to Spreadsheet Tool

Note: In GeoGebra 5.0 this tool was replaced by an option in the Context Menu of the Spreadsheet View.

Select objects and choose **Record to Spreadsheet** in the Context Menu, while the *Spreadsheet View* is opened.

This works for numbers, points, and vectors.

Note: GeoGebra will use the first two empty columns of the *Spreadsheet View* to record the values of the selected objects.

Move around Point Tool

Note: This tool is only available in the GeoGebra Desktop Version.

Select the center point of the rotation first. Then, you may rotate free objects around this point by dragging them with the mouse.

Note: See also Rotate command.

Point Tools

Point tools are by default grouped under icon (the second from the left) in the toolbar. Currently there are six point tools:

- Point
- Point on Object
- Intersect
- Midpoint or Centre
- Attach / Detach Point
- Complex Number
- Extremum
- Roots

Point Tool

Click on the drawing pad in the Graphics View in order to create a new point. The coordinates of the point are fixed when the mouse button is released.

Note: By clicking on a segment (or interval), straight line, polygon, conic section, function, or curve you can create a point on this object. Clicking on the intersection of two objects creates this intersection point (see also Intersect ToolIntersect tool and Intersect CommandIntersect command). See also Point CommandPoint command.

Attach / Detach Point Tool

To **attach a point** to a path or region click a free point and the path or region. From now on, the point can still be moved via Move Tool, but only within the path or region.

To **detach a point** that is defined as point on path or region simply select the point. The point will become free.

Note: You can also use Point Command and PointIn Command for attaching a point. See also CopyFreeObject Command.

Complex Number Tool

Click into the Graphics View in order to create a new complex number. The value of the complex number point is fixed when the mouse button is released.

Point on Object Tool

To create a point, which is fixed to an object, activate the tool first and then select the object. This new point can be moved via Move Tool, but only within the object.

Note: To put a point in the interior of a Circle or Ellipse you will need to increase the Opacity from 0 first. If you click on the perimeter of an object (e.g. Circle, Ellipse, Polygon), then the point will be fixed to the perimeter rather than the interior.

Intersect Tool

Intersection points of two objects can be created in two ways:

- Selecting two objects creates all intersection points (if possible).
- Directly clicking on an intersection of the two objects creates only this *single intersection point*.

Note: Sometimes it's useful to display only the portions of the intersecting objects near the intersection point.

To do so, open the Properties Dialog of the intersection point, and check the option *Show trimmed intersection lines* in the *Basic* tab of the *Properties* dialog of the object, then hide the intersecting objects.

Outlying Intersections

For segments, rays, or arcs you may specify whether you want to *Allow outlying intersections* on tab *Basic* of the Properties Dialog. This can be used to get intersection points that lie on the extension of an object. For example, the extension of a segment or a ray is a straight line.

Note: See also Intersect command.

Midpoint or Center Tool

You may click on either two points or one segment to get its midpoint. You can also click on a conic section (circle or ellipse) in order to create its center point.

Note: See also Center (, : Centre) and Midpoint commands.

Line Tools

Line tools are by default grouped under icon (the third from left) in the toolbar. Currently there are seven line tools:

- Line
- Segment
- Segment with Given Length
- Ray
- Polyline
- Vector
- Vector from Point

Line Tool

Selecting two points A and B creates a straight line through A and B .

Notes: The line's direction vector is $(B - A)$ See also Line CommandLine command.

Segment Tool

Select two points A and B in order to create a segment between A and B .

Note: In the Algebra View, the segment's length is displayed. See also Segment CommandSegment command.

Segment with Given Length Tool

Select the point that should be the starting point of the segment. Specify the desired length of the segment in the appearing window.

Note: This tool creates a segment with a specific length and an endpoint which may be rotated around the starting point by using the Move ToolMove tool. See also Segment CommandSegment command.

Ray Tool

Selecting two points A and B creates a ray starting at A through B .

Note: In the Algebra View the equation of the corresponding line is displayed. See also Ray CommandRay command.

Vector from Point Tool

Select a point A and a vector v to create the new point $B = A + v$ as well as the vector from A to B .

Note: See also Vector command.

Vector Tool

Select the starting point and then the end point of the vector.

Note: See also the Vector command.

Special Line Tools

Special line tools are by default grouped under icon (the fourth from the left) in the toolbar. Currently there are eight line tools:

- Perpendicular Line
- Parallel Line
- Perpendicular Bisector
- Angle Bisector
- Tangents
- Polar or Diameter Line
- Best Fit Line
- Locus

Best Fit Line Tool

Creates the best fit line for a set of points, chosen as follows :

- Creating a selection rectangle that contains all points.
- Selecting a list of points .

Note: See also FitLine command.

Parallel Line Tool

Selecting a line g and a point A defines a straight line through A parallel to g .

Notes: The line's direction is the direction of line g . See also Line CommandLine command.

Angle Bisector Tool

Angle bisectors can be defined in two ways:

- Selecting three points A , B , and C produces the angle bisector of the enclosed angle, where point B is the apex.
- Selecting two lines produces their two angle bisectors.

Notes: The direction vectors of all angle bisectors have length 1. See also AngleBisector CommandAngleBisector command.

Perpendicular Line Tool

Selecting a line (or a segment) and a point creates a straight line through that point perpendicular to the line (or segment).

Notes:

- The line's direction is equivalent to the perpendicular vector of the selected line or segment. (See also PerpendicularVector command).
- This tool has also a version applicable to 3D objects: Select point and perpendicular line or plane.
- See also PerpendicularLine command.

Tangents Tool

Tangents to a conic section can be produced in several ways (see also Tangent command):

- Selecting a point and a conic produces all tangents through the point to the conic.
- Selecting a line and a conic produces all tangents to the conic that are parallel to the selected line.
- Selecting a point and a function produces the tangent line to the function in its point having the same x-coordinate of the selected point (e.g. if the point is A , the tangent is drawn in $x = x(A)$).
- Selecting two circles produces the common tangents to them (up to 4).

Note: $x(A)$ represents the x -coordinate of point A . If point A lies on the function graph, the tangent runs through point A .

Note: Type rather than if you want a **conic** (parabola) rather than a **function**.

Polar or Diameter Line Tool

This tool creates the polar or diameter line of a conic section.

- Select a point and a conic section to get the polar line of the given point (relative to the given conic).
- Select a line or a vector and a conic section to get the conjugate diameter of the diameter parallel to the given line or vector (relative to the given conic).

Note: See also Polar command.

Perpendicular Bisector Tool

Click on either a segment (or interval) s or two points A and B in order to create a perpendicular bisector.

Notes: The bisector's direction is equivalent to the perpendicular vector of segment (or interval) s or AB . See also PerpendicularVector Command, PerpendicularVector command, See also PerpendicularBisector Command, PerpendicularBisector command.

Locus Tool

Select a point B that depends on another point A and whose locus should be drawn. Then, select point A to create the locus of point B .

Note: Point A has to be a point on an object (e.g. line, segment/interval, circle).

Example: Type $f(x) = x^2 - 2x - 1$ into the Input Bar and press the Enter-key. Place a new point A on the x -axis (see Point ToolPoint tool or Point CommandPoint command). Create point $B = (x(A), f(x(A)))$ that depends on point A . Select the tool Locus and successively select point B and point A . Drag point A along the x -axis to see point B moving along its locus line.

Warning: Locus is undefined, if the dependent point depends on Point Command with two parameters or PathParameter Command.

Note: See also Locus command.

Polygon Tools

Polygon tools are by default grouped under icon (the fifth from the left) in the toolbar. Currently there are four polygon tools:

- Polygon
- Regular Polygon
- Rigid Polygon
- Vector Polygon

Rigid Polygon Tool

Successively select at least three free points which will be the vertices of the polygon. Then, click the first point again in order to close the polygon. The resulting polygon will keep the shape: you can move it and rotate it by moving two vertices.

Holding down the Alt key when drawing a rigid polygon allows to get angles that are a multiple of 15° .

Notes: The polygon's area is displayed in the Algebra View. See also RigidPolygon CommandRigid Polygon command.

Vector Polygon Tool

Successively select at least three free points which will be the vertices of the polygon. Then, select the first point again in order to close the polygon. The resulting polygon will keep the shape when the first point is dragged, but the other vertices can be moved freely.

Holding down the Alt key when drawing a vector polygon allows to get angles that are a multiple of 15° .

Note: In the Algebra View, the polygon's area is displayed. See also [Polygon Command](#)[Polygon command](#).

Polyline Tool

Graphics View

Successively select at least three points which will be the vertices of the polyline. Finally select the starting point again to finish the polyline.

Holding down the Alt key when drawing a Polyline allows to get angles that are a multiple of 15° .

Note: In the Algebra View, the polyline length is displayed. See also [Polyline Command](#)[Polyline command](#)

Spreadsheet View

Select a set of spreadsheet cells where the selected cells form pairs across rows or columns. Then, click on the tool button to open a dialog for naming, modifying and creating a polyline.

Regular Polygon Tool

Select two points A and B and specify the number n of vertices in the input field of the appearing dialog window. This gives you a regular polygon with n vertices including points A and B .

Note: See also [Polygon command](#).

Polygon Tool

Successively select at least three points which will be the vertices of the polygon. Then, click the first point again in order to close the polygon.

Holding down the Alt key when drawing a Polygon allows to get angles that are a multiple of 15° .

Notes: The polygon area is displayed in the Algebra View. See also [Polygon Command](#)[Polygon command](#).

Circle & Arc Tools

Circle and arc tools are by default grouped under icon (the sixth from the left) in the toolbar. Currently there are nine circle and arc tools:

- Circle with Centre through Point
- Circle with Centre and Radius
- Compasses
- Circle through 3 Points
- Semicircle through 2 Points
- Circular Arc
- Circumcircular Arc
- Circular Sector
- Circumcircular Sector

Circle with Center and Radius Tool

Select the center point, then enter the radius in the text field of the appearing dialog window.

Note: See also [Circle command](#).

Circle through 3 Points Tool

Selecting three points defines a circle through these points.

Note: If the three points lie on the same line, the circle degenerates to this line. See also Circle CommandCircle command.

Circle with Center through Point Tool

Selecting a point M and a point P defines a circle with center M through P .

Note: See also Circle command.

Circumcircular Arc Tool

Select three points to create a circular arc through these points. Thereby, the first selected point is the starting point of the arc, the second one lies on the arc, and the third selected point is the endpoint of the arc.

Note: See also CircumcircularArc command.

Circumcircular Sector Tool

Select three points to create a circular sector through these points. Thereby, the first given point is the starting point of the sector's arc, the second one lies on the arc, and the third given point is the endpoint of the sector's arc.

Note: See also CircumcircularSector command.

Compass Tool

Select a segment or two points to specify the radius. Then, click on a point that should be the center of the new circle.

Circular Sector Tool

First, select the center point of the circular sector. Then, select the starting point of the sector's arc and another point that specifies the length of the sector's arc.

Notes: While the second selected point always lies on the sector's arc, the third selected point does not need to lie on it. See also CircularSector CommandCircularSector command.

Semicircle through 2 Points Tool

Select two points A and B to create a semicircle above the segment (or interval) AB .

The length of the semicircle is shown in *Algebra View*.

Note: See also Semicircle command.

Circular Arc Tool

First, select the center point of the circular arc. Then, select the starting point of the arc and another point that specifies the length of the arc.

Notes: The Arc is always drawn counter-clockwise. While the second selected point always lies on the circular arc, the third point does not need to lie on it. See also CircularArc CommandCircularArc command.

Conic Section Tools

Conic section tools are by default grouped under icon (the sixth from the right) in the toolbar. Currently there are four conic section tools:

- Ellipse
- Hyperbola
- Parabola
- Conic through 5 Points

Ellipse Tool

Select the two foci of the ellipse. Then, specify a third point that lies on the ellipse.

Note: See also Ellipse command.

Hyperbola Tool

Select the two foci of the hyperbola. Then, specify a third point that lies on the hyperbola.

Note: See also Hyperbola command.

Conic through 5 Points Tool

Selecting five points produces a conic section through these points.

Notes: If four of these five points lie on a line, the conic section is not defined. See also Conic CommandConic command.

Parabola Tool

Select a point (focus) and the directrix of the parabola, in any order.

Note:

- If you select the directrix line first, a preview of the resulting parabola is shown.
- See also Parabola command.

Measurement Tools

Measurement tools are by default grouped under icon (the fifth from the right) in the toolbar. Currently there are six measurement tools:

- Angle
- Angle with Given Size
- Distance or Length
- Area
- Slope
- List

Distance or Length Tool

This tool returns the distance between two points, two lines, or a point and a line as a number, and shows a dynamic text in the Graphics View. It can also be used to measure the length of a segment (or interval), the circumference of a circle, or the perimeter of a polygon.

Note: See also Distance and Length commands.

Angle Tool

With this tool you can create angles in different ways:

- Click on three points to create an angle between these points. The second point selected is the vertex of the angle.
- Click on two segments to create the angle between them.
- Click on two lines to create the angle between them.
- Click on two vectors to create the angle between them.
- Click on a polygon to create all angles of this polygon.

Notes: If the polygon was created by selecting its vertices in counter clockwise orientation, the Angle tool gives you the interior angles of the polygon. Angles are created in counter clockwise orientation. Therefore, the order of selecting these objects is relevant for the Angle tool. If you want to limit the maximum size of an angle select the desired limits from the drop-down list on tab Basic of the Properties Dialog. See also Angle CommandAngle command.

Slope Tool

By selecting a line, this tool gives you the slope of a line and shows a slope triangle in the Graphics View, whose size may be changed using Properties Dialog

Note: See also Slope command.

Area Tool

This tool gives you the area of a polygon, circle, or ellipse as a number and shows a dynamic text in the Graphics View.

Note: See also Area command.

Angle with Given Size Tool

Select a leg point, then the angle vertex and type the angle's size into the input box of the appearing window.

Notes: This tool creates a point C and an angle α , where α is the angle ABCSee also Angle CommandAngle command

Transformation Tools

Transformation tools are by default grouped under icon (the fourth from the right) in the toolbar. Currently there are six transformation tools: <links/>

Translate by Vector Tool

Select the object you want to translate. Then, click on the translation vector or click twice to make a vector. You can also just drag to clone an object with this tool.

Note: See also Translate command.

Reflect about Line Tool

Select the object you want to reflect. Then, click on a line to specify the mirror/line of reflection.

Note: See also Reflect command.

Reflect about Point Tool

Select the object you want to reflect. Then, click on a point to specify the mirror/point of reflection.

Note: See also Reflect command.

Rotate around Point Tool

Select the object you want to rotate. Then, click on a point to specify the center of rotation and enter the rotation angle into the text field of the appearing dialog window.

Note: See also Rotate command.

Reflect about Circle Tool

Inverts a geometric object about a circle. Select the object you want to invert. Then, click on a circle to specify the mirror/circle of inversion.

Note: See also Reflect command.

Dilate from Point Tool

Select the object to be dilated. Then, click on a point to specify the dilation center and enter the dilation factor into the text field of the appearing dialog window (see also Dilate command).

Special Object Tools

Special object tools are by default grouped under icon (the third from the right) in the toolbar. Currently there are six special object tools:

<links/>

Freehand Shape Tool

The Freehand Shape Tool lets you either sketch a function, or you can draw a freehand circle, segment or polygon and it will be recognized and converted to an exact shape. If a function f is created, you can compute its value at certain point, place a point on it or perform some transformations. Derivatives for these functions are not supported (tangents are supported as a numerical approximation)

Notes: You can use all the curve fitting commands on functions created by this Tool, e.g. FitSin_Command You can use the Integral_CommandIntegral command on functions created with this tool e.g. Integral(f, 1, 5) to shade under it and find the area

Image Tool

This tool allows you to insert an image into the Graphics View.

First, select the *Image* tool from the toolbox of *Special Object Tools* (third toolbox from the right).

Then, specify the desired location of the image in one of the following two ways:

- Click in the *Graphics View* to specify the position of the image's lower left corner.
- Click on a point to specify this point as the lower left corner of the image.

Then, a file-open dialog appears that allows you to select the image file from the files saved on your computer.

Notes:

- In the GeoGebra Desktop Version you can use the keyboard shortcut Alt-click, after selecting the tool **Image**, in order to paste an image directly from your computer's clipboard into the *Graphics View*.
- Transparent GIF and PNG files are supported, but for PNGs you may need to edit them first so that they have an alpha channel (for example using IrfanView^[1], save using the PNGOUT plugin and choose **RGB+Alpha**)
- Inserted images will be auxiliary objects by default.

Properties of Images

The position of an image may be absolute on screen or relative to the coordinate system. You can specify this on tab *Basic* of the Properties Dialog of the image.

You may specify up to three corner points of the image on tab *Position* of the *Properties Dialog*. This gives you the flexibility to scale, rotate, and even distort images (see also Corner command).

- *Corner 1*: position of the lower left corner of the image
- *Corner 2*: position of the lower right corner of the image

Note: This corner may only be set if *Corner 1* was set before. It controls the width of the image.

- *Corner 4*: position of the upper left corner of the image

Note: This corner may only be set if *Corner 1* was set before. It controls the height of the image.

Example: Create three points A , B , and C and insert an image into the *Graphics View* to explore the effects of the corner points. Set point A as the first and point B as the second corner of your image. By dragging points A and B in Move mode you can change the width of the image. Now, remove point B as the second corner of the image. Set point A as the first and point C as the fourth corner and explore how dragging the points now influences the height of the image. Finally, you may set all three corner points and see how dragging the points distorts your image.

Example: In order to attach your image to a point A , and set its width to 3 and its height to 4 units, you could do the following:
Set Corner 1 to A
Set Corner 2 to $A + (3, 0)$
Set Corner 4 to $A + (0, 4)$

Note: If you now drag point A in Move mode, the size of your image does not change.

You may specify an image as a *Background* image on tab *Basic* of the *Properties Dialog*. A background image lies behind the coordinate axes and cannot be selected with the mouse any more.

Note: In order to change the background setting of an image, you may open the *Properties Dialog* by selecting *Object Properties* from the Edit Menu.

The *Transparency* of an image can be changed in order to see objects or axes that lie behind the image. You can set the transparency of an image by specifying a *Filling* value between 0 % and 100 % on tab *Style* of the *Properties Dialog*.

References

[1] <http://www.irfanview.com/>

Pen Tool

The Pen Tool allows the user to add freehand notes and drawings to the *Graphics View*. This makes the Pen Tool particularly useful when using GeoGebra for presentations or with multimedia interactive whiteboards. To add a freehand note onto a region of the *Graphics View*, activate the tool and draw on your touch screen or with your mouse by holding the left mouse button

GeoGebra stores the notes you traced in the *Graphic View* as a *PenStroke* (assigning the name *stroke* to it). Therefore you can use several operations on them (e.g. move, rotate, reflect,...).

The default colour of the pen is black, but you can change the pen properties (colour, style, and thickness) using the Styling Bar.

Erasing

To erase a portion of your notes created in the *Graphics View* with the Pen Tool, press and hold the right mouse button while moving it on the notes you want to delete. Release the mouse button to finish.

In order to delete the *stroke* you previously created, erase it totally in the *Graphics View*, or use the Context Menu of the object, or just delete the corresponding item in Algebra View.

Slider Tool

Click on any free place in the Graphics View to create a slider for a number or an angle. The appearing dialog window allows you to specify the *Name*, *Interval [min, max]*, and *Increment* of the number or angle, as well as the *Alignment* and *Width* of the slider (in pixels), and its *Speed* and *Animation* modality.

Note: In the Slider dialog window you can enter a degree symbol $^{\circ}$ or *pi* (π) for the interval and increment by using the following keyboard shortcuts: Alt-O for the degree symbol $^{\circ}$ Alt-P for the pi symbol π

The position of a slider may be absolute in the *Graphics View* (this means that the slider is not affected by zooming, but always remains in the visible part of the *Graphics View*) or relative to the coordinate system (This can be changed afterwards in the Properties Dialog of the corresponding number or angle).

Note: In GeoGebra, a slider is the graphical representation of a Numbers and Angles#Free Numbers and Anglesfree number or free angle. You can easily create a slider for any existing Numbers and Angles#Free Numbers and Anglesfree number or angle by showing this object in the Graphics View (see Context Menu; see tool Show / Hide Object ToolShow/Hide Object).

Fixed sliders

Like other objects, sliders can be fixed. To translate a fixed slider when Move Tool is selected, you can drag it with your right mouse button. When **Slider Tool** is selected, you can use either left or right button. Sliders made with the **Slider Tool** are fixed by default. You can change value of fixed slider by simply clicking it.

Note: See also the Slider command.

Relation Tool

Select two objects to get information about their relation in a pop-up window (see also Relation command) .

Function Inspector Tool

Select the tool, then the function that you want to explore.

- In the tab *Interval* you can specify the interval, where the tool will find minimum, maximum, root, etc. of the function.
- In the tab *Points* several points of the function are given (step can be changed). Slope etc. can be found at these points.

Text Tool

With this tool you can create static and dynamic text or LaTeX formulas in the Graphics View.

At first, you need to specify the location of the text in one of the following ways:

- Click in the *Graphics View* to create a new text at this location.
- Select a point to create a new text that is attached to this point.

Note: You may specify the position of a text as absolute on screen or relative to the coordinate system on tab *Basic* of the Properties Dialog.

Then, a dialog appears where you may enter your text, which can be static, dynamic, or mixed.

The text you type directly in the *Edit* field is considered as static, i.e. it's not affected by the objects modifications. If you need to create a dynamic text, which displays the changing values of an object, select the related object from the *Objects* drop-down list (GeoGebra Desktop Version) or the tab in the *Advanced* menu (GeoGebra Web and Tablet Apps). The corresponding name is shown, enclosed in a grey box, in the *Edit* field. In the GeoGebra Desktop Version, right-clicking on the grey box allows you to select, if "Definition" or "Value" of the dynamic object is displayed.

It is also possible to perform algebraic operations or apply specific commands to these objects. Just select the grey box and type in the algebraic operation or GeoGebra text command desired. The results of these operations will be dynamically shown in the resulting text, in the *Graphics View*.

Best visual results are obtained when using LaTeX formatting for the formulas. Its use is simple and intuitive: just check the *LaTeX Formula* box, and select the desired formula template from the drop-down list. A variety of mathematical symbols and operators is also available in a drop-down-list (GeoGebra Desktop Version) respectively in a tab of the *Advanced* menu (GeoGebra Web and Tablet Apps Version).

Action Object Tools

These tools allow you to create Action Objects. They are by default grouped under icon (the second from the right in the *Graphics View Toolbar*) in the toolbar. Currently there are four action object tools:

<links/>

Check Box Tool

Clicking in the Graphics View creates a check box (see section Boolean values) that allows you to show and hide one or more objects. In the appearing dialog window you can specify which objects should be affected by the check box.

Note: You may select these objects from the list provided in the dialog window or select them with the mouse in any view.

Input Box Tool

Click in the Graphics View to insert an Input Box. In the appearing dialog you may set its caption and the linked object.

Note: See also InputBox command

Button Tool

Activate the tool and click in the Graphics View to insert a button. In the appearing dialog you may set its caption and OnClick script.

General Tools

General tools are by default grouped under icon (the first from the right) in the toolbar. Currently there are seven general tools:

<links/>

Custom Tools

GeoGebra allows you to create your own construction tools based on an existing construction. Once created, your custom tool can be used both with a pointing device and as a command in the Input Bar. All tools are automatically saved in your GeoGebra file.

Note: Outputs of the tool are not movable (i.e. you can't drag them with the mouse), even if they are defined as Point [<Path>]. In case you need movable output, you can define a list of commands and use it with Execute Command.

Creating custom tools

To create a custom tool, use the option Create new tool from Tools Menu.

Saving custom tools

When you save the construction as GGB file, all custom tools are stored in it. To save the tools in separate file(s) use the Tool Manager Dialog (option *Manage Tools* from Tools Menu).

Note: Custom tools will be saved as GGT file to distinguish them from normal GeoGebra files (GGB).

Accessing custom tools

If you open a new GeoGebra interface using item *New* from the File Menu, after you created a custom tool, it will still be part of the GeoGebra toolbar. However, if you open a new GeoGebra window (item *New Window* from the *File Menu*), or open GeoGebra on another day, your custom tools won't be part of the toolbar any more.

There are different ways of making sure that your user defined tools are displayed in the toolbar of a new GeoGebra window:

- After creating a new user defined tool you can save your settings using item *Save Settings* from the Options Menu. From now on, your customized tool will be part of the GeoGebra toolbar.

Note: You can remove the custom tool from the toolbar after opening item *Customize Toolbar* from the *Tools Menu*. Then, select your custom tool from the list of tools on the left hand side of the appearing dialog window and click the *Remove* button (GeoGebra Desktop Version) or drag and drop it to the right hand side (GeoGebra Web and Tablet Apps Version). Don't forget to save your settings after removing the custom tool.

Importing custom tools

- After saving your custom tool on your computer (as a GGT file), you can import it into a new GeoGebra window at any time. Just select item *Open* from the *File Menu* and open the file of your custom tool.

Note:Opening a GeoGebra tool file (GGT) in GeoGebra doesn't affect your current construction. It only makes this tool part of the current GeoGebra toolbar. You can also load GGT file by dragging it from file manager and dropping into GeoGebra window.

Show / Hide Label Tool

Click on an object to show or hide its label.

Zoom Out Tool

Click on any place in the Graphics View to zoom out.

Notes:The position of your click determines the center of zoom. See also *ZoomOut* Command*ZoomOut* command. See also *Zoom_In* Tool*Zoom In* tool. See also the *Customizing the Graphics View* section.

Zoom In Tool

Click on any place in the Graphics View to zoom in.

Notes:The position of your click determines the center of zoom. See also *ZoomIn* Command*ZoomIn* command. See also *Zoom_Out* Tool*Zoom Out* tool. See also the section *Customizing the Graphics View*.

Delete Tool

Click on any object you want to delete.

Notes: You can use the Edit Menu#Undo Undo button if you accidentally delete the wrong object See also Delete CommandDelete command.

Move Graphics View Tool

Move Graphics View Tool in the Graphics View

You may drag and drop the background of the Graphics View to change its visible area or scale each of the coordinate axes by dragging it with your pointing device.

Note: You can also move the background or scale each of the axes by pressing the Shift key (MS Windows: also Ctrl key) and dragging it with the mouse, no matter which Tool is selected.

Move Graphics View Tool in the 3D Graphics View

You may translate the three-dimensional coordinate system by dragging the background of the *3D Graphics View* with your pointing device. Thereby, you can switch between two modes by clicking on the background of the *3D Graphics View*:

- **Mode x-y-plane:** You may translate the scene parallel to the *x*-*y*-plane.
- **Mode z-axis:** You may translate the scene parallel to the *z*-axis.

Show / Hide Object Tool

Select the object you want to show or hide after activating this tool. Then, switch to another tool in order to apply the visibility changes to this object.

Note: When you activate this tool, all objects that should be hidden are displayed in the Graphics View highlighted. In this way, you can easily show hidden objects again by deselecting them before switching to another tool.

Copy Visual Style Tool

This tool allows you to copy visual properties (e.g. color, size, line style) from one object to one or more other objects. To do so, first select the object whose properties you want to copy. Then, select all other objects that should adopt these properties.

CAS Tools

The *CAS View Toolbar* is only available if the *CAS View* is active and provides the following *Tools*. For more information please check our complete list of all available *Tools* as well as the general information about *Toolbars*, creating *Custom Tools* and Customizing the *Toolbar*.

Evaluation Tools

- Evaluate Tool
- Numeric Tool
- Keep Input Tool

Calculation Tools

- Factor Tool
- Expand Tool
- Substitute Tool
- Solve Tool
- Solve Numerically Tool
- Derivative Tool
- Integral Tool

Analysis Tools

- Probability Calculator (GeoGebra Desktop)
- Function Inspector Tool (GeoGebra Desktop)

General Tools

- Delete Tool
-

Derivative Tool

You can use this tool in CAS View only.

Enter the expression you want to derive and press the Enter-key. Then mouse-click on the expression and choose the tool.

Evaluate Tool

You can use this tool in CAS View only.

After choosing the tool, enter the expression you want to evaluate and press the Enter-key.

Expand Tool

You can use this tool in CAS View only.

After choosing the tool, enter the expression you want to expand and press the Enter-key.

Factor Tool

You can use this tool in CAS View only.

Type the expression to factor and press Enter to confirm, then click on the entered expression and select the tool.

Integral Tool

You can use this tool in CAS View only.

Enter the expression you want to integrate and press the Enter-key. Then mouse-click on the expression and choose the tool.

Keep Input Tool

You can use this tool in CAS View only.

If you want your entered expression not to be changed in any way, choose the tool before entering it.

Numeric Tool

You can use this tool in CAS View only.

After choosing the tool, enter the expression you want a numerical approximation of and press the Enter-key.

Notes: The number of decimals depends on the global rounding you choose in the Options Menu. See also Numeric Command Numeric command.

Solve Tool

You can use this tool only in CAS View.

Type in the equation you want to solve and press the Enter-key. Then select the equation and activate the tool.

This tool can solve a system of equations as well. Write each equation in an extra cell, then choose all the cells and the tool. You will get the solution for each variable.

Note: See also the Solve command.

Solve Numerically Tool

This tool can be used only in CAS View.

Type in the equation you want to solve and press Enter to confirm. Then select the equation and activate the tool.

Note: This tool allows you to solve a system of equations as well. Type each equation in separate cells, then select them all and activate the tool to obtain the solution of the system.

Note: For non-polynomial equations, a starting value for the iteration will be added, eg $x = 1$. See the NSolve Command for more details.

Substitute Tool

You can use this tool in CAS View only.

Enter the expression and choose the tool. This opens a dialog-window, where you can determine, which expression you want to substitute.

Spreadsheet Tools

The *Spreadsheet View Toolbar* is only available if the *Spreadsheet View* is active and provides the following *Tools*. For more information please check our complete list of all available *Tools* as well as the general information about *Toolbars*, creating Custom Tools and Customizing the *Toolbar*.

Move Tools

- Move Tool

Data Analysis Tools

- One Variable Analysis Tool
- Two Variable Regression Analysis Tool
- Multiple Variable Analysis Tool
- Probability Calculator

List and Table Tools

- List Tool
- List of Points Tool
- Matrix Tool
- Table Tool
- PolyLine Tool

Calculation Tools

- Sum Tool
- Mean Tool
- Count Tool
- Maximum Tool
- Minimum Tool

One Variable Analysis Tool

Select a set of cells or columns with number data in the spreadsheet, then click this tool to open a dialog that creates graphs and calculates one-variable statistics from the data. The dialog has four panels: a statistics panel, a data panel and two graph panels. The data panel and second graph panel are not visible when the dialog is first opened. Use the options menu to show them.

Options

Drag and Drop

Multiple Variable Analysis Tool

Select two or more columns of data in the spreadsheet, then click this tool to open a dialog that creates graphs and calculates statistics from the data.

Two Variable Regression Analysis Tool

Select two columns with paired number data in the spreadsheet, then activate this tool to open a dialog that creates graphs and calculates two-variable statistics from the data. The dialog has four panels: a statistics panel, a data panel and two graph panels. Only one graph panel is visible when the dialog is first opened. Use the options menu to show the others.

The dialog opens with a scatterplot of the selected data. In the drop down menu below the graph you can select different regression models for the data. When a model is selected its graph is drawn on the plot and the equation is shown beneath.

Options

Drag and Drop

Count Tool

This tool is available only in the Spreadsheet View. There are two methods how to use it:

- Select a destination cell, then choose this tool and after that select a cell range. Count of selected cells will occur in destination cell.
- Select a cell range containing more than one cell and then choose this tool. If the cell range has multiple rows, count of selected cells each column will be inserted under this column. If the selection has only one row, count of the cells in this row will be inserted to the right of the selection. If you hold Shift when selecting the icon of this tool, the count of cells in each row of the selected area is computed.

Maximum Tool

This tool is available only in the Spreadsheet View. There are two methods how to use it:

- Select a destination cell, then choose this tool and after that select a cell range. Maximum of selected cells will occur in destination cell.
- Select a cell range containing more than one cell and then choose this tool. If the cell range has multiple rows, maximum of each column will be inserted under this column. If it has only one row, maximum of the row will be inserted to the right of the selection. If you hold Shift when selecting the icon of this tool, the maximum of each row in the selected area is computed.

Minimum Tool

This tool is available only in the Spreadsheet View. There are two methods how to use it:

- Select a destination cell, then choose this tool and after that select a cell range. Minimum of selected cells will occur in destination cell.
- Select a cell range containing more than one cell and then choose this tool. If the cell range has multiple rows, minimum of each column will be inserted under this column. If it has only one row, minimum of the row will be inserted to the right of the selection. If you hold Shift when selecting the icon of this tool, the minimum of each row in the selected area is computed.

Mean Tool

This tool is available only in the Spreadsheet View. There are two methods how to use it:

- Select a destination cell, then choose this tool and after that select a cell range. Mean of selected cells will occur in destination cell.
- Select a cell range containing more than one cell and then choose this tool. If the cell range has multiple rows, mean of each column will be inserted under this column. If it has only one row, mean of the row will be inserted to the right of the selection. If you hold Shift when selecting the icon of this tool, the mean of each row in the selected area is computed.

List Tool

Graphics View

In the Graphics View simply drag a rectangle around the objects you wish to put in the list. Then, select the tool button to create a list from the selected objects.

Spreadsheet View

In the Spreadsheet View select a set of cells. Then, select the tool button to open a dialog for naming, modifying and creating a list from the selected cells.

Note: Lists created by this tool also appear in the Algebra View

List of Points Tool

Select a set of spreadsheet cells where the selected cells form pairs across rows or columns. Then, click on the tool button to open a dialog for naming, modifying and creating a list of points from the selected cells.

Matrix Tool

Select a set of spreadsheet cells. Then, click on the tool button to open a dialog for naming, modifying and creating a matrix from the selected cells.

Sum Tool

This tool is available only in the Spreadsheet View. There are two methods how to use it:

- Select a destination cell, then choose this tool and after that select a cell range. Sum of selected cells will occur in destination cell.
- Select a cell range containing more than one cell and then choose this tool. If the cell range has multiple rows, sum of each column will be inserted under this column. If it has only one row, sum of the row will be inserted to the right of the selection. If you hold Shift when selecting the icon of this tool, sum of each row in selected area is computed.

Table Tool

Select a set of spreadsheet cells. Then, click on the tool button to open a dialog for naming, modifying and creating the table. (The table is created as a text in Graphics View.)

3D Graphics Tools

The *3D Graphics View Toolbar* is only available if the *3D Graphics View* is active and provides the following *Tools*. For more information please check our complete list of all available *Tools* as well as the general information about *Toolbars*, creating *Custom Tools* and Customizing the *Toolbar*.

Movement Tools

- Move Tool

Point Tools

- Point Tool
- Point on Object Tool
- Intersect Tool
- Midpoint or Center Tool
- Attach / Detach Point Tool

Line Tools

- Line Tool
- Segment Tool
- Segment with Given Length Tool
- Ray Tool
- Vector Tool
- Vector from Point Tool

Special Line Tools

- Perpendicular Line Tool
- Parallel Line Tool
- Angle Bisector Tool
- Tangents Tool
- Polar or Diameter Line Tool
- Locus Tool

Polygon Tools

- Polygon Tool

Circle, Arc, and Conics Tools

- Circle with Axis through Point Tool
- Circle with Center, Radius and Direction Tool
- Circle through 3 Points Tool
- Circumcircular Arc Tool
- Circumcircular Sector Tool
- Ellipse Tool
- Hyperbola Tool
- Parabola Tool
- Conic through 5 Points Tool

Intersection Tools

- Intersect Two Surfaces Tool

Plane Tools

- Plane through 3 Points Tool
- Plane Tool
- Perpendicular Plane Tool
- Parallel Plane Tool

Geometric Solids Tools

- Pyramid Tool
- Prism Tool
- Extrude to Pyramid or Cone Tool
- Extrude to Prism or Cylinder Tool
- Cone Tool
- Cylinder Tool
- Regular Tetrahedron Tool
- Cube Tool
- Net Tool

Sphere Tools

- Sphere with Center through Point Tool
- Sphere with Center and Radius Tool

Measurement Tools

- Angle Tool
- Distance or Length Tool
- Area Tool
- Volume Tool

Transformation Tools

- Reflect about Plane Tool
- Reflect about Line Tool
- Reflect about Point Tool
- Rotate around Line Tool
- Translate by Vector Tool
- Dilate from Point Tool

Special Objects Tools

- Text Tool

General Tools

- Rotate 3D Graphics View Tool
- Move Graphics View Tool
- Zoom In Tool
- Zoom Out Tool
- Show / Hide Object Tool
- Show / Hide Label Tool
- Copy Visual Style Tool
- Delete Tool
- View in front of Tool

Rotate 3D Graphics View Tool

Press the left mouse button and drag the 3D Graphics View to rotate it.

Note: You can also obtain the same result at any time by pressing the right mouse button and dragging the *3D Graphics View*.

Rotate around Line Tool

Select the object you want to rotate. Then, click on a line to specify the axis and enter an angle.

Note: See also Rotate command.

Plane Tool

Select three points, or point and line, or two lines, or a polygon to create a plane.

Note: See also Plane command.

Plane through 3 Points Tool

Select three points to create a plane.

Note: See also Plane command.

View in front of Tool

Select an object to move the point of view of the construction in front of the selected object.

Cone Tool

Select the center of the base, then the apex, and enter the radius in the text field of the appearing dialog window.

Note: See also Cone and InfiniteCone commands.

Circle with Axis through Point Tool

Select the axis (that may be a line, a ray or a segment), then a point on the circle.

Note: See also Circle command.

Circle with Center, Radius and Direction Tool

This tool returns a circle in different positions, depending on the object chosen as "direction" (line or plane).

- Selecting a line, a ray or a segment and a point on the circle, then typing the radius value in the related field of the appearing dialog window, GeoGebra generates a circle whose axis is directed as the object chosen as "direction".
- Selecting a plane (xOy plane or any other previously defined plane) and a point on the circle, then typing the radius value in the related field of the appearing dialog window, GeoGebra generates a circle in a plane parallel to the one chosen as "direction", through the given point.

Note: The order of selection of the first two objects is non influential. See also Circle CommandCircle command.

Sphere with Center through Point Tool

Select the center point, then another point to create a sphere.

Note: See also Sphere command.

Sphere with Center and Radius Tool

Select the center point and enter the radius in the text field of the appearing dialog window.

Note: See also Sphere command.

Volume Tool

Select a solid (pyramid, prism, sphere, cone, cylinder, etc.) to calculate its volume and to display a text containing its value.

Note: See also Volume command.

Cube Tool

Select two points to create a cube:

- Select two points in the xOy plane to obtain a cube lying on that plane.
- Select two points on the same side of $z=c$ to obtain a cube lying on the plane having $z=c$ as one of its equations.
- Select a plane and two points on this plane, or on a parallel plane, to obtain a cube lying on the last plane.

Note: See also Cube command.

Extrude to Prism or Cylinder Tool

In 3D Graphics View, left-click and drag a polygon or a circle to create a prism or cylinder.

Otherwise select a polygon or a circle, then enter the value of the height (as a number or formula) in the appearing dialog box to create a right prism or cylinder.

Extrude to Pyramid or Cone Tool

In 3D Graphics View, select and drag a polygon or a circle to create a pyramid or cone.

Otherwise select a polygon or a circle, then enter the value of the height (as a number or formula) in the appearing dialog box to create a right pyramid or cone, whose apex is above the centroid of the base.

Cylinder Tool

Select the center of the bottom base, then the center of the top base, and enter the radius in the text field of the appearing dialog window.

Note: See also Cylinder and InfiniteCylinder commands.

Intersect Two Surfaces Tool

Select two planes, or two spheres, or a plane and a solid (sphere, cube, prism, cone, cylinder, ...) to get their intersection curve if the two objects have points in common.

Note: See also Intersect command.

Reflect about Plane Tool

Select the object you want to reflect, then click on a plane to specify the mirror/plane of reflection.

Note: See also Reflect command.

Net Tool

Select a polyhedron to obtain its net into the plane containing its bottom base.

Note: A slider is also created (see the Algebra View and Graphics View) to manage the opening. See also Net CommandNet command.

Perpendicular Plane Tool

Select a point and a line to get the plane through the given point and perpendicular to the selected line.

Note: See also PerpendicularPlane command.

Parallel Plane Tool

Select a point and a plane to get the plane through the given point and parallel to the selected plane.

Prism Tool

Select or create a polygon as bottom base, then select or create a vertex of the top base.

Note: See also Prism command.

Pyramid Tool

Select or create a polygon as base, then select or create the apex point.

Note: See also Pyramid command.

Regular Tetrahedron Tool

Choose one of the following to create a regular tetrahedron:

- Select two points in the xOy plane to obtain a regular tetrahedron lying on that plane.
- Select two points on the same side of $z=c$ to obtain a regular tetrahedron lying on the plane having $z=c$ as one of its equations.
- Select a plane and two points on this plane or on a parallel plane, to obtain a regular tetrahedron lying on this last plane.

Note: See also Tetrahedron command.

Commands

Commands

Using commands you can produce new and modify existing objects. Please check the list displayed on the right, where commands have been categorized with respect to their field of application, or check the full commands list for further details.

Notes: Press the Enter key after every input to create the corresponding object. A command's result may be named by entering a label followed by an equal sign (=). In the example below, the new point is named S.

Example: To get the intersection point of two lines g and h you can enter $S = \text{Intersect}[g, h]$ (see Intersect Command).

- You can also use indices within the names of objects: A_1 is entered as A_1 while S_{AB} is created using S_{AB} . This is part of LaTeX syntax.

Geometry Commands

Currently there are following Geometry commands: <links/>

AffineRatio Command

AffineRatio(<Point A>, <Point B>, <Point C> **)**

Returns the affine ratio λ of three collinear points A, B and C , where $C = A + \lambda * AB$.

Example: `AffineRatio((-1, 1), (1, 1), (4, 1))` yields 2.5

Angle Command

Angle(<Object>)

- **Conic:** Returns the angle of twist of a conic section's major axis (see command Axes).

Example: `Angle(x^2/4+y^2/9=1)` yields 90° or 1.57 if the default angle unit is *radians*.

Note: It is not possible to change the Angle Unit to Radian in GeoGebra 5.0 Web and Tablet App Version.

- **Vector:** Returns the angle between the x -axis and given vector.

Example: `Angle(Vector((1, 1)))` yields 45° or the corresponding value in *radians*.

- **Point:** Returns the angle between the x -axis and the position vector of the given point.

Example: `Angle((1, 1))` yields 45° or the corresponding value in *radians*.

- **Number:** Converts the number into an angle (result in $[0,360^\circ]$ or $[0,2\pi]$ depending on the default angle unit).

Example: `Angle(20)` yields 65.92° when the default unit for angles is *degrees*.

- **Polygon:** Creates all angles of a polygon in mathematically positive orientation (counter clockwise).

Example: `Angle(Polygon((4, 1), (2, 4), (1, 1)))` yields 56.31° , 52.13° and 71.57° or the corresponding values in *radians*.

Note: If the polygon was created in counter clockwise orientation, you get the interior angles. If the polygon was created in clockwise orientation, you get the exterior angles.

Angle(<Vector>, <Vector>)

Returns the angle between two vectors (result in $[0,360^\circ]$ or $[0,2\pi]$ depending on the default angle unit).

Example:

`Angle(Vector((1, 1)), Vector((2, 5)))` yields 23.2° or the corresponding value in *radians*.

Angle(<Line>, <Line>)

Returns the angle between the direction vectors of two lines (result in $[0,360^\circ]$ or $[0,2\pi]$ depending on the default angle unit).

Example:

- `Angle(y = x + 2, y = 2x + 3)` yields 18.43° or the corresponding value in *radians*.
- `Angle(Line((-2, 0, 0), (0, 0, 2)), Line((2, 0, 0), (0, 0, 2)))` yields 90° or the corresponding value in *radians*.

and in *CAS View* :

- `Angle(x + 2, 2x + 3)` yields .
- Define `f(x) := x + 2` and `g(x) := 2x + 3` then command `Angle(f(x), g(x))` yields .

Angle(<Line>, <Plane>)

Returns the angle between the line and the plane.

Example:

- `Angle(Line((1, 2, 3), (-2, -2, 0)), z = 0)` yields 30.96° or the corresponding value in *radians*.

Angle(<Plane>, <Plane>)

Returns the angle between the two given planes.

Example:

- `Angle(2x - y + z = 0, z = 0)` yields 114.09° or the corresponding value in *radians*.

Angle(<Point>, <Apex>, <Point>)

Returns the angle defined by the given points (result in $[0,360^\circ]$ or $[0,2\pi]$ depending on the default angle unit).

Example:

`Angle((1, 1), (1, 4), (4, 2))` yields 56.31° or the corresponding value in radians.

Angle(<Point>, <Apex>, <Angle>)

Returns the angle of size α drawn from *point* with *apex*.

Example:

`:*Angle((0, 0), (3, 3), 30)` yields 30° and the point $(1.9, -1.1)$.

Note: The point *Rotate(<Point>, <Angle>, <Apex>)* is created as well.

Angle(<Point>, <Point>, <Point>, <Direction>)

Returns the angle defined by the points and the given *Direction*, that may be a line or a plane (result in $[0,360^\circ]$ or $[0,2\pi]$ depending on the default angle unit).

Note: Using a *Direction* allows to bypass the standard display of angles in 3D which can be set as just $[0,180^\circ]$ or $[180^\circ,360^\circ]$, so that given three points A, B, C in 3D the commands `Angle(A, B, C)` and `Angle(C, B, A)` return their real measure instead of the one restricted to the set intervals.

Example:

`Angle((1, -1, 0), (0, 0, 0), (-1, -1, 0), zAxis)` yields 270° and `Angle((-1, -1, 0), (0, 0, 0), (1, -1, 0), zAxis)` yields 90° or the corresponding values in radians.

Note: See also Angle and Angle with Given Size tools.

AngleBisector Command

AngleBisector(<Line>, <Line>)

Returns both angle bisectors of the lines.

Example: `AngleBisector(x + y = 1, x - y = 2)` yields *a*: $x = 1.5$ and *b*: $y = -0.5$.

AngleBisector(<Point>, <Point>, <Point>)

Returns the angle bisector of the angle defined by the three points.

Example: `AngleBisector((1, 1), (4, 4), (7, 1))` yields *a*: $x = 4$.

Note: The second point is apex of this angle.

Note: See also Angle Bisector tool .

Arc Command

Arc(<Circle>, <Point M >, <Point N>)

Returns the directed arc (counterclockwise) of the given circle, with endpoints M and N.

Arc(<Ellipse>, <Point M>, <Point N>)

Returns the directed arc (counterclockwise) of the given ellipse, with endpoints M and N.

Arc(<Circle>, <Parameter Value>, <Parameter Value>)

Returns the circle arc of the given circle, whose endpoints are identified by the specified values of the parameter.

Note: Internally the following parametric forms are used:

Circle: $(r \cos(t), r \sin(t))$ where r is the circle's radius.

Arc(<Ellipse>, <Parameter Value>, <Parameter Value>)

Returns the circle arc of the given ellipse, whose endpoints are identified by the specified values of the parameter.

Note: Internally the following parametric forms are used:

Ellipse: $(a \cos(t), b \sin(t))$ where a and b are the lengths of the semimajor and semiminor axes.

Note: See also CircumcircularArc command.

Area Command

Area(<Point>, ..., <Point>)

Calculates the area of the polygon defined by the given points.

Example: `Area((0, 0), (3, 0), (3, 2), (0, 2))` yields 6.

Area(<Conic>)

Calculates the area of a conic section (circle or ellipse).

Example: `Area(x^2 + y^2 = 2)` yields 6.28.

Area(<Polygon>)

Calculates the area of the polygon.

Notes: for Polygons, the absolute value of the Algebraic Area is calculated (which gives unexpected answers for self-intersecting polygons) In order to calculate the area between two function graphs, you need to use the command IntegralBetween CommandIntegralBetween. See also the Area ToolArea tool.

AreEqual Command

AreEqual(<Object>, <Object>)

Decides if the objects are equal.

Normally this command computes the result numerically. This behavior can be changed by using the Prove command.

Example: `AreEqual(Circle((0, 0), 1), x^2+y^2=1)` yields *true* since the two circles have the same center and radius.

Notes: `AreEqual(Segment((1, 2), (3, 4)), Segment((3, 4), (1, 6)))` is different from `Segment((1, 2), (3, 4)) == Segment((3, 4), (1, 6))` as the latter compares just the lengths. See also AreCollinear Command, AreConcyclic Command, AreConcurrent Command, AreCongruent Command, AreEqual Command, ArePerpendicular Command, AreParallel Command, IsTangent Command.

AreCollinear Command

AreCollinear(<Point>, <Point>, <Point>)

Decides if the points are collinear.

Normally this command computes the result numerically. This behavior can be changed by using the Prove command.

Example: `AreCollinear((1, 2), (3, 4), (5, 6))` yields *true* since all the three points lying on the same line.

Note: See also AreConcurrent, AreConcyclic, AreCongruent, AreEqual, ArePerpendicular, AreParallel, IsTangent commands.

AreConcyclic Command

AreConcyclic(<Point>, <Point>, <Point>, <Point>)

Decides if the points are concyclic.

Normally this command computes the result numerically. This behavior can be changed by using the Prove command.

Example: AreConcyclic((1, 2), (3, 4), (1, 4), (3, 2)) yields *true* since the points are lying on the same circle.

Note: See also AreCollinear, AreConcurrent, AreCongruent, AreEqual, ArePerpendicular, AreParallel, IsTangent commands.

AreCongruent Command

AreCongruent(<Object>, <Object>)

Decides if the objects are congruent.

Normally this command computes the result numerically. This behavior can be changed by using the Prove command.

Example: AreCongruent(Circle((0, 0), 1), x^2+y^2=1) and
AreCongruent(Circle((1, 1), 1), x^2+y^2=1) yield *true* since the two circles have the same radius.

Note: See also AreEqual, AreCollinear, AreConcyclic, AreConcurrent, ArePerpendicular, AreParallel, IsTangent commands.

AreConcurrent Command

AreConcurrent(<Line>, <Line>, <Line>)

Decides if the lines are concurrent. If the lines are parallel, they considered to have a common point in infinity, thus this command returns *true* in this case.

Normally this command computes the result numerically. This behavior can be changed by using the Prove command.

Example: AreConcurrent(Line((1, 2), (3, 4)), Line((1, 2), (3, 5)), Line((1, 2), (3, 6))) yields *true* since all three lines contain the point (1,2).

Note: See also AreCollinear, AreConcyclic, AreCongruent, AreEqual, ArePerpendicular, AreParallel, IsTangent commands.

ArePerpendicular Command

ArePerpendicular(<Line>, <Line>)

Decides if the lines are perpendicular.

Normally this command computes the result numerically. This behavior can be changed by using the Prove command.

Example: ArePerpendicular(Line((-1, 0), (0, -1)), Line((0, 0), (2, 2))) yields *true* since the given lines are perpendicular.

Note: See also AreCollinear, AreConcurrent, AreConcyclic, AreCongruent, AreEqual, AreParallel, IsTangent commands.

AreParallel Command

AreParallel(<Line>, <Line>)

Decides if the lines are parallel.

Normally this command computes the result numerically. This behavior can be changed by using the Prove command.

Example: `AreParallel(Line[(1, 2), (3, 4)], Line((5, 6), (7, 8)))` yields *true* since the given lines are parallel.

Note: See also AreCollinear, AreConcurrent, AreCongruent, AreConcyclic, AreEqual, ArePerpendicular, IsTangent commands.

Barycenter Command

Barycenter(<List of Points>, <List of Weights>)

Set the center of a system of points in the list, defined as the average of their positions, weighted by their value, using the proper formula.

Examples:

- `Barycenter({(2, 0), (0, 2), (-2, 0), (0, -2)}, {1, 1, 1, 1})` yields point $A(0, 0)$
- `Barycenter({(2, 0), (0, 2), (-2, 0), (0, -2)}, {2, 1, 1, 1})` yields point $B(0.4, 0)$. The x -coordinate of this point was determined by $= 0.4$

Centroid Command

Centroid(<Polygon>)

Returns the centroid of the polygon.

Example:

Let $A = (1, 4)$, $B = (1, 1)$, $C = (5, 1)$ and $D = (5, 4)$ be the vertices of a polygon.
Polygon(A, B, C, D) yields $\text{poly1} = 12$. Centroid(poly1) yields the centroid $E = (3, 2.5)$.

CircularArc Command

CircularArc(<Midpoint>, <Point A>, <Point B>)

Creates a circular arc with midpoint between the two points.

Notes: The arc length is displayed in Algebra View. Point B does not have to lie on the arc. See also Circular Arc Tool/Circular Arc tool.

CircularSector Command

CircularSector(<Midpoint>, <Point A>, <Point B>)

Creates a circular sector with midpoint between the two points.

Notes: The sector area is displayed in Algebra View. Point B does not have to lie on the arc of the sector. See also Circular Sector Tool/Circular Sector tool.

CircumcircularArc Command

`CircumcircularArc(<Point>, <Point>, <Point>)`

Creates a circular arc through three points, where the first point is the starting point and the third point is the endpoint of the circumcircular arc.

Note: See also Circumcircular Arc tool.

CircumcircularSector Command

`CircumcircularSector(<Point>, <Point>, <Point>)`

Creates a circular sector whose arc runs through the three points, where the first point is the starting point and the third point is the endpoint of the arc.

Note: See also Circumcircular Sector through Three Points tool.

Circumference Command

`Circumference(Conic)`

If the given conic is a circle or ellipse, this command returns its circumference. Otherwise the result is undefined.

Example: `Circumference (x^2 + 2y^2 = 1)` yields 5.4.

Note: See also Perimeter command.

ClosestPoint Command

`ClosestPoint(<Path>, <Point>)`

Returns a new point on a path which is the closest to a selected point.

Note: For Functions, this command now uses closest point (rather than vertical point). This works best for polynomials; for other functions the numerical algorithm is less stable.

`ClosestPoint(<Line>, <Line>)`

Returns a new point on the first line which is the closest to the second line.

CrossRatio Command

`CrossRatio(<Point A>, <Point B>, <Point C>, <Point D>)`

Calculates the cross ratio λ of four collinear points A, B, C and D , where:

$$\lambda = \text{AffineRatio}[B, C, D] / \text{AffineRatio}[A, C, D].$$

Example: `CrossRatio((-1, 1), (1, 1), (3, 1), (4, 1))` yields 1.2

Cubic Command

`Cubic(<Point>, <Point>, <Point>, <Number>)`

Gives n -th triangle cubic ^[1] of the given triangle ABC .

Example:

Let $A = (0, 1)$, $B = (2, 1)$ and $C = (1, 2)$.

`Cubic(A, B, C, 2)` yields the implicit curve $-x^3 + 3x^2 + 5x y^2 - 14x y + 7x - 5y^2 + 14y = 9$.

Note: This command is in development, set of supported index n is changing.

Some common triangle cubics

Index n	Cubic
1	Neuberg Cubic
2	Thomson Cubic
3	McCay Cubic
4	Darboux Cubic
5	Napoleon/Feuerbach Cubic
7	Lucas Cubic
17	1st Brocard Cubic
18	2nd Brocard Cubic

References

[1] <https://bernard-gibert.pagesperso-orange.fr/ctc.html>

Direction Command

Direction(<Line>)

Yields the direction vector of the line.

Example: `Direction(-2x + 3y + 1 = 0)` yields the vector

Note: A line with equation $ax + by = c$ has the direction vector $(b, -a)$.

Distance Command

Distance(<Point>, <Object>)

Yields the shortest distance between a point and an object.

Example: `Distance((2, 1), x^2 + (y - 1)^2 = 1)` yields 1

Note: The command works for points, segments, lines, conics, functions and implicit curves. For functions it uses a numerical algorithm which works better for polynomials. Example: Let f be a function and A be a point. `Distance(A, f)` yields the distance between A and $(x(A), f(x(A)))$.

Distance(<Line>, <Line>)

Yields the distance between two lines.

Examples:

- `Distance(y = x + 3, y = x + 1)` yields 1.41
- `Distance(y = 3x + 1, y = x + 1)` yields 0

Note: The distance between intersecting lines is 0. Thus, this command is only interesting for parallel lines.

Note: See also Distance or Length tool .

Distance(<Point>, <Point>)

Yields the distance between the two points.

Example: `Distance((2, 1, 2), (1, 3, 0))` yields 3

Distance(<Line>, <Line>)

Yields the distance between two lines.

Example: Let $a: X = (-4, 0, 0) + \lambda*(4, 3, 0)$ and $b: X = (0, 0, 0) + \lambda*(0.8, 0.6, 0)$. `Distance(a, b)` yields 2.4

Envelope Command

`Envelope(<Path>, <Point>)`

Creates the envelope equation of a set of output paths while the moving point is bound to another object.

An envelope is a curve that is tangent to each member of the family of the output paths at some point.

Example:

A ladder is leaning against the wall and sliding down. ^[1]

The contour of its trace will be the envelope of the ladder. Strictly speaking, GeoGebra computes the envelope of the entire line containing the ladder as a segment. Only such envelopes can be computed where the appropriate construction leads to an algebraic equation system.

Note: See also Locus, LocusEquation commands and GeoGebra Automated Reasoning Tools: A Tutorial ^[2].

References

[1] <http://www.geogebra.org/student/m67909>

[2] <https://github.com/kovzol/gg-art-doc/tree/master/pdf/english.pdf>

Intersect Command

`Intersect(<Object>, <Object>)`

Yields the intersection points of two objects.

Examples:

- Let $a: -3x + 7y = -10$ be a line and $c: x^2 + 2y^2 = 8$ be an ellipse. `Intersect(a, c)` yields the intersection points $E = (-1.02, -1.87)$ and $F = (2.81, -0.22)$ of the line and the ellipse.
- `Intersect(y = x + 3, Curve(t, 2t, t, 0, 10))` yields $A = (3, 6)$.
- `Intersect(Curve(2s, 5s, s, -10, 10), Curve(t, 2t, t, -10, 10))` yields $A = (0, 0)$.

`Intersect(<Object>, <Object>, <Index of Intersection Point>)`

Yields the n^{th} intersection point of two objects. Each object must be a line, conic, polynomial function or implicit curve.

Example:

Let $a(x) = x^3 + x^2 - x$ be a function and $b: -3x + 5y = 4$ be a line. `Intersect(a, b, 2)` yields the intersection point $C = (-0.43, 0.54)$ of the function and the line.

`Intersect(<Object>, <Object>, <Initial Point>)`

Yields an intersection point of two objects by using a numerical, iterative method with initial point.

Example:

Let $a(x) = x^3 + x^2 - x$ be a function, $b: -3x + 5y = 4$ be a line, and $C = (0, 0.8)$ be the initial point. `Intersect(a, b, C)` yields the intersection point $D = (-0.43, 0.54)$ of the function and the line by using a numerical, iterative method.

`Intersect(<Function>, <Function>, <Start x-Value>, <End x-Value>)`

Yields the intersection points numerically for the two functions in the given interval.

Example:

Let $f(x) = x^3 + x^2 - x$ and $g(x) = 4 / 5 + 3 / 5 x$ be two functions. `Intersect(f, g, -1, 2)` yields the intersection points $A = (-0.43, 0.54)$ and $B = (1.1, 1.46)$ of the two functions in the interval $[-1, 2]$.

`Intersect(<Curve 1>, <Curve 2>, <Parameter 1>, <Parameter 2>)`

Finds one intersection point using a numerical, iterative method starting at the given parameters.

Example:

Let $a = \text{Curve}(\cos(t), \sin(t), t, 0, \pi)$ and $b = \text{Curve}(\cos(t) + 1, \sin(t), t, 0, \pi)$.

`Intersect(a, b, 0, 2)` yields the intersection point $A = (0.5, 0.87)$.

CAS Syntax

`Intersect(<Function>, <Function>)`

Yields a list containing the intersection points of two objects.

Example:

Let $f(x) := x^3 + x^2 - x$ and $g(x) := x$ be two functions. `Intersect(f(x), g(x))` yields the intersection points list: $\{(1, 1), (0, 0), (-2, -2)\}$ of the two functions.

`Intersect(<Object>, <Object>)`

Examples:

- `Intersect(<Line> , <Object>)` creates the intersection point(s) of a line and a plane, segment, polygon, conic, etc.
- `Intersect(<Plane> , <Object>)` creates the intersection point(s) of a plane and segment, polygon, conic, etc.
- `Intersect(<Conic> , <Conic>)` creates the intersection point(s) of two conics
- `Intersect(<Plane> , <Plane>)` creates the intersection line of two planes
- `Intersect(<Plane> , <Polyhedron>)` creates the polygon(s) intersection of a plane and a polyhedron.
- `Intersect(<Sphere> , <Sphere>)` creates the circle intersection of two spheres
- `Intersect(<Plane> , <Quadric>)` creates the conic intersection of the plane and the quadric (sphere, cone, cylinder, ...)

Notes:

- to get all the intersection points in a list you can use eg `{Intersect(a,b)}`
- See also `IntersectConic` and `IntersectPath` commands.
- See also `Intersect` tool.

IntersectPath Command

IntersectPath(<Line>, <Polygon>)

Creates the intersection path between line and polygon.

Example: IntersectPath(*a*, triangle) creates a segment between the first and second intersection point of line *a* and polygon *triangle*.

IntersectPath(<Polygon>, <Polygon>)

Creates the intersection polygon between two given polygons.

Example: IntersectPath(quadrilateral, triangle) creates a new polygon as intersection of the two given polygons.

Note: The new polygon can either be a quadrilateral, a pentagon or a hexagon. This depends on the position of the vertices of the given polygons.

IntersectPath(<Plane>, <Polygon>)

Creates the intersection path between plane and polygon.

Example: IntersectPath(*a*, triangle) creates a segment between the first and second intersection point of plane *a* and polygon *triangle* in the plane of the polygon.

IntersectPath(<Plane>, <Quadric>)

Creates the intersection path between plane and quadric.

Example: IntersectPath(*a*, sphere) creates a circle as intersection between plane *a* and quadric *sphere*.

Note: See also Intersect and IntersectConic commands.

Length Command

Length(<Object>)

Yields the length of the object.

Examples:

- Length(<Vector>) yields the length of the vector.
- Length(<Point>) yields the length of the position vector of the given point.
- Length(<List>) yields the length of the list, which is the number of elements in the list.
- Length(<Text>) yields the number of characters in the text.
- Length(<Locus>) returns the number of points that the given locus is made up of. Use Perimeter(Locus) to get the length of the locus itself. For details see the article about First Command.
- Length(<Arc>) returns the *arc length* (i.e. just the length of the curved section) of an arc or sector.

Length(<Function>, <Start x-Value>, <End x-Value>)

Yields the length of the function graph in the given interval.

Example: Length(2x, 0, 1) returns 2.236067977, about .

Length(<Function>, <Start Point>, <End Point>)

Yields the length of the function graph between the two points.

Note: If the given points do not lie on the function graph, their x-coordinates are used to determine the interval.

Length(<Curve>, <Start t-Value>, <End t-Value>)

Yields the length of the curve between the two values **of the parameter**.

Length(<Curve>, <Start Point>, <End Point>)

Yields the length of the curve between the two points that lie on the curve.

CAS Syntax

Length(<Function>, <Start x-Value>, <End x-Value>)

Calculates the length of a function graph between the two points.

Example: Length(2 x, 0, 1) yields .

Length(<Function>, <Variable>, <Start x-Value>, <End x-Value>)

Calculates the length of a function graph from *Start Point* to *End Point*.

Example: Length(2 a, a, 0, 1) yields .

Note: See also Distance or Length tool.

Line Command

`Line(<Point>, <Point>)`

Creates a line through two points A and B .

`Line(<Point>, <Parallel Line>)`

Creates a line through the given point parallel to the given line.

`Line(<Point>, <Direction Vector>)`

Creates a line through the given point with direction vector v .

Note: See also Line and Parallel Line tools.

PerpendicularBisector Command

`PerpendicularBisector(<Segment>)`

Yields the perpendicular bisector of a segment.

`PerpendicularBisector(<Point>, <Point>)`

Yields the perpendicular bisector of a line segment between two points.

`PerpendicularBisector(<Point>, <Point>, <Direction>)`

Yields the perpendicular bisector of a line segment between two points which is perpendicular to the direction.
 $<\text{Direction}>$ can either be a vector, an axis, a line or a segment.

Note: See also Perpendicular Bisector tool.

Locus Command

Locus(<Point Creating Locus Line Q>, <Point P>)

Returns the locus curve of the point Q , which depends on the point P .

Note: Point P needs to be a point on an object (e. g. line, segment, circle).

Locus(<Point Creating Locus Line Q>, <Slider t>)

Returns the locus curve of the point Q , which depends on the values assumed by the slider t .

Locus(<Slopefield>, <Point>)

Returns the locus curve which equates to the slopefield at the given point.

Locus(<f(x, y)>, <Point>)

Returns the locus curve which equates to the solution of the differential equation in the given point. The solution is calculated numerically.

Loci are specific object types, and appear as auxiliary objects. Besides Locus command, they are the result of some Discrete Math Commands and SolveODE Command. Loci are paths and can be used within path-related commands such as Point. Their properties depend on how they were obtained, see e.g. Perimeter Command and First Command.

Note: See also Locus tool.

Warning: A locus is undefined when the dependent point is the result of a Point Command with two parameters, or a PathParameter Command.

LocusEquation Command

LocusEquation(<Locus>)

Calculates the equation of a Locus and plots this as an Implicit Curve.

LocusEquation(<Point Creating Locus Line Q>, <Point P>)

Calculates the equation of a Locus by using inputs tracer point Q and mover point P , and plots this as an Implicit Curve.

Example:

Let us construct a parabola as a locus: Create free Points A and B , and Line d lying through them (this will be the directrix of the parabola). Create free point F for the focus. Now create P on Line d (the mover point), then create line p as a perpendicular line to d through P . Also create line b as perpendicular bisector of Line Segment FP . Finally, point Q (the point creating locus line) is to be created as intersection of Lines p and b . Now `LocusEquation(Q, P)` will find and plot the exact equation of the locus.

LocusEquation(<Boolean Expression>, <Free Point>)

Calculates the locus of the free point such that the boolean condition is satisfied.

Example: `LocusEquation(AreCollinear(A, B, C), A)` for free points A, B, C calculates the set of positions of A that make A, B and C collinear—i.e. a line through B and C .

Notes:

- The Locus must be made from a Point (not from a Slider)
- Works only for a restricted set of geometric loci, i.e. using points, lines, circles, conics. (Rays and line segments will be treated as (infinite) lines.)
- If the locus is too complicated then it will return 'undefined'.
- If there is no locus then the implicit curve is the empty set $0=1$.

- If the locus is the whole plane then the implicit curve is the equation $0=0$.
- The calculation is done using Gröbner bases, so sometimes extra branches of the curve will appear that were not in the original locus.
- Further informations and examples on geogebra.org^[1]. A collection of implicit locus examples^[2] is also available.
- See also Locus command and GeoGebra Automated Reasoning Tools: A Tutorial^[2].

References

- [1] <https://www.geogebra.org/m/KZVzqVEM>
[2] <https://www.geogebra.org/m/mbXQuvUV>

Midpoint Command

Midpoint(<Segment>)

Returns the midpoint of the segment.

Example: Let $s = \text{Segment}((1, 1), (1, 5))$. $\text{Midpoint}(s)$ yields $(1, 3)$.

Midpoint(<Conic>)

Returns the center of the conic.

Example: $\text{Midpoint}(x^2 + y^2 = 4)$ yields $(0, 0)$.

Midpoint(<Interval>)

Returns the midpoint of the interval (as number).

Example: $\text{Midpoint}(2 < x < 4)$ yields 3 .

Midpoint(<Point>, <Point>)

Returns the midpoint of two points.

Example: $\text{Midpoint}((1, 1), (5, 1))$ yields $(3, 1)$.

Midpoint(<Quadric>)

Returns the midpoint of the given quadric (e.g. sphere, cone, etc.)

Example: $\text{Midpoint}(x^2 + y^2 + z^2 = 1)$ yields $(0, 0, 0)$.

Note: See also Midpoint or Center tool.

PerpendicularLine Command

PerpendicularLine(<Point>, <Line>)

Creates a line through the point perpendicular to the given line.

Example:

Let $c: -3x + 4y = -6$ be a line and $A = (-2, -3)$ a point. `PerpendicularLine(A, c)` yields the line $d: -4x - 3y = 17$.

Note: For 3D objects a third argument is added to this command to specify the behavior: if 2D view is active, plane $z=0$ is used as third argument, if 3D view is active, *space* is used instead. See `PerpendicularLine(<Point>, <Line>, <Context>)` further below for details.

PerpendicularLine(<Point>, <Segment>)

Creates a line through the point perpendicular to the given segment.

Example:

Let c be the segment between the two points $A = (-3, 3)$ and $B = (0, 1)$. `PerpendicularLine(A, c)` yields the line $d: -3x + 2y = 15$.

PerpendicularLine(<Point>, <Vector>)

Creates a line through the point perpendicular to the given vector.

Example:

Let $u = \text{Vector}((5, 3), (1, 1))$ and $A = (-2, 0)$ a point. `PerpendicularLine(A, u)` yields the line $c: 2x + y = -4$.

PerpendicularLine(<Point>, <Plane>)

Creates a perpendicular line to the plane through the given point.

PerpendicularLine(<Line> , <Line>)

Creates a perpendicular line to the given lines through the intersection point of the two lines.

PerpendicularLine(<Point>, <Direction>, <Direction>)

Creates a perpendicular line to the given directions (that can be lines or vectors) through the given point.

PerpendicularLine(<Point>, <Line>, <Context>)

Creates a perpendicular line to the line through the point and depending on the context.

- `PerpendicularLine(<Point>, <Line>, <Plane>)` creates a perpendicular line to the given line through the point and parallel to the plane.
- `PerpendicularLine(<Point>, <Line>, space)` creates a perpendicular line to the given line through the point. The two lines have an intersection point. This command yields undefined if the point is on the line in 3D.

Note: See also Perpendicular Line tool.

Perimeter Command

Perimeter(<Polygon>)

Returns the perimeter of the polygon.

Example: `Perimeter(Polygon((1, 2), (3, 2), (4, 3)))` yields 6.58.

Perimeter(<Conic>)

If the given conic is a circle or ellipse, this command returns its perimeter. Otherwise the result is undefined.

Example: `Perimeter(x^2 + 2y^2 = 1)` yields 5.4.

Note: See also Circumference command.

Perimeter(<Locus>)

If the given locus is finite, this command returns its approximate perimeter. Otherwise the result is undefined.

Point Command

Point(<Object>)

Returns a point on the geometric object. The resulting point can be moved along the path.

Point(<Object>, <Parameter>)

Returns a point on the geometric object with given path parameter.

Point(<Point>, <Vector>)

Creates a new point by adding the vector to the given point.

Point(<List>)

Converts a list containing two numbers into a Point.

Example: `Point({1, 2})` yields (1, 2).

Notes:

- See also Point tool.
- See also Points and vectors

PointIn Command

PointIn(<Region>)

Returns a point restricted to given region.

Note: See also Attach / Detach Point Tool.

Polyline Command

Polyline(<List of Points>)

Creates an open polygonal chain (i.e. a connected series of segments) having the initial vertex in the first point of the list, and the final vertex in the last point of the list.

Note: The polygonal chain length is displayed in the Algebra View.

Polyline(<Point>, ..., <Point>)

Creates an open polygonal chain (i.e. a connected series of segments) having the initial vertex in the first entered point, and the final vertex in the last entered point.

Notes: The polygonal chain length is displayed in the Algebra View. It is also possible to create a discontinuous polygonal: Example: Polyline((1, 3), (4, 3), (?,?), (6, 2), (4, -2), (2, -2)) yields the value 9.47 in Algebra View, and the corresponding polygonal in Graphics View.

Note: See also Polygon command.

Polygon Command

Polygon(<Point>, ..., <Point>)

Returns a polygon defined by the given points.

Example: Polygon ((1, 1), (3, 0), (3, 2), (0, 4)) yields a quadrilateral.

Polygon(<Point>, <Point>, <Number of Vertices>)

Creates a regular polygon with n vertices.

Example: Polygon ((1, 1), (4, 1), 6) yields a hexagon.

Polygon(<List of Points>)

Returns a polygon defined by the points in the list.

Example: Polygon ({(0, 0), (2, 1), (1, 3)}) yields a triangle.

Polygon(<Point>, <Point>, <Number of Vertices n>, <Direction>)

Creates a regular polygon with n vertices, and directed by the *direction* (e.g. a plane to which the polygon will be parallel, if possible).

Note: See also Polygon and Regular Polygon tools.

Prove Command

Prove(<Boolean Expression>)

Returns whether the given boolean expression is true or false in general.

Normally, GeoGebra decides whether a boolean expression is true or not by using numerical computations. However, the Prove command uses symbolic methods to determine whether a statement is *true* or *false* in general. If GeoGebra cannot determine the answer, the result is *undefined*.

Example:

We define three free points, $A = (1, 2)$, $B = (3, 4)$, $C = (5, 6)$. The command `AreCollinear(A, B, C)` yields *true*, since a numerical check is used on the current coordinates of the points. Using `Prove(AreCollinear(A, B, C))` you will get *false* as an answer, since the three points are not collinear in general, i.e. when we change the points.

Example:

Let us define a triangle with vertices A , B and C , and define $D = \text{MidPoint}(B, C)$, $E = \text{MidPoint}(A, C)$, $p = \text{Line}(A, B)$, $q = \text{Line}(D, E)$. Now both $p \parallel q$ and `Prove(p || q)` yield *true*, since a midline of a triangle will always be parallel to the appropriate side. <ggb_applet width="525" height="366" version="5.0" id="40121" enableRightClick="false" showAlgebraInput="false" enableShiftDragZoom="false" showMenuBar="false" showToolBar="false" showToolBarHelp="true" enableLabelDrags="false" showResetIcon="false"/>

Note: See also `ProveDetails` command, Boolean values, GeoGebra Automated Reasoning Tools: A Tutorial ^[2] and technical details of the algorithms ^[1].

References

[1] <http://dev.geogebra.org/trac/wiki/TheoremProving>

ProveDetails Command

`ProveDetails(<Boolean Expression>)`

Returns some details of the result of the automated proof.

Normally, GeoGebra decides whether a boolean expression is true or not by using numerical computations. However, the ProveDetails command uses symbolic methods to determine whether a statement is true or false in general. This command works like the Prove command, but also returns some details of the result as a list:

- An empty list {} if GeoGebra cannot determine the answer.
- A list with one element: {**false**}, if the statement is not true in general.
- A list with one element: {**true**}, if the statement is always true (in all cases when the diagram can be constructed).
- A list with more elements, containing the boolean value *true* and another list for some so-called *non-degeneracy conditions*, if the statement is true under certain conditions, e.g. {*true*, {"AreCollinear(A,B,C),AreEqual(C,D)"}}. This means that if none of the conditions are true (and the diagram can be constructed), then the statement will be *true*.
- A list {**true**,{"..."}}, if the statement is true under certain conditions, but these conditions cannot be translated to human readable form for some reasons.

Example:

Let us define a triangle with vertices *A*, *B* and *C*, and define $D=\text{MidPoint}(B,C)$, $E=\text{MidPoint}(A,C)$, $p=\text{Line}(A,B)$, $q=\text{Line}(D,E)$. Now `ProveDetails(p||q)` returns {*true*}, which means that if the diagram can be constructed, then the midline *DE* of the triangle is parallel to the side *AB*.

Example:

Let *AB* be the segment *a*, and define $C=\text{MidPoint}(A,B)$, $b=\text{PerpendicularBisector}(A,B)$, $D=\text{Intersect}(a,b)$. Now `ProveDetails(C==D)` returns {*true*, {"*AreEqual(A,B)*"}}: it means that if the points *A* and *B* differ, then the points *C* and *D* will coincide.

Example:

Let *AB* be the segment *a*, and define $l=\text{Line}(A,B)$. Let *C* be an arbitrary point on line *l*, moreover let $b=\text{Segment}(B,C)$, $c=\text{Segment}(A,C)$. Now `ProveDetails(a==b+c)` returns {*true*, {"*a+b==c*", "*b==a+c*"}}: it means that if neither , nor , then .

It is possible that the list of the non-degeneracy conditions is not the simplest possible set. For the above example, the simplest set would be the empty set.

Note: See also Prove command, Boolean values, GeoGebra Automated Reasoning Tools: A Tutorial ^[2] and technical details of the algorithms ^[1].

Radius Command

Radius(<Conic>)

Returns the radius of a conic.

Examples:

- Returns the radius of a circle c (e.g. $c:(x - 1)^2 + (y - 1)^2 = 9$) Radius (c) yields $a = 3$.
- Returns the radius of a circle formula Radius ($(x - 2)^2 + (y - 2)^2 = 16$) yields $a = 4$.

Ray Command

Ray(<Start Point>, <Point>)

Creates a ray starting at a point through a point.

Ray(<Start Point>, <Direction Vector>)

Creates a ray starting at the given point which has the direction vector.

Notes: When computing intersections with other objects, only intersections lying on the ray are considered. To change this, you can use Intersect Tool#Outlying_IntersectionsOutlying Intersections option. See also Ray ToolRay tool.

RigidPolygon Command

RigidPolygon(<Polygon>)

Creates a copy of any polygon that can only be translated by dragging its first vertex and rotated by dragging its second vertex.

RigidPolygon(<Polygon>, <Offset x>, <Offset y>)

Creates a copy of any polygon with the given offset that can only be translated by dragging its first vertex and rotated by dragging its second vertex.

RigidPolygon(<Free Point>, ..., <Free Point>)

Creates polygon whose shape cannot be changed. This polygon can be translated by dragging its first vertex and rotated by dragging its second vertex.

Note: The copy will join in every change of the original polygon.

If you want to change the shape of the copy, you have to change the original.

Sector Command

Sector(<Conic>, <Point>, <Point>)

Yields a conic sector between two points on the conic section and calculates its area.

Examples:

- Let $c: x^2 + 2y^2 = 8$ be an ellipse, $D = (-2.83, 0)$ and $E = (0, -2)$ two points on the ellipse. $\text{Sector}(c, D, E)$ yields $d = 4.44$.
- Let $c: x^2 + y^2 = 9$ be a circle, $A = (3, 0)$ and $B = (0, 3)$ two points on the circle. $\text{Sector}(c, A, B)$ yields $d = 7.07$

Note: This works only for a circle or ellipse.

Sector(<Conic>, <Parameter Value>, <Parameter Value>)

Yields a conic sector between two parameter values between 0 and 2π on the conic section and calculates its area.

Example: Let $c: x^2 + y^2 = 9$ be a circle. $\text{Sector}(c, 0, 3/4 \pi)$ yields $d = 10.6$

Note: Internally the following parametric forms are used: Circle: $(r \cos(t), r \sin(t))$ where r is the circle's radius. Ellipse: $(a \cos(t), b \sin(t))$ where a and b are the lengths of the semimajor and semiminor axes.

Segment Command

Segment(<Point>, <Point>)

Creates a segment between two points.

Segment(<Point>, <Length>)

Creates a segment with the given length starting at the point.

Notes: When computing intersections with other objects, only intersections lying on the segment are considered. To change this, you can use `Intersect Tool#Outlying_Intersections` option. See also `Segment_ToolSegment` and `Segment_With_Given_Length_ToolSegment_With_Given_Length` tools.

Slope Command

Slope(<Line>)

Returns the slope of the given line.

Note: This command also draws the slope triangle whose size may be changed on tab Style of the Properties Dialog.

Note: See also Slope tool.

Tangent Command

Tangent(<Point>, <Conic>)

Creates (all) tangents through the point to the conic section.

Example: `Tangent((5, 4), 4x^2 - 5y^2 = 20)` yields $x - y = 1$.

Tangent(<Point>, <Function>)

Creates the tangent to the function at $x = x(A)$.

Note: $x(A)$ is the x -coordinate of the given point A .

Example: `Tangent((1, 0), x^2)` yields $y = 2x - 1$.

Tangent(<Point on Curve>, <Curve>)

Creates the tangent to the curve in the given point.

Example: `Tangent((0, 1), Curve(cos(t), sin(t), t, 0, π))` yields $y = 1$.

Tangent(<x-Value>, <Function>)

Creates the tangent to the function at x -Value.

Example: `Tangent(1, x^2)` yields $y = 2x - 1$.

Tangent(<Line>, <Conic>)

Creates (all) tangents to the conic section that are parallel to the given line.

Example: `Tangent(y = 4, x^2 + y^2 = 4)` yields $y = 2$ and $y = -2$.

Tangent(<Circle>, <Circle>)

Creates the common tangents to the two Circles (up to 4).

Example: `Tangent(x^2 + y^2 = 4, (x - 6)^2 + y^2 = 4)` yields $y = 2, y = -2, 1.49x + 1.67y = 4.47$ and $-1.49x + 1.67y = -4.47$.

Tangent(<Point>, <Spline>)

Creates the tangent to the spline in the given point.

Example: Let $A = (0, 1)$, $B = (4, 4)$ and $C = (0, 4)$. `Tangent(A, Spline({A, B, C}))` yields line a : $y = 0.59x + 1$.

Tangent(<Point>, <Implicit Curve>)

Creates the tangent to the implicit curve in the given point.

Example: `Tangent((1, 1), x^2+y^2=1)` yields lines $x=1$ and $y=1$.

Note: See also Tangents tool.

Vertex Command

Vertex(<Conic>)

Returns (all) vertices of the conic section.

Vertex(<Inequality>)

Returns the points of intersection of the borders.

Examples: `Vertex((x + y < 3) && (x - y > 1))` returns point $A = (2, 1)$.
`{Vertex((x + y < 3) \wedge (x - y > 1) \&\& (y > -2))}` returns list1 = {(2, 1), (5, -2), (-1, -2)}.
`Vertex((y > x^2) \wedge (y < x))` returns two points $A = (0, 0)$ and $B = (1, 1)$.
`{Vertex((y > x^2) \wedge (y < x))}` returns list1 = {(0, 0), (1, 1)}.

Vertex(<Polygon>)

Returns (all) vertices of the polygon.

Vertex(<Polygon>, <Index n>)

Returns n -th vertex of the polygon.

Note: To get the vertices of the objects polygon / conic / inequality in a list, use `{Vertex(Object)}`.

Vertex(<Segment>, <Index>)

Returns the start-point (Index = 1) or end-point (Index = 2) of the Segment.

TriangleCurve Command

TriangleCurve(<Point P>, <Point Q>, <Point R>, <Equation in A, B, C>)

Creates implicit polynomial, whose equation in barycentric coordinates with respect to points P, Q, R is given by the fourth parameter; the barycentric coordinates are referred to as A, B, C .

Example: If P, Q, R are points, `TriangleCurve(P, Q, R, (A - B) * (B - C) * (C - A) = 0)` gives a cubic curve consisting of the medians of the triangle PQR .

Example: `TriangleCurve(A, B, C, A*C = 1/8)` creates a hyperbola such that tangent, through A or C , to this hyperbola splits triangle ABC in two parts of equal area.

Example: `TriangleCurve(A, B, C, A^2 + B^2 + C^2 - 2B C - 2C A - 2A B = 0)` creates the Steiner inellipse of the triangle ABC , and `TriangleCurve(A, B, C, B C + C A + A B = 0)` creates the Steiner circumellipse of the triangle ABC .

Note: The input points can be called A, B or C , but in this case you cannot use e.g. $x(A)$ in the equation, because A is interpreted as the barycentric coordinate.

TriangleCenter Command

`TriangleCenter(<Point>, <Point>, <Point>, <Number>)`

gives n -th triangle center of triangle ABC . Works for $n < 3054$.

Example: Let $A = (1, -2)$, $B = (6, 1)$ and $C = (4, 3)$. `TriangleCenter(A, B, C, 2)` yields the centroid $D = (3.67, 0.67)$ of the triangle ABC .

Some common triangle centers

Index n	Center
1	Incenter
2	Centroid
3	Circumcenter
4	Orthocenter
5	Nine-point center
6	Symmedian point
7	Gergonne point
8	Nagel point
13	First isogonic center

Trilinear Command

`Trilinear(<Point>, <Point>, <Point>, <Number>, <Number>, <Number>)`

creates a point whose trilinear coordinates are the given numbers with respect to triangle with given points.

Some examples

Point	<Number>	<Number>	<Number>
A	1	0	0
B	0	1	0
C	0	0	1
Circumcenter	cos()	cos()	cos()
Center of Incircle	1	1	1
Center of excircle tangent to [BC]	-1	1	1
Center of excircle tangent to [AC]	1	-1	1
Center of excircle tangent to [AB]	1	1	-1
Centroid			
Orthocenter	cos() cos()	cos() cos()	cos()cos()

Algebra Commands

<links/>

Div Command

Div(<Dividend Number>, <Divisor Number>)

Returns the quotient (integer part of the result) of the two numbers.

Example:

Div(16, 3) yields 5.

Div(<Dividend Polynomial>, <Divisor Polynomial>)

Returns the quotient of the two polynomials.

Example:

Div(x^2 + 3 x + 1, x - 1) yields $f(x) = x + 4$.

Expand Command

Expand(<Expression>)

Expands the expression.

Example: Expand((2 x - 1)^2 + 2 x + 3) yields .

Note: This command needs to load the Computer Algebra System, so can be slow on some computers. Try using the Polynomial Command instead.

CAS Syntax

Expand(<Expression>)

Expands the expression.

Example: Expand((2 x - 1)^2 + 2 x + 3) yields .

Factor Command

Factor(<Polynomial>)

Factors the polynomial.

Example:

Factor($x^2 + x - 6$) yields $(x - 2)(x + 3)$.

Note: This command needs to load the Computer Algebra System, so can be slow on some computers.

Hint: In the CAS View you can also use the following syntax:

Factor(<Number>) Expresses a number in its prime factorization Example:Factor(360) yields $2^3 3^2 5$.

Factor(<Expression>, <Variable>)

Factors an expression with respect to a given variable.

Example:

- Factor($x^2 - y^2$, x) yields $(x - y)(x + y)$, the factorization of $x^2 - y^2$ with respect to x ,
- Factor($x^2 - y^2$, y) yields $-(y - x)(y + x)$, the factorization of $x^2 - y^2$ with respect to y .

Note: This command factors expressions over the Rational Numbers. To factor over irrational real numbers, see the Ifactor Command. To factor over complex numbers, see the CFactor Command and CIFactor Command.

FromBase Command

FromBase("<Number as Text>", <Base>)

Converts given number from given base into decimal base. The base must be between 2 and 36. The number must be an integer.

Example:

- FromBase("FF", 16) returns 255.
- FromBase("100000000", 2) returns 256.

Note: See also ToBase command

IFactor Command

IFactor(<Polynomial>)

Factors over the irrationals.

Example:

`IFactor(x^2 + x - 1)` gives

CAS Syntax

IFactor(<Expression>)

Factors over the irrationals.

Example:

`IFactor(x^2 + x - 1)` returns

IFactor(<Expression>, <Variable>)

Factors over the irrationals with respect to a given variable.

Example:

`IFactor(a^2 + a - 1, a)` returns

Note: See also CIFactor command.

GCD Command

GCD(<Number>, <Number>)

Calculates the greatest common divisor of the two numbers .

Example:

`GCD(12, 15)` yields 3.

GCD(<List of Numbers>)

Calculates the greatest common divisor of the list of numbers.

Example:

`GCD({12, 30, 18})` yields 6.

Hint: In the CAS View you can also use the following syntax:

GCD(<Polynomial>, <Polynomial>)

Calculates the greatest common divisor of the two polynomials.

Example:

`GCD(x^2 + 4 x + 4, x^2 - x - 6)` yields $x + 2$.

GCD(<List of Polynomials>)

Calculates the greatest common divisor of the list of polynomials.

Example:

`GCD({x^2 + 4 x + 4, x^2 - x - 6, x^3 - 4 x^2 - 3 x + 18})` yields $x + 2$.

Note: See also LCM Command.

LCM Command

UK English: LCM = lowest common multiple

LCM(<Number>, <Number>)

Calculates the least common multiple of two numbers.

Example:

`LCM(12, 15)` yields 60.

LCM(<List of Numbers>)

Calculates the least common multiple of the elements in the list.

Example:

`LCM({12, 30, 18})` yields 180.

Hint: In the CAS View you can also use the following syntax:

LCM(<Polynomial>, <Polynomial>)

Calculates the least common multiple of the two polynomials.

Example:

`LCM(x^2 + 4 x + 4, x^2 - x - 6)` yields .

LCM(<List of Polynomials>)

Calculates the least common multiple of the polynomials in the list.

Example:

`LCM({x^2 + 4 x + 4, x^2 - x - 6, x^3 - 4 x^2 - 3 x + 18})` yields .

Note: See also GCD Command.

Max Command

Max(<List>)

Returns the maximum of the numbers within the list.

Example: Max({ -2, 12, -23, 17, 15 }) yields 17.

Note: If the input consists of non-numeric objects, then this command considers the numbers associated with those objects. If you have a list of segments for example, the command *Max(<List>)* will yield the maximum segment length.

Max(<Interval>)

Returns the upper bound of the interval.

Example: Max(2 < x < 3) yields 3.

Note: Open and closed intervals are treated the same.

Max(<Number>, <Number>)

Returns the maximum of the two given numbers.

Example: Max(12, 15) yields 15.

Max(<Function>, <Start x-Value>, <End x-Value>)

Calculates (numerically) the **local** maximum point of the function in the given interval. The function should be continuous and have only one *local* maximum point in the interval.

Note: For polynomials you should use the Extremum Command.

Example: Max(exp(x) x^2, -3, -1) creates the point (-2, 0.54134).

Max(<List of Data>, <List of Frequencies>)

Returns the maximum of the list of data with corresponding frequencies.

Example: Max({ 1, 2, 3, 4, 5 }, { 5, 3, 4, 2, 0 }) yields 4, the highest number of the list whose frequency is greater than 0.

Note:

- If you want the maximum of two functions $f(x)$ and $g(x)$ then you can define $(f(x) + g(x) + \text{abs}(f(x) - g(x))) / 2$
- See also Extremum Command, Min Command and Function Inspector Tool.

Min Command

Min(<List>)

Returns the minimum of the numbers within the list.

Example: `Min({-2, 12, -23, 17, 15})` yields `-23`.

Note: If the input consists of non-numeric objects, then this command considers the numbers associated with those objects. If you have a list of segments for example, the command `Min(<List>)` will yield the minimum segment length.

Min(<Interval>)

Returns the lower bound of the interval.

Example: `Min(2 < x < 3)` yields `2`.

Note: Open and closed intervals are not distinguished.

Min(<Number>, <Number>)

Returns the minimum of the two given numbers.

Example: `Min(12, 15)` yields `12`.

Min(<Function>, <Start x-Value>, <End x-Value>)

Calculates (numerically) the **local** minimum point for function in the given interval. Function should be continuous and have only one *local* minimum point in the interval.

Note: For polynomials you should use the Extremum Command.

Example: `Min(exp(x) x^3, -4, -2)` creates the point `(-3, -1.34425)`.

Min(<List of Data>, <List of Frequencies>)

Returns the minimum of the list of data with corresponding frequencies.

Example: `Min({1, 2, 3, 4, 5}, {0, 3, 4, 2, 3})` yields `2`, the lowest number of the first list whose frequency is greater than `0`.

Note:

- If you want the minimum of two functions $f(x)$ and $g(x)$ then you can define $(f(x) + g(x)) - \text{abs}(f(x) - g(x))/2$
- See also Max Command, Extremum Command and Function Inspector Tool.

Mod Command

Mod(<Dividend Number>, <Divisor Number>)

Yields the remainder when dividend number is divided by divisor number.

Example: `Mod(9, 4)` yields 1.

Mod(<Dividend Polynomial>, <Divisor Polynomial>)

Yields the remainder when the dividend polynomial is divided by the divisor polynomial.

Example: `Mod(x^3 + x^2 + x + 6, x^2 - 3)` yields $4x + 9$.

Note:

If you want a function to do this, you can define it yourself, e.g. `mod(x, y) = y (x / y - floor(x / y))`.

PrimeFactors Command

PrimeFactors(<Number>)

Returns the list of primes whose product is equal to the given number.

Examples:

- `PrimeFactors(1024)` yields {2, 2, 2, 2, 2, 2, 2, 2, 2}.
- `PrimeFactors(42)` yields {2, 3, 7}.

CAS Syntax

PrimeFactors(<Number>)

Returns the list of primes whose product is equal to the given number.

Examples:

- `PrimeFactors(1024)` yields {2, 2, 2, 2, 2, 2, 2, 2, 2}.
- `PrimeFactors(42)` yields {2, 3, 7}.

Note: See also Factor command.

Simplify Command

Simplify(<Function>)

Simplifies the terms of the given function, if possible.

Example: Simplify ($x + x + x$) yields the function $f(x) = 3x$.

Simplify(<Text>)

Attempts to tidy up text expressions by removing repeated negatives etc.

Example: For $a = b = c = -1$ Simplify (" $f(x) = " + a + "x^2 + " + b + "x + " + c") yields the text $f(x) = -x^2 - x - 1$.$

Note:

The FormulaText Command normally produces better results and is simpler.

Note:

This command needs to load the Computer Algebra System, so can be slow on some computers. Try using the Polynomial Command instead.

CAS Syntax

Simplify(<Function>)

Simplifies the terms of the given function, if possible. Undefined variables can be included in the terms.

Examples:

- Simplify($3 * x + 4 * x + a * x$) yields $a x + 7x$.
- Assume($x < 2$, Simplify(sqrt($x - 2\sqrt{x-1}$))) yields $-\sqrt{abs(x - 1)} + 1$
- Assume($x > 2$, Simplify(sqrt($x - 2\sqrt{x-1}$))) yields $\sqrt{x - 1} + 1$

Note: See also Factor Command, Assume Command, PartialFractions Command, Expand Command, Polynomial Command.

ToBase Command

ToBase(<Number>, <Base>)

Converts given number into different base. The base must be between 2 and 36. The number must be an integer.

Examples:

- ToBase(255, 16) returns "FF".
- ToBase(256, 2) returns "100000000".

Note: See also FromBase command.

Text Commands

- ContinuedFraction
- FormulaText
- FractionText
- LetterToUnicode
- Ordinal
- ReplaceAll
- RotateText
- Split
- StandardForm
- SurdText
- TableText
- Text
- TextToUnicode
- UnicodeToLetter
- UnicodeToText
- VerticalText

See also Insert Text Tool.

ContinuedFraction Command

`ContinuedFraction(<Number>)`

Creates continued fraction of given number. The result is a LaTeX text object. The fraction is computed numerically within precision 10^{-8} .

Example:

`ContinuedFraction(5.45)` gives

`ContinuedFraction(<Number>, <Level>)`

Creates continued fraction of given number. Number of quotients is less than or equal to *Level*, but never exceeds the number of quotients needed to achieve the precision mentioned above.

Example:

`ContinuedFraction(5.45, 3)` gives

`ContinuedFraction(<Number>, <Level>, <Shorthand truelfalse>)`

Meaning of first two arguments is same as above, the argument *Level* is optional. When *Shorthand* is true, shorter syntax for the result is used: the LaTeX text contains a list of the integer parts of the continued fraction.

Examples:

- `ContinuedFraction(5.45, true)` gives [5; 2, 4, 1, 1]
- `ContinuedFraction(5.45, 3, true)` gives [5; 2, 4, ...]

FractionText Command

`FractionText(<Number>)`

Converts the number to a fraction, which is displayed as a (LaTeX) text object in the Graphics View.

Example: If $a: y = 1.5 x + 2$ is a line, then `FractionText(Slope(a))` gives you the fraction $3/2$ as a text.

`FractionText(<Point>)`

Displays the coordinates of the point as fractions in the Graphics View.

Example: If $A=(1.33, 0.8)$ is a point, then `FractionText(A)` gives you the coordinates as a text.

Note: See also SurdText command.

FormulaText Command

FormulaText(<Object>)

Returns the formula for the object as a LaTeX text.

Example:

Let $a = 2$ and $f(x) = a x^2$. FormulaText(f) returns $2 x^2$ (as a LaTeX text).

Note: By default, values are substituted for variables.

FormulaText(<Object>, <Boolean for Substitution of Variables>)

Returns the formula for the object as LaTeX text. The Boolean variable determines if values are substituted for variables (*true*) or if variable names are shown in the text (*false*).

Example:

Let $a = 2$ and $f(x) = a x^2$.

FormulaText(f , *true*) returns $2 x^2$ (as a LaTeX text).

FormulaText(f , *false*) returns $a x^2$ (as a LaTeX text).

FormulaText(<Object>, <Boolean for Substitution of Variables>, <Boolean Show Name>)

Returns the formula for the object as LaTeX text. The first Boolean variable determines if values are substituted for variables (*true*) or if variable names are shown in the text (*false*), the second Boolean variable determines if the object name is shown in the text (*true*) or not (*false*).

Example:

Let $a = 2$ and $f(x) = a x^2$.

FormulaText(f , *true*, *true*) returns $f(x) = 2 x^2$ (as a LaTeX text).

FormulaText(f , *false*, *false*) returns $a x^2$ (as a LaTeX text).

LetterToUnicode Command

`LetterToUnicode("<Letter>")`

Turns a single letter into its ([w:Unicode|Unicode number]).

Example: `LetterToUnicode ("a")` returns the number 97.

Note: The letter needs to be in between a set of quotation marks.

Note: See also UnicodeToLetter Command and TextToUnicode Command.

Ordinal Command

`Ordinal(<Integer>)`

Turns a number into an ordinal (as a text).

Example: `Ordinal (5)` returns "5th".

RotateText Command

`RotateText(<Text>, <Angle>)`

Returns text rotated by given angle. LaTeX is used for rendering of the result.

Example: `RotateText ("a = 5", 45°)`

Notes: The text needs to be enclosed in double quotes ". The text is rotated around the top left corner (also known as corner 4) of the box containing the text, and placed at the origin of the coordinate system.

Example: If you want to place the text "GeoGebra", rotated by 42°, at point (6,6), use the command
`Text(RotateText("GeoGebra", 42°), (6, 6), true, true)`

- The angle is in radians unless you explicitly use the degree symbol °.

ScientificText Command

ScientificText(<Number>)

Displays the number in scientific notation

Example: ScientificText (0.002) gives 2×10^{-3} .

ScientificText(<Number>, <Precision>)

Displays the number in scientific notation, rounded to the number of significant digits specified by **precision**

Example: ScientificText (e, 5) gives 2.7183×10^0 .

SurdText Command

SurdText(<Number>)

Creates text representation of the number in the form .

Examples:

- SurdText (2.414213562373095) creates
- SurdText (2.439230484541326) creates

SurdText(<Number>, <List>)

Creates text representation of the number as multiples of the constants in the list. If the list is empty it uses a list of common constants.

Examples:

- SurdText (3.718281828459045, {exp(1)}) creates
- SurdText (5.382332347441762, {sqrt(2), sqrt(3), sqrt(5)}) creates
- SurdText (1.693147180559945, {ln(2)}) creates

SurdText(<Point>)

Creates text representation of the point, where coordinates are in the form .

Example:

SurdText ((2.414213562373095, 1.414213562373095)) creates (

Notes:

- In order to use this command in a Text object, the option **LaTeX Formula** needs to be enabled in the *Text* tab of the Properties Dialog of the text.
- Since this command works with a rounded decimal number as input, sometimes the result will be unexpected.
- If a suitable answer can't be found, the number will be returned.

Example: SurdText (1.23456789012345) returns 1.23456789012345.

- See also the FractionText and ScientificText commands.

TableText Command

TableText(<List>, <List>, ...)

Creates a text that contains a table of the list objects.

Note: By default, each list is displayed in its own row of the table.

Examples: `TableText({x^2, 4}, {x^3, 8}, {x^4, 16})` creates a table as a text object with three rows and two columns. All items of the table are left aligned. `TableText(Sequence(i^2, i, 1, 10))` creates a table as a text object with one row. All items of the table are left aligned.

TableText(<List>, <List>, ..., <Alignment of Text>)

Creates a text that contains a table of the list objects. The optional text “Alignment of text” controls the orientation and alignment of the table text, as well as the alignment of the separator in decimal values.

Note: Possible values are "vl", "vc", "vr", "v", "h", "hl", "hc", "hr" and "." or "%" - the default value is "hl". "v" = vertical, i. e. lists are columns "h" = horizontal, i. e. lists are rows "l" = left aligned "r" = right aligned "c" = centered "." = aligned on decimal points "a" = like "." but also displays the padding zeros "%" = converted to a percentage, and aligned on decimal points "p" = like "%" but also displays the padding zeros

Examples: `TableText({1, 2, 3, 4}, {1, 4, 9, 16}, "v")` creates a text with two columns and four rows whose elements are left aligned. `TableText({1, 2, 3, 4}, {1, 4, 9, 16}, "h")` creates a text with two rows and four columns whose elements are left aligned. `TableText({11.2, 123.1, 32423.9, "234.0"}, "vr")` creates a text with one column whose elements are right aligned. `TableText({A1:A10, B1:B10, C1:C10}, "vl")` creates a text with three columns whose elements (left aligned) are the objects in the given Spreadsheet cells. `TableText({{2011.56, 2, 3.7, 4}, {1, 4.2, 9, 16.365}}, "v.")` creates a text whose elements are aligned on decimal points. `TableText({{2011.56, 2, 3.7, 4}, {1, 4.2, 9, 16.365}}, "v%")` creates a text whose elements are converted to a percentage, and aligned on decimal points

Online examples by Mike ^[1]

It's also possible to insert:

- different types of brackets, using the following symbols |||, ||, { }, [] or ()
- line separators, using the symbol _
- column separators, using the symbol |
- different colourings

Examples: `TableText({1, 2}, {3, 4}, "c())")` creates the text `\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}`. `TableText({1, 2}, {3, 4}, "cl_")` creates the text `\begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix}`. `TableText({1, 2}, {3, 4}, "|||")` creates the text `\begin{Vmatrix} 1 & 2 \\ 3 & 4 \end{Vmatrix}`. `TableText({{2x+3y=5, "5x+8y=12"}}, "v")` creates the text `\left(\begin{matrix} 2x+3y=5 \\ 5x+8y=12 \end{matrix}\right)`. `TableText({{1, 2, 3, 4}, {1, 2, 3, 4}, "-/_v"})` creates a table with border and no separation lines. `TableText({{1, 2, 3, 4}, {1, 2, 3, 4}, {1, 2, 3, 4}, {1, 2, 3, 4}, "11001 _110001 h"})` creates a table with border and one separation line to the right of the first column and under the first row of contents. The value 1 in the syntax means that there is a separation line between the numbers and the value 0 means that there is no separation line or border. `TableText({{\color{blue}{0, 1, 2, 3, 4}}, {\color{red}{4, 3, 2, 1, 0}}}, "v")` creates a table having the objects in the first row coloured in blue, the ones in the second row coloured in red.

Note: The Style Bar of a *TableText* object allows the user to customize the object's appearance, background and text colour and text style.

References

[1] <https://www.geogebra.org/m/Eq5T3vV3>

Text Command

Text(<Object>)

Returns the formula of the object as a text object.

Note: By default, values are substituted to variables.

Example: If $a = 2$ and $c = a^2$, then `Text(c)` returns the text "4".

Text(<Object>, <Boolean for Substitution of Variables>)

Returns the formula of the object as a text object. The Boolean variable determines whether values are substituted to variables (*true*) or variable names are shown in the text (*false*).

Example:

If $a = 2$ and $c = a^2$, then

- `Text(c, true)` returns the text "4" and
- `Text(c, false)` returns the text " a^2 ".

Text(<Object>, <Point>)

Returns the formula of the object as a text object at the position of the given point.

Example: `Text("hello", (2, 3))` returns the text "hello" at the position (2, 3).

Text(<Object>, <Point>, <Boolean for Substitution of Variables>)

Returns the formula of the object as a text object, at the position of the given point. The Boolean variable determines whether values are substituted to variables (*true*) or variable names are shown in the text (*false*).

Example: If $a = 2$ and $c = a^2$, then `Text(c, (2, 1), true)` returns the text "4" at the position (2, 1).

Text(<Object>, <Point>, <Boolean for Substitution of Variables>, <Boolean for LaTeX formula>)

Returns the formula of the object as a text object at the position of the given point. First Boolean variable determines whether values are substituted to variables (*true*) or variable names are shown in the text (*false*). If the second Boolean variable is *true*, the text is rendered using LaTeX.

Example: If $a = 2$ and $c = a^2$, then `Text(c, (2, 1), true, true)` returns the LaTeX text "4" at the position (2, 1).

Text(<Object>, <Point>, <Boolean for Substitution of Variables>, <Boolean for LaTeX formula>, <Horizontal alignment [-1|0|1]>, <Vertical alignment [-1|0|1]>)

Returns the formula of the object as a text object at the position of the given point. The first Boolean variable determines whether values are substituted to variables (*true*) or variable names are shown in the text (*false*). If the second Boolean variable is *true*, the text is rendered using LaTeX. The values -1, 0, 1 for horizontal and vertical alignment shift the text object from the default position respectively as follows:

- -1: horizontal shift leftwards / vertical shift downwards
- 0: centers the text object horizontally / vertically at the given point
- 1: horizontal shift rightwards / vertical shift upwards

Example: If $a = 2$ and $c = a^2$, then `Text(c, (2, 1), true, true, -1, 0)` returns the LaTeX text "4" to the left of (2, 1), and vertically aligned with the point.

Note: See also Text tool.

TextToUnicode Command

TextToUnicode("<Text>")

Turns the text into a list of Unicode numbers, one for each character.

Examples: TextToUnicode("Some text") gives you the list of Unicode numbers {83, 111, 109, 101, 32, 116, 101, 120, 116}. If text1 is "hello", then TextToUnicode(text1) gives you the list of Unicode numbers {104, 101, 108, 108, 111}.

Note: See also UnicodeToText Command and LetterToUnicode Command.

UnicodeToLetter Command

UnicodeToLetter(<Integer>)

Converts the integer Unicode number back into a letter which is displayed as a text object in the Graphics View.

Example: UnicodeToLetter(97) yields the text "a".

Note: See also LetterToUnicode Command and UnicodeToText Command.

UnicodeToText Command

UnicodeToText(<List of Integers>)

Converts the integer Unicode numbers back into text.

Example: UnicodeToText ({104, 101, 108, 108, 111}) yields the text "hello".

Note: See also TextToUnicode Command and UnicodeToLetter Command.

VerticalText Command

`VerticalText(<Text>)`

Returns text rotated by 90° counter-clockwise. LaTeX is used for rendering of the result.

`VerticalText(<Text>, <Point>)`

This allows the location of the Text to be specified.

Logic Commands

<links/>

CountIf Command

`CountIf(<Condition>, <List>)`

Counts the number of elements in the list satisfying the condition.

Example: `CountIf(x < 3, {1, 2, 3, 4, 5})` gives you the number 2. `CountIf(x < 3, A1:A10)`, where A1:A10 is a range of cells in the spreadsheet, counts all cells whose values are less than 3.

Note: For list of numbers arbitrary condition may be used. For list of other objects one can use only conditions of the form `x==constant` or `x!=constant`.

`CountIf(<Condition>, <Variable>, <List>)`

As above, using a more flexible syntax.

Example:

Given points P, Q, R `CountIf(x(A) < 3, A, {P, Q, R})` will count only the points whose x-coordinate is less than 3. The variable A is replaced in turn with P then Q then R for the check. Therefore `CountIf(x(A) < 3, A, {(0, 1), (4, 2), (2, 2)})` yields the number 2.

IsDefined Command

IsDefined(<Object>)

Returns *true* or *false* depending on whether the object is defined or not.

Example: `IsDefined(Circle((1,1), -2))` returns *false*.

If Command

If(<Condition>, <Then>)

Yields a copy of the object *Then* if the condition evaluates to *true*, and an undefined object if it evaluates to *false*.

Examples:

- Let $n = 3$. `If(n==3, x + y = 4)` yields line $x + y = 4$, because the condition on number n is met.
- Let $n = 4$. `If(n==3, x + y = 4)` creates an *undefined* object, because the condition on number n is not met .

If(<Condition>, <Then>, <Else>)

Yields a copy of object *Then* if the condition evaluates to *true*, and a copy of object *Else* if it evaluates to *false*.

Both objects must be of the same type.

Example: Let n be a number. `If(n==3, x + y = 4, x - y = 4)` yields line $x + y = 4$ when $n = 3$, and line $x - y = 4$ for all n not equal to 3.

If(<Condition 1>, <Then 1>, <Condition 2>, <Then 2>, ... , <Else (optional)>)

Yields a copy of "Then 1" when first condition is satisfied, "Then 2" if second condition is satisfied etc. If none of the conditions are satisfied and Else is given, this command yields a copy of Else. Otherwise undefined is returned.

Example: `If(a == 1, "Matthew", a == 2, "Larry", a == 3, "Vivian", "Alex")` When $a=1$ this returns the text "Matthew", for $a=2$ it returns "Larry", for $a=3$ "Vivian" and for all other values of a it yields "Alex".

Note: This syntax isn't supported in the CAS View

Conditional Functions

The **If** command can be used to create conditional functions. Such conditional functions may be used as arguments in any command that takes a function argument, such as Derivative, Integral, and Intersect.

Examples:

- `f(x) = If(x < 3, sin(x), x^2)` yields a piecewise function that equals $\sin(x)$ for $x < 3$ and x^2 for $x \geq 3$.
- `f(x) = If(0 <= x <= 3, sin(x))` yields a function that equals $\sin(x)$ for x between 0 and 3 (and undefined otherwise).

Note: A shorter syntax for this is `f(x) = sin(x), 0 <= x <= 3`

- `f(x) = If(x<-1, x^2, -1<=x<=1, 1, -x^2+2)` yields the piecewise function .

Notes:

- Derivative of $If(condition, f(x), g(x))$ gives $If(condition, f'(x), g'(x))$. It does not do any evaluation of limits at the critical points.

- See section: Boolean values for the symbols used in conditional statements.

If Command in Scripting

If command can be used in scripts to perform different actions under certain conditions.

Example: Let n be a number, and A a point. The command `If (Mod(n, 7) == 0, SetCoords(A, n, 0), SetCoords(A, n, 1))` modifies the coordinates of point A according to the given condition. In this case it would be easier to use `SetCoords(A, n, If (Mod(n, 7) == 0, 0, 1))`.

Note: Arguments of **If** must be Objects or Scripting Commands, not assignments. Syntax `b = If(a > 1, 2, 3)` is correct, but `b = 2` or `b = 3` would not be accepted as parameters of **If**.

IsInRegion Command

`IsInRegion(<Point>, <Region>)`

Returns *true* if the point is in given region and *false* otherwise.

Example: `IsInRegion((1, 2), Polygon((0, 0), (2, 0), (1, 3)))` returns *true*.

IsInteger Command

`IsInteger(<Number>)`

Returns *true* or *false* depending whether the number is an integer or not.

Example: `IsInteger(972 / 9)` returns *true*.

KeepIf Command

KeepIf(<Condition>, <List>)

Creates a new list that only contains those elements of the initial list that fulfil the condition.

Example: `KeepIf(x<3, {1, 2, 3, 4, 1, 5, 6})` returns the new list `{1, 2, 1}`.

Note: For list of numbers arbitrary condition may be used. For list of other objects one can use only conditions of the form `x==constant` or `x!=constant`.

KeepIf(<Condition>, <Variable>, <List>)

This syntax allows a more flexible condition.

Example: For Points P, Q, R `KeepIf(x(A) < 3, A, {P, Q, R})` will filter the points whose x -coordinate is less than 3 out of the list. The variable A is replaced in turn with P then Q then R for the check.

Relation Command

Relation(<List>)

Shows a message box that gives you information about the relation between two or more (up to 4) objects.

Relation(<Object>, <Object>)

Shows a message box that gives you information about the relation between two objects.

This command allows you to find out whether

- two lines are perpendicular
- two lines are parallel
- two (or more) objects are equal
- a point lies on a line or conic
- a line is tangent or a passing line to a conic
- three points are collinear
- three lines are concurrent (or parallel)
- four points are concyclic (or collinear).

Some of these checks can also be performed symbolically. If GeoGebra supports symbolic check for a certain property, the "More" button appears. By clicking it, GeoGebra may provide more information whether the property is true in general (eventually under certain conditions).

Note: See also Relation tool.

Financial Commands

<links/>

FutureValue Command

FutureValue(<Rate>, <Number of Periods>, <Payment>, <Present Value (optional)>, <Type (optional)>)

Returns the future value of an investment based on periodic, constant payments and a constant interest rate.

- **<Rate>** Interest rate per period.
- **<Number of Periods>** Total number of payment periods in an annuity.
- **<Payment>** The amount paid in each period.
- **<Present Value (optional)>** Total amount that a series of future payments is worth now. If you do not enter a value, it is assumed to be 0.
- **<Type (optional)>** Indicates when payments are due. If you do not enter a value or you enter 0 the payment is due at the end of the period. If you enter 1 it is due at the beginning of the period.

Example:

FutureValue(10%/12, 15, -200, 0, 1) yields a future value of 3207.99.

Note: Make sure that you are consistent about the units you use for <Rate> and <Number of Periods>. If you make monthly payments on a four-year loan at an annual interest rate of 10 percent, use 10%/12 for rate and 4*12 for number of payments.

Note: For all arguments, cash paid out is represented by negative numbers and cash received by positive numbers.

Note: See also Payment, Rate, Present Value and Periods commands.

Payment Command

Payment(<Rate>, <Number of Periods>, <Present Value>, <Future Value (optional)>, <Type (optional)>)

Calculates the payment for a loan based on constant payments and a constant interest rate.

- **<Rate>** Interest rate per period.
- **<Number of Periods>** Total number of payments for the loan.
- **<Present Value>** Total amount that a series of future payments is worth now.
- **<Future Value (optional)>** A cash balance you want to attain after the last payment. If you do not enter a future value, it is assumed to be 0.
- **<Type (optional)>** Indicates when payments are due. If you do not enter a value or you enter 0 the payment is due at the end of the period. If you enter 1 it is due at the beginning of the period.

Example:

`Payment (6%/12, 10, 10000, 0, 1)` yields a monthly payment for a loan of -1022.59.

Note: Make sure that you are consistent about the units you use for **<Rate>** and **<Number of Periods>**. If you make monthly payments on a four-year loan at an annual interest rate of 6 percent, use $6\%/12$ for rate and $4*12$ for number of payments.

Note: For all arguments, cash paid out is represented by negative numbers and cash received by positive numbers.

Note: See also Rate, Periods, Present Value and Future Value commands.

Periods Command

Periods(<Rate>, <Payment>, <Present Value>, <Future Value (optional)>, <Type (optional)>)

Returns the number of periods for an annuity based on periodic, fixed payments and a fixed interest rate.

- **<Rate>** Interest rate per period.
- **<Payment>** The amount paid in each period.
- **<Present Value>** Total amount that a series of future payments is worth now.
- **<Future Value (optional)>** A cash balance you want to attain after the last payment. If you do not enter a future value, it is assumed to be 0.
- **<Type (optional)>** Indicates when payments are due. If you do not enter a value or you enter 0 the payment is due at the end of the period. If you enter 1 it is due at the beginning of the period.

Example:

`Periods (10%/12, -200, -400, 10000)` yields a number of payments of 39.98.

`Periods (10%/12, -200, -400, 10000, 1)` yields a number of payments of 39.7.

Note: If you make monthly payments on an annual interest rate of 10 percent, use $10\%/12$ for **<Rate>**.

Note: For all arguments, cash paid out is represented by negative numbers and cash received by positive numbers.

Note: See also Payment, Rate, Present Value and Future Value commands.

PresentValue Command

`PresentValue(<Rate>, <Number of Periods>, <Payment>, <Future Value (optional)>, <Type (optional)>)`

Returns the total amount of payments of an investment.

- **<Rate>** Interest rate per period.
- **<Number of Periods>** Total number of payments for the loan.
- **<Payment>** The amount paid in each period.
- **<Future Value (optional)>** A cash balance you want to attain after the last payment. If you do not enter a future value, it is assumed to be 0.
- **<Type (optional)>** Indicates when payments are due. If you do not enter a value or you enter 0 the payment is due at the end of the period. If you enter 1 it is due at the beginning of the period.

Example:

`PresentValue(12%/12, 4*12, -100, 5000, 0)` yields a present value of 696.06.

`PresentValue(12%/12, 4*12, -100, 5000, 1)` yields a present value of 734.07.

Note: Make sure that you are consistent about the units you use for **<Rate>** and **<Number of Periods>**. If you make monthly payments on a four-year loan at an annual interest rate of 12 percent, use $12\%/12$ for rate and $4*12$ for number of payments.

Note: For all arguments, cash paid out is represented by negative numbers and cash received by positive numbers.

Note: See also Payment, Periods, Rate and Future Value commands.

Rate Command

`Rate(<Number of Periods>, <Payment>, <Present Value>, <Future Value (optional)>, <Type (optional)>, <Guess (optional)>)`

Returns the interest rate per period of an annuity.

- **<Number of Periods>** Total number of payments for the loan.
- **<Payment>** The amount paid in each period.
- **<Present Value>** Total amount that a series of future payments is worth now.
- **<Future Value (optional)>** A cash balance you want to attain after the last payment. If you do not enter a future value, it is assumed to be 0.
- **<Type (optional)>** Indicates when payments are due. If you do not enter a value or you enter 0 the payment is due at the end of the period. If you enter 1 it is due at the beginning of the period.
- **<Guess (optional)>** Your guess for what the rate will be.

Example:

`Rate(5*12, -300, 10000)` yields a monthly rate of 0.02 (2%).

Note: If you make monthly payments on a five-year loan use $5*12$ for **<Number of Periods>**.

Note: For all arguments, cash paid out is represented by negative numbers and cash received by positive numbers..

Note: See also Payment, Periods, Present Value and Future Value commands.

Functions & Calculus Commands

<links/>

Asymptote Command

Asymptote(<Conic>)

Yields both asymptotes of the conic.

Example: `Asymptote(x^2 - y^2 / 4 = 1)` returns line $-2x + y = 0$ and line $-2x - y = 0$.

Asymptote(<Function>)

GeoGebra will attempt to find the asymptotes of the function and return them in a list. It may not find them all, for example vertical asymptotes of non-rational functions such as $\ln(x)$. **This syntax is not available in the Graphing and Geometry Apps**

Example: `Asymptote((x^3 - 2x^2 - x + 4) / (2x^2 - 2))` returns the list $\{y = 0.5x - 1, x = 1, x = -1\}$.

Asymptote(<Implicit Curve>)

Yields a list containing all the asymptotes of the Implicit Curve.

Example: `Asymptote(x^3 + y^3 + y^2 - 3x = 0)` returns the list $\{x + y = -0.33\}$.

Coefficients Command

Coefficients(<Polynomial>)

Yields the list of all coefficients of the polynomial .

Example:

`Coefficients(x^3 - 3x^2 + 3x)` yields $\{1, -3, 3, 0\}$, the list of all coefficients of .

Note: There's a special mode (for non-polynomials) for the output of the fitting commands eg if `f(x) = FitExp(l1)` then `Coefficients(f)` will return the calculated parameters

Coefficients(<Conic>)

Returns the list of the coefficients a, b, c, d, e, f of a conic in standard form:

Note: For a line in implicit form $l: ax + by + c = 0$ it is possible to obtain the coefficients using the syntax $x(l), y(l), z(l)$. Example: Given line: $3x + 2y - 2 = 0$: $x(l)$ returns 3 $y(l)$ returns 2 $z(l)$ returns -2

CAS Syntax

`Coefficients(<Polynomial>)`

Yields the list of all coefficients of the polynomial in the main variable.

Example:

`Coefficients(x^3 - 3 x^2 + 3 x)` yields $\{1, -3, 3, 0\}$, the list of all coefficients of .

`Coefficients(<Polynomial>, <Variable>)`

Yields the list of all coefficients of the polynomial in the given variable.

Example:

- `Coefficients(a^3 - 3 a^2 + 3 a, a)` yields $\{1, -3, 3, 0\}$, the list of all coefficients of
- `Coefficients(a^3 - 3 a^2 + 3 a, x)` yields $\{a^3 - 3 a^2 + 3 a\}$.

CompleteSquare Command

`CompleteSquare(<Quadratic Function>)`

Returns the quadratic function in the form: .

Example:

`CompleteSquare(x^2 - 4x + 7)` yields $1(x - 2)^2 + 3$.

CAS Syntax

`CompleteSquare(<Quadratic Function>)`

Returns the quadratic function in the form: .

Example:

`CompleteSquare(x^2 - 4x + 7)` yields $(x - 2)^2 + 3$.

ComplexRoot Command

ComplexRoot(<Polynomial>)

Finds the complex roots of a given polynomial in x . Points are created in Graphics View.

Example:

`ComplexRoot (x^2 + 4)` yields $(0 + 2i)$ and $(0 - 2i)$

CAS Syntax

ComplexRoot(<Polynomial>)

Finds the complex roots of a given polynomial in x .

Example:

`ComplexRoot (x^2 + 4)` yields $\{-2i, 2i\}$

Note:

Use CSolve Command instead.

Curvature Command

Curvature(<Point>, <Function>)

Calculates the curvature of the function in the given point.

Example: `Curvature((0, 0), x^2)` yields 2.

Curvature(<Point>, <Curve>)

Calculates the curvature of the curve in the given point.

Example: `Curvature((0, 0), Curve(cos(t), sin(2t), t, 0, π))` yields 0.

Curvature(<Point>, <Object>)

Yields the curvature of the object (function, curve, conic) in the given point.

Examples:

- `Curvature((0, 0), x^2)` yields 2
- `Curvature((0, 0), Curve(cos(t), sin(2t), t, 0, π))` yields 0
- `Curvature((-1, 0), Conic({1, 1, 1, 2, 2, 3}))` yields 2

CurvatureVector Command

CurvatureVector(<Point>, <Function>)

Yields the curvature vector of the function in the given point.

Example: `CurvatureVector((0, 0), x^2)` yields vector $(0, 2)$.

CurvatureVector(<Point>, <Curve>)

Yields the curvature vector of the curve in the given point.

Example: `CurvatureVector((0, 0), Curve(cos(t), sin(2t), t, 0, π))` yields vector $(0, 0)$.

CurvatureVector(<Point>, <Object>)

Yields the curvature vector of the object (function, curve, conic) in the given point.

Examples:

- `CurvatureVector((0, 0), x^2)` yields vector $(0, 2)$
- `CurvatureVector((0, 0), Curve(cos(t), sin(2t), t, 0, π))` yields vector $(0, 0)$
- `CurvatureVector((-1, 0), Conic({1, 1, 1, 2, 2, 3}))` yields vector $(0, -2)$

Curve Command

Curve(<Expression>, <Expression>, <Parameter Variable>, <Start Value>, <End Value>)

Yields the Cartesian parametric curve for the given x -expression (first <Expression>) and y -expression (second <Expression>) (using parameter variable) within the given interval [*Start Value*, *End Value*].

Example: `Curve(2 cos(t), 2 sin(t), t, 0, 2π)` creates a circle with radius 2 around the origin of the coordinate system.

Curve(<Expression>, <Expression>, <Expression>, <Parameter Variable>, <Start Value>, <End Value>)

Yields the 3D Cartesian parametric curve for the given x -expression (first <Expression>), y -expression (second <Expression>) and z -expression (third <Expression>) (using parameter variable) within the given interval [*Start Value*, *End Value*].

Example: `Curve(cos(t), sin(t), t, t, 0, 10π)` creates a 3D spiral.

Notes:

- *End Value* must be greater than or equal to *Start Value* and both must be finite.
- x , y and z are not allowed as parameter variables.
- See Curves for details, also see the Derivative Command and the Parametric Derivative Command.

Degree Command

Degree(<Polynomial>)

Gives the degree of a polynomial (in the main variable).

Example: `Degree(x^4 + 2 x^2)` yields 4

CAS Syntax

Degree(<Polynomial>)

Gives the degree of a polynomial (in the main variable or monomial).

Example:

- `Degree(x^4 + 2 x^2)` yields 4
- `Degree(x^6 y^3 + 2 x^2 y^3)` yields 9

Degree(<Polynomial>, <Variable>)

Gives the degree of a polynomial in the given variable.

Example:

- `Degree(x^4 y^3 + 2 x^2 y^3, x)` yields 4
- `Degree(x^4 y^3 + 2 x^2 y^3, y)` yields 3

Denominator Command

Denominator(<Function>)

Returns the denominator of a function.

Example: `Denominator(5 / (x^2 + 2))` yields $f(x)=(x^2 + 2)$.

Denominator(<Number>)

For a rational number returns its denominator. It uses a numerical method, which limits this command to numbers with small denominator. For irrational input the denominator of its continued fraction is returned.

Note: See also Numerator Command and FractionText Command.

CAS Syntax

Denominator(<Expression>)

Returns the denominator of a rational number or expression.

Example: `Denominator(2 / 3 + 1 / 15)` yields 15.

Derivative Command

Derivative(<Function>)

Returns the derivative of the function with respect to the main variable.

Example: `Derivative(x^3 + x^2 + x)` yields $3x^2 + 2x + 1$.

Derivative(<Function>, <Number>)

Returns the n^{th} derivative of the function with respect to the main variable, whereupon n equals <Number>.

Example: `Derivative(x^3 + x^2 + x, 2)` yields $6x + 2$.

Derivative(<Function>, <Variable>)

Returns the partial derivative of the function with respect to the given variable.

Example: `Derivative(x^3 y^2 + y^2 + xy, y)` yields $2x^3y + x + 2y$.

Derivative(<Function>, <Variable>, <Number>)

Returns the n^{th} partial derivative of the function with respect to the given variable, whereupon n equals <Number>.

Example: `Derivative(x^3 + 3x y, x, 2)` yields $6x$.

Derivative(<Curve>)

Returns the derivative of the curve.

Example: `Derivative(Curve(cos(t), t sin(t), t, 0, π))` yields curve $x = -\sin(t)$, $y = \sin(t) + t \cos(t)$.

Note: This only works for parametric curves.

Derivative(<Curve>, <Number>)

Returns the n^{th} derivative of the curve, whereupon n equals <Number>.

Example: `Derivative(Curve(cos(t), t sin(t), t, 0, π), 2)` yields curve $x = -\cos(t)$, $y = 2\cos(t) - t \sin(t)$.

Note: This only works for parametric curves.

Note: You can use `f'(x)` instead of `Derivative(f)`, or `f''(x)` instead of `Derivative(f, 2)`, and so on.

CAS Syntax

Derivative(<Expression>)

Returns derivative of an expression with respect to the main variable.

Example: `Derivative(x^2)` yields $2x$.

Derivative(<Expression>, <Variable>)

Returns derivative of an expression with respect to the given variable.

Example: `Derivative(a x^3, a)` yields x^3 .

Derivative(<Expression>, <Variable>, <Number>)

Returns the n^{th} derivative of an expression with respect to the given variable, whereupon n equals <Number>.

Examples:

- `Derivative(y x^3, x, 2)` yields $6xy$.

- `Derivative(x^3 + 3x y, x, 2)` yields $6x$.

Extremum Command

Extremum(<Polynomial>)

Yields all local extrema of the polynomial function as points on the function graph.

Example:

`Extremum(x3 + 3x2 - 2x + 1)` creates local extrema (0.29, 0.70) and (-2.29, 9.30) and shows them in the Graphics View.

Extremum(<Function>, <Start x-Value>, <End x-Value>)

Calculates (numerically) the extremum of the function in the open interval (<Start x-Value>, <End x-Value>).

Example:

`Extremum((x4 - 3x3 - 4x2 + 4) / 2, 0, 5)` creates local extremum (2.93, -16.05) in the given interval and shows it in the Graphics View.

Note: The function should be continuous in [<Start x-Value>, <End x-Value>], otherwise false extrema near discontinuity might be calculated.

Factors Command

Factors(<Polynomial>)

Gives a list of lists of the type *{factor, exponent}* such that the product of all these factors raised to the power of the corresponding exponents equals the given polynomial. The factors are sorted by degree in ascending order.

Example: `Factors(x8 - 1)` yields `\{\{x - 1, 1\}, \{x + 1, 1\}, \{x2 + 1, 1\}, \{x4 + 1, 1\}\}`.

Note: Not all of the factors are irreducible over the reals.

Factors(<Number>)

Gives matrix of the type such that the product of all these primes raised to the power of the corresponding *exponents* equals the given number. The primes are sorted in ascending order.

Example:

- `Factors(1024)` yields (2 10), since .
- `Factors(42)` yields , since .

Note: See also PrimeFactors Command and Factor Command.

Note: In the CAS View undefined variables can be used as input and the results are returned as proper matrices. Example: `Factors(a8 - 1)` yields `\left(\begin{array}{c} a - 1 \& 1 \\ a + 1 \& 1 \\ a^2 + 1 \& 1 \\ a^4 + 1 \& 1 \end{array} \right)`.

Function Command

Function(<List of Numbers>)

Yields the following function: The first two numbers determine the start x -value and the end x -value. The rest of the numbers are the y -values of the function in between in equal distances.

Example:

- `Function[{2, 4, 0, 1, 0, 1, 0}]` yields a triangular wave between $x = 2$ and $x = 4$.
- `Function[{-3, 3, 0, 1, 2, 3, 4, 5}]` yields a linear equation with slope = 1 between $x = -3$ and $x = 3$.

Function(<Expression>, <Parameter Variable 1>, <Start Value>, <End Value>, <Parameter Variable 2>, <Start Value>, <End Value>)

Restricts the visualization of the representative surface of a function of two variables in 3D space.

Example:

The expression $a(x, y) = x + 0y$ creates a function of two variables, whose graph in 3D space is the **plane** $z = a(x, y) = x$.

`Function[u, u, 0, 3, v, 0, 2]` creates the function of two variables $b(u, v) = u$, whose graph in 3D space is the **rectangle** `Polygon[(0, 0, 0), (3, 0, 3), (3, 2, 3), (0, 2, 0)]` contained in plane $z = a(x, y) = x$.

ImplicitCurve Command

ImplicitCurve(<List of Points>)

Creates implicit curve through given set of points. The length of the list must be for implicit curve of degree .

ImplicitCurve(<f(x,y)>)

Creates the implicit curve $f(x, y) = 0$.

Integral Command

`Integral(<Function>)`

Gives the indefinite integral with respect to the main variable.

Example: `Integral(x^3)` yields .

`Integral(<Function>, <Variable>)`

Gives the partial integral with respect to the given variable.

Example: `Integral(x^3+3x y, x)` gives $+x^2y$.

`Integral(<Function>, <Start x-Value>, <End x-Value>)`

Gives the definite integral over the interval *[Start x-Value , End x-Value]* with respect to the main variable.

Note: This command also shades the area between the function graph of f and the x -axis.

`Integral(<Function>, <Start x-Value>, <End x-Value>, <Boolean Evaluate>)`

Gives the definite integral of the function over the interval *[Start x-Value , End x-Value]* with respect to the main variable and shades the related area if *Evaluate* is *true*. In case *Evaluate* is *false* the related area is shaded but the integral value is not calculated.

CAS Syntax

In the CAS View undefined variables are allowed as input as well.

Example: `Integral(cos(a t), t)` yields .

Furthermore, the following command is only available in the *CAS View*:

`Integral(<Function>, <Variable>, <Start x-Value>, <End x-Value>)`

Gives the definite integral over the interval *[Start x-Value , End x-Value]* with respect to the given variable.

Example: `Integral(cos(t), t, a, b)` yields .

Note:

- The answer isn't guaranteed to be continuous, eg `Integral(floor(x))`, that is the integral of the function $\lfloor x \rfloor$ - in that case you can define your own function to use eg $F(x) = (\text{floor}(x)^2 - \text{floor}(x))/2 + x \text{floor}(x) - \text{floor}(x)^2$, i.e. the function
- in some versions of GeoGebra, a numerical algorithm is used so integrating up to an asymptote or similar eg `Integral(ln(x), 0, 1)` won't work. In this case try `Integral(ln(x), 0, 1, false)`

IntegralBetween Command

IntegralBetween(<Function>, <Function>, <Number>, <Number>)

Gives the definite integral of the difference $f(x) - g(x)$ of two function f and g over the interval $[a, b]$, where a is the first number and b the second, with respect to the main variable.

Example: IntegralBetween(sin(x), cos(x), 0, pi)

Note: This command also shades the area between the function graphs of f and g .

IntegralBetween(<Function>, <Function>, <Number>, <Number>, <Boolean Evaluate>)

Gives the definite integral of the difference $f(x) - g(x)$ of two function f and g over the interval $[a, b]$, where a is the first number and b the second, with respect to the main variable and shadows the related area if *Evaluate* is *true*. In case *Evaluate* is *false* the related area is shaded but the integral value is not calculated.

CAS Syntax

IntegralBetween(<Function>, <Function>, <Number>, <Number>)

Gives the definite integral of the difference $f(x) - g(x)$ of two function f and g over the interval $[a, b]$, where a is the first number and b the second, with respect to the main variable.

Example: IntegralBetween(sin(x), cos(x), pi / 4, pi * 5 / 4) yields .

IntegralBetween(<Function>, <Function>, <Variable>, <Number>, <Number>)

Gives the definite integral of a variable of the difference $f(x) - g(x)$ of two function f and g over the interval $[a, b]$, where a is the first number and b the second, with respect to the given variable.

Example: IntegralBetween(a * sin(t), a * cos(t), t, pi / 4, pi * 5 / 4) yields .

Iteration Command

`Iteration(<Function>, <Start Value>, <Number of Iterations>)`

Iterates the function n times ($n = \text{number of iterations}$) using the given start value.

Examples:

- After defining $f(x) = x^2$ the command `Iteration(f, 3, 2)` gives you the result $(3^2)^2 = 81$.
- *Repeated addition:* To obtain the repeated addition of 7 to the number 3, define $g(x) = x + 7$, then `Iteration(g, 3, 4)` yields $((3+7)+7)+7 = 31$.

`Iteration(<Expression>, <Variable Name>, ..., <Start Values>, <Number of Iterations>)`

Iterates the expression n times ($n = \text{number of iterations}$) using the given start value. The result is then the last element of the output of `IterationList` Command, with the same parameters.

Example: `:Iteration(a^2+1, a, {(1+i)/(sqrt(2))}, 5)` will do a repeated iteration on a complex number

Note: See `IterationList` Command for further details.

IterationList Command

`IterationList(<Function>, <Start Value>, <Number of Iterations>)`

Gives you a list of length $n+1$ ($n = \text{number of iterations}$) whose elements are iterations of the function starting with the start value.

Example: After defining $f(x) = x^2$ the command `IterationList(f, 3, 2)` gives you the list $\{3, 9, 81\}$.

You can also use this command to define a sequence where a_{k+1} depends on a_k and k . If the input function f is a function of two variables and start value is a list of two numbers $\{s, a_s\}$, then the output list consists of numbers $a_s, a_{s+1}, \dots, a_{s+n}$ where for $k > s$ we have $a_{k+1} = f(k, a_k)$.

Example: Define $f(k, a) = (k+1) * a$, which corresponds to the recursive definition of factorial. The command `IterationList(f, {3, 6}, 4)` gives you the list $\{6, 24, 120, 720, 5040\}$.

`IterationList(<Expression>, <Variable Name>, ..., <Start Values>, <Number of Iterations>)`

Gives you a list of length $n+1$ ($n = \text{number of iterations}$) whose elements are iterations of the expression starting with the given start value. In each iteration the variables in the expression are substituted by last elements of the list. There should be at least as many start values as there are variables, otherwise the result is *undefined*.

Example: Let A, B be points. The command `IterationList(Midpoint(A, C), C, {B}, 3)` internally computes values $C_0 = B$, $C_1 = \text{Midpoint}(A, C_0)$, $C_2 = \text{Midpoint}(A, C_1)$, $C_3 = \text{Midpoint}(A, C_2)$ and yields $\{C_0, C_1, C_2, C_3\}$. Hence for $A = (0,0)$ and $B = (8,0)$ the result will be $\{(8,0), (4,0), (2,0), (1,0)\}$.

Example: Let f_0, f_1 be numbers. `IterationList(a + b, a, b, {f_0, f_1}, 5)` fills the first 2 values of the resulting list from the start values. Afterwards the values are computed as $f_2 = f_0 + f_1$, $f_3 = f_1 + f_2$, $f_4 = f_2 + f_3$, $f_5 = f_3 + f_4$. Hence for $f_0 = f_1 = 1$ the result will be $\{1, 1, 2, 3, 5, 8\}$.

Note: Only the first syntax is supported in the CAS currently

Note: See also `Iteration_Command`.

LeftSum Command

`LeftSum(<Function>, <Start x-Value>, <End x-Value>, <Number of Rectangles>)`

Calculates the left sum of the function in the interval using n rectangles.

Example: `LeftSum(x^2 + 1, 0, 2, 4)` yields $a = 3.75$

Notes: This command draws the rectangles of the left sum as well. This command is designed as a visual aid so won't give accurate answers if the number of rectangles is too large. See also the commands: RectangleSum CommandRectangleSum, TrapezoidalSum CommandTrapezoidalSum, LowerSum CommandLowerSum and UpperSum CommandUpperSum.

Limit Command

`Limit(<Function>, <Value>)`

Computes the limit of the function for the given value of the main function variable. (This may also yield infinity.)

Example: `Limit((x^2 + x) / x^2, +∞)` yields 1 .

Note: Not all limits can be calculated by GeoGebra, so *undefined* will be returned in those cases (as well as when the correct result is undefined).

CAS Syntax

`Limit(<Expression>, <Value>)`

Computes the limit of the expression for the given value of the main function variable.

Example: `Limit(a sin(x) / x, 0)` yields a .

`Limit(<Expression>, <Variable>, <Value>)`

Computes the limit of the expression for the given value of the given function variable.

Example: `Limit(a sin(v) / v, v, 0)` yields a .

Note:

- Not all limits can be calculated by GeoGebra, so ? will be returned in those cases (as well as when the correct result is undefined).
- If you want the limit of a piecewise-defined function you need to use LimitAbove or LimitBelow, for example `LimitAbove(IF(x>1, x^2, -2x), 1)`
- See also Asymptote Command, LimitAbove Command and LimitBelow Command.

LimitAbove Command

`LimitAbove(<Function>, <Value>)`

Computes the right one-sided limit of the function for the given value of the main function variable.

Example: `LimitAbove(1 / x, 0)` yields .

Note: Not all limits can be calculated by GeoGebra, so *undefined* will be returned in those cases (as well as when the correct result is undefined).

CAS Syntax

`LimitAbove(<Expression>, <Value>)`

Computes the right one-sided limit of the function for the given value of the main function variable.

Example: `LimitAbove(1 / x, 0)` yields .

`LimitAbove(<Expression>, <Variable>, <Value>)`

Computes the right one-sided limit of the multivariate function for the given value of the given function variable.

Example: `LimitAbove(1 / a, a, 0)` yields .

Note: Not all limits can be calculated by GeoGebra, so ? will be returned in those cases (as well as when the correct result is undefined).

Note: See also Limit Command and LimitBelow Command.

LimitBelow Command

`LimitBelow(<Function>, <Value>)`

Computes the left one-sided limit of the function for the given value of the main function variable.

Example: `LimitBelow(1 / x, 0)` yields .

Note: Not all limits can be calculated by GeoGebra, so *undefined* will be returned in those cases (as well as when the correct result is undefined).

CAS Syntax

`LimitBelow(<Expression>, <Value>)`

Computes the left one-sided limit of the function for the given value of the main function variable.

Example: `LimitBelow(1 / x, 0)` yields .

`LimitBelow(<Expression>, <Variable>, <Value>)`

Computes the left one-sided limit of the multivariate function for the given value of the given function variable.

Example: `LimitBelow(1 / a, a, 0)` yields .

Note: Not all limits can be calculated by GeoGebra, so ? will be returned in those cases (as well as when the correct result is undefined).

Note: See also Limit Command and LimitAbove Command.

LowerSum Command

`LowerSum(<Function>, <Start x-Value>, <End x-Value>, <Number of Rectangles>)`

Calculates the *lower sum* of the given function on the interval [*Start x-Value*, *End x-Value*], using *n* rectangles.

Example: `LowerSum(x^2, -2, 4, 6)` yields 15.

Note: This command draws the rectangles for the lower sum as well. This command is designed as a visual aid so won't give accurate answers if the number of rectangles is too large. See also the commands: `UpperSum` `CommandUpperSum`, `LeftSum` `CommandLeftSum`, `RectangleSum` `CommandRectangleSum`, and `TrapezoidalSum` `CommandTrapezoidalSum`.

NSolveODE Command

`NSolveODE(<List of Derivatives>, <Initial x-coordinate>, <List of Initial y-coordinates>, <Final x-coordinate>)`

Solves (numerically) the system of differential equations

Example:

```
f'(t, f, g, h) = g
g'(t, f, g, h) = h
h'(t, f, g, h) = -t h + 3t g + 2f + t
NSolveODE({f', g', h'}, 0, {1,2,-2}, 10)
NSolveODE({f', g', h'}, 0, {1,2,-2}, -5) (solves the system backwards in time).
```

Example:

```
x1'(t, x1, x2, x3, x4) = x2
x2'(t, x1, x2, x3, x4) = x3
x3'(t, x1, x2, x3, x4) = x4
x4'(t, x1, x2, x3, x4) = -8x1 + sin(t) x2 - 3x3 + t^2
x10 = -0.4
x20 = -0.3
x30 = 1.8
x40 = -1.5
NSolveODE({x1', x2', x3', x4'}, 0, {x10, x20, x30, x40}, 20)
```

Example:

Pendulum:

```
g = 9.8
l = 2
a = 5 (starting location)
b = 3 (starting force)
y1'(t, y1, y2) = y2
y2'(t, y1, y2) = (-g) / l sin(y1)
NSolveODE({y1', y2'}, 0, {a, b}, 20)
len = Length(numericalIntegral1)
```

```
c = Slider(0, 1, 1 / len, 1, 100, false, true, true, false)
x1 = 1 sin(y(Point(numericalIntegral1, c)))
y1 = -1 cos(y(Point(numericalIntegral1, c)))
A = (x1, y1)
Segment((0, 0), A)
StartAnimation()
```

Note: See also SlopeField command, SolveODE command.

Numerator Command

Numerator(<Function>)

Returns the numerator of the function.

Example: `Numerator((3x2 + 1) / (2x - 1))` yields $f(x) = 3x^2 + 1$.

Numerator(<Number>)

For a rational number returns its numerator. It uses a numerical method, which limits this command to numbers with small denominator. For irrational input the numerator of its continued fraction is returned.

Note: See also Denominator Command and FractionText Command.

CAS Syntax

Numerator(<Expression>)

Returns the numerator of a rational number or expression.

Examples:

- `Numerator(2/3 + 1/15)` yields 11 .
- If variables a , b and c haven't been previously defined in GeoGebra, then `Numerator(a/b)` yields a and `Numerator(Simplify(a + b/c))` yields $a c + b$

OsculatingCircle Command

OsculatingCircle(<Point>, <Function>)

Yields the osculating circle of the function in the given point.

Example: `OsculatingCircle((0, 0), x^2)` yields $x^2 + y^2 - y = 0$.

OsculatingCircle(<Point>, <Curve>)

Yields the osculating circle of the curve in the given point.

Example: `OsculatingCircle((1, 0), Curve(cos(t), sin(2t), t, 0, 2π))` yields $x^2 + y^2 + 6x - 7 = 0$.

OsculatingCircle(<Point>, <Object>)

Yields the osculating circle of the object (function, curve, conic) in the given point.

Examples:

- `OsculatingCircle((0, 0), x^2)` yields $x^2 + y^2 - y = 0$
- `OsculatingCircle((1, 0), Curve(cos(t), sin(2t), t, 0, 2π))` yields $x^2 + y^2 + 6x - 7 = 0$
- `OsculatingCircle((-1, 0), Conic({1, 1, 1, 2, 2, 3}))` yields $x^2 + y^2 + 2x + 1y = -1$

Note:

- This command is for 2D objects only. For 3D, you can make a custom tool for example <https://www.geogebra.org/m/tan7dxjt>

ParametricDerivative Command

ParametricDerivative(<Curve>)

Returns a new parametric curve given by .

Example: `ParametricDerivative(Curve(2t, t^2, t, 0, 10))` returns the parametric curve $(x(t) = 2t, y(t) = t^2)$. The curve given as argument to the command is the function $f(x) =$, and the result is the derivative of that function: $f'(x) =$.

PartialFractions Command

`PartialFractions(<Function>)`

Yields, if possible, the partial fraction of the given function for the main function variable. The graph of the function is plotted in the Graphics View.

Example: `PartialFractions(x^2 / (x^2 - 2x + 1))` yields $1 + \frac{1}{x-1}$.

Hint: In the CAS View you can also use the following syntax: `PartialFractions(<Function>, <Variable>)` Yields, if possible, the partial fraction of the given function for the given function variable. Example: `PartialFractions(a^2 / (a^2 - 2a + 1), a)` yields $1 + \frac{1}{a-1} + \frac{2}{(a-1)^2}$.

PathParameter Command

`PathParameter(<Point On Path>)`

Returns the parameter (i.e. a number ranging from 0 to 1) of the point that belongs to a path.

Example:

Let $f(x) = x^2 + x - 1$ and $A = (1, 1)$ is a point attached to this function.

`PathParameter(A)` yields $a = 0.47$.

In the following table is a function used to map all real numbers into interval (-1,1) and is a linear map from line AB to reals which sends A to 0 and B to 1.

Line AB

Ray AB

Segment AB

Circle with center C and radius r

Point , where has path parameter

Ellipse with center C and semiaxes ,

Point , where has path parameter

Hyperbola

Point has path parameter or

Parabola with vertex V and direction of axis . Point has path parameter .

Polyline $A_1 \dots A_n$

If X lies on $A_k A_{k+1}$, path parameter of X is

Polygon $A_1 \dots A_n$

If X lies on $A_k A_{k+1}$ (using $A_{n+1}=A_1$), path parameter of X is

List of paths $L=\{p_1, \dots, p_n\}$

If X lies on p_k and path parameter of X w.r.t. p_k is t , path parameter of X w.r.t. L is

List of points $L=\{A_1, \dots, A_n\}$

Path parameter of A_k is . Point[L,t] returns .

Locus

Implicit polynomial

No formula available.

Polynomial Command

Polynomial(<Function>)

Yields the expanded polynomial function.

Example: `Polynomial((x - 3)^2)` yields $x^2 - 6x + 9$.

Polynomial(<List of Points>)

Creates the interpolation polynomial of degree $n-1$ through the given n points.

Example: `Polynomial({(1, 1), (2, 3), (3, 6)})` yields $0.5x^2 + 0.5x$.

CAS Syntax

Polynomial(<Function>)

Expands the function and writes it as a polynomial in x (grouping the coefficients).

Example: `Polynomial((x - 3)^2 + (a + x)^2)` yields $2x^2 + (2a - 6)x + a^2 + 9$.

Polynomial(<Function>, <Variable>)

Expands the function and writes it as a polynomial in the variable (grouping the coefficients).

Example: `Polynomial((x - 3)^2 + (a + x)^2, a)` yields $a^2 + 2xa + 2x^2 - 6x + 9$.

RectangleSum Command

RectangleSum(<Function>, <Start x-Value>, <End x-Value>, <Number of Rectangles>, <Position for rectangle start>)

Calculates between the *Start x-Value* and the *End x-Value* the sum of rectangles with left height starting at a fraction d ($0 \leq d \leq 1$) of each interval, using n rectangles.

When $d = 0$ it is equivalent to the LeftSum command, and when $d = 1$ it computes the right sum of the given function.

Note: This command draws the rectangles of the left sum as well. This command is designed as a visual aid so won't give accurate answers if the number of rectangles is too large. See also the commands: UpperSum, CommandUpperSum, LowerSum, CommandLowerSum, LeftSum, CommandLeftSum, TrapezoidalSum, CommandTrapezoidalSum.

Root Command

Root(<Polynomial>)

Yields all roots of the polynomial as intersection points of the function graph and the x -axis.

Example: `Root (0.1*x^2 - 1.5*x + 5)` yields $A = (5, 0)$ and $B = (10, 0)$.

Root(<Function>, <Initial x-Value>)

Yields one root of the function using the initial value a for a numerical iterative method.

Example: `Root (0.1*x^2 - 1.5*x + 5, 6)` yields $A = (5, 0)$.

Root(<Function>, <Start x-Value>, <End x-Value>)

Let a be the *Start x-Value* and b the *End x-Value*. This command yields one root of the function in the interval $[a, b]$ using a numerical iterative method.

Example: `Root (0.1*x^2 - 1.5*x + 5, 8, 13)` yields $A = (10, 0)$.

CAS Syntax

Root(<Polynomial>)

Yields all roots of the polynomial as intersection points of the function graph and the x -axis.

Example: `Root (x^3 - 3 * x^2 - 4 * x + 12)` yields $\{x = -2, x = 2, x = 3\}$.

Note:

In the CAS View, this command is only a special variant of Solve Command.

RootList Command

RootList(<List>)

Converts a given list of numbers $\{a_1, a_2, \dots, a_n\}$ to a list of points $\{(a_1, 0), (a_2, 0), \dots, (a_n, 0)\}$, which is also displayed in the Graphics View.

Example:

Command `RootList ({3, 4, 5, 2, 1, 3})` returns the list of points `list1={ (3,0), (4,0), (5,0), (2,0), (1,0), (3,0) }`

Roots Command

Roots(<Function>, <Start x-Value>, <End x-Value>)

Calculates the roots for function in the given interval. The function must be continuous on that interval. Because this algorithm is numeric, it may not find all the roots in some cases.

Example:

`Roots(f, -2, 1)` with the function $f(x) = 3x^3 + 3x^2 - x$ yields $A = (-1.264, 0)$, $B = (0, 0)$, $C = (0.264, 0)$

Note: See also Root command

SlopeField Command

SlopeField(<f(x,y)>)

Plots a slope field for the differential equation

Example: `SlopeField(x+y)` plots the slope field.

SlopeField(<f(x,y)>, <Number n>)

Plots a slopefield for the differential equation on an n by n grid (if the Graphics View is square) or a smaller grid if not. Default is 40.

SlopeField(<f(x,y)>, <Number n>, <Length Multiplier a>)

Plots a slopefield for the differential equation . The Length Multiplier $0 < a \leq 1$ determines how long the segments are.

SlopeField(<f(x,y)>, <Number n>, <Length Multiplier a>, <Min x>, <Min y>, <Max x>, <Max y>)

Plots a slopefield for the differential equation inside the specified rectangle (rather than filling the Graphics View)

Notes: Use the following tools: Move_Graphics_View_ToolMove Graphics View, Zoom_In_ToolZoom In, Zoom_Out_ToolZoom Out and observe the effect. See also SolveODE CommandSolveODE, Locus CommandLocus, Integral CommandIntegral

SolveODE Command

SolveODE(<f(x, y)>)

Attempts to find the exact solution of the first order ordinary differential equation (ODE) .

Example: `SolveODE (2x / y)` yields , where is a constant.

Note: will be created as an auxiliary object with a corresponding slider.

SolveODE(<f(x, y)>, <Point on f>)

Attempts to find the exact solution of the first order ODE and use the solution which goes through the given point.

Example: `SolveODE (y / x, (1, 2))` yields $y = 2x$.

SolveODE(<f(x, y)>, <Start x>, <Start y>, <End x>, <Step>)

Solves first order ODE numerically with given start point, end and step for x .

Example: `SolveODE (-x*y, x(A), y(A), 5, 0.1)` solves using previously defined A as a starting point.

Note: Length CommandLength(<Locus>) allows you to find out how many points are in the computed locus. First CommandFirst(<Locus>, <Number>) allows you to extract the points as a list. To find the "reverse" solution, just enter a negative value for End x, for example `SolveODE(-x*y, x(A), y(A), -5, 0.1)`

SolveODE(<y'>, <x'>, <Start x>, <Start y>, <End t>, <Step>)

Solves first order ODE with given start point, maximal value of an internal parameter t and step for t . This version of the command may work where the first one fails e.g. when the solution curve has vertical points.

Example: `SolveODE (-x, y, x(A), y(A), 5, 0.1)` solves using previously defined A as a starting point.

Note: To find the "reverse" solution, just enter a negative value for End t , for example `SolveODE (-x, y, x(A), y(A), -5, 0.1)` .

SolveODE(<b(x)>, <c(x)>, <f(x)>, <Start x>, <Start y>, <Start y'>, <End x>, <Step>)

Solves second order ODE .

Example: `SolveODE (x^2, 2x, 2x^2 + x, x(A), y(A), 0, 5, 0.1)` solves the second order ODE using previously defined A as a starting point.

Note: Always returns the result as locus. The algorithms are currently based on Runge-Kutta numeric methods.

Note: See also SlopeField command.

CAS Syntax

SolveODE(<Equation>)

Attempts to find the exact solution of the first or second order ODE. For first and second derivative of y you can use y' and y'' respectively.

Example: `SolveODE (y' = y / x)` yields $y = c_1 x$.

SolveODE(<Equation>, <Point(s) on f>)

Attempts to find the exact solution of the given first or second order ODE which goes through the given point(s).

Example: `SolveODE (y' = y / x, (1, 2))` yields $y = 2x$.

SolveODE(<Equation>, <Point(s) on f>, <Point(s) on f'>)

Attempts to find the exact solution of the given first or second order ODE and goes through the given *point(s) on f* and *f* goes through the given *point(s) on f'*.

Example: SolveODE (y'' - 3y' + 2 = x, (2, 3), (1, 2)) yields .

SolveODE(<Equation>, <Dependent Variable>, <Independent Variable>, <Point(s) on f>)

Attempts to find the exact solution of the given first or second order ODE which goes through the given point(s).

Example: SolveODE (v' = v / w, v, w, (1, 2)) yields $v = 2w$.

SolveODE(<Equation>, <Dependent Variable>, <Independent Variable>, <Point(s) on f>, <Point(s) on f'>)

Attempts to find the exact solution of the given first or second order ODE which goes through the given *point(s) on f* and *f* goes through the given *point(s) on f'*.

Example: SolveODE (v' = v / w, v, w, (1, 2), (0, 2)) yields $v = 2w$.

Note: For compatibility with input bar, if the first parameter is just an expression without y' or y'' , it is supposed to be right hand side of ODE with left hand side y' .

Spline Command

Spline(<List of Points>)

Creates a cubic spline through all points.

Spline(<List of Points>, <Order ≥ 3 >)

Creates a spline with given order through all points.

Spline(<List of Points>, <Order ≥ 3 >, <Weight Function>)

Creates a spline with given order through all points. The weight function says what should be the difference of t values for point P_i and P_{i+1} given their difference $P_{i+1} - P_i = (x, y)$. To get the spline you expect from "function" algorithm you should use $\text{abs}(x)+0*y$, to get the GeoGebra's default spline you can use $\sqrt{x^2+y^2}$.

Note: The choice of default makes the result behave nicely when transformed, making sure that $\text{Rotate}(\text{Spline}(\text{list}), a)$ gives the same as $\text{Spline}(\text{Rotate}(\text{list}, a))$.

TaylorPolynomial Command

`TaylorPolynomial(<Function>, <x-Value>, <Order Number>)`

Creates the power series expansion for the given function at the point *x-Value* to the given order.

Example: `TaylorPolynomial(x^2, 3, 1)` gives $9 + 6(x - 3)$, the power series expansion of x^2 at $x = 3$ to order 1.

CAS Syntax

`TaylorPolynomial(<Expression>, <x-Value>, <Order Number>)`

Creates the power series expansion for the given expression at the point *x-Value* to the given order.

Example: `TaylorPolynomial(x^2, a, 1)` gives $a^2 + 2a(x - a)$, the power series expansion of x^2 at $x = a$ to order 1.

`TaylorPolynomial(<Expression>, <Variable>, <Variable Value>, <Order Number>)`

Creates the power series expansion for the given expression with respect to the given variable at the point *Variable Value* to the given order.

Examples:

- `TaylorPolynomial(x^3 sin(y), x, 3, 2)` gives $27 \sin(y) + 27 \sin(y)(x - 3) + 9 \sin(y)(x - 3)^2$, the power series expansion with respect to *x* of $x^3 \sin(y)$ at $x = 3$ to order 2.
- `TaylorPolynomial(x^3 sin(y), y, 3, 2)` gives $x^3 \sin(3) + x^3 \cos(3)(y - 3) - x^3(y - 3)^2$, the power series expansion with respect to *y* of $x^3 \sin(y)$ at $y = 3$ to order 2.

Note: The order has got to be an integer greater or equal to zero.

TrapezoidalSum Command

TrapezoidalSum(<Function>, <Start x-Value>, <End x-Value>, <Number of Trapezoids>)

Calculates the trapezoidal sum of the function in the interval [*Start x-Value*, *End x-Value*] using *n* trapezoids.

Example: `TrapezoidalSum(x^2, -2, 3, 5)` yields 12.5.

Notes: This command draws the trapezoids of the trapezoidal sum as well. This command is designed as a visual aid so won't give accurate answers if the number of rectangles is too large. See also the commands: LowerSum CommandLowerSum, LeftSum CommandLeftSum, RectangleSum CommandRectangleSum and UpperSum CommandUpperSum.

TrigExpand Command

TrigExpand(<Expression>)

Transforms a trigonometric expression into an expression using only simple variables as arguments.

Example: `TrigExpand(tan(x + y))` gives .

It can also expand a product into a linear expression

Example: `TrigExpand(sin(x)sin(x/3))` gives .

TrigExpand(<Expression>, <Target Function>)

Transforms a trigonometric expression into an expression using only simple variables as arguments, preferring the given target function.

Example: `TrigExpand(tan(x + y), tan(x))` gives .

CAS Syntax

CAS syntaxes may show different results, depending on the selected output mode:

Example: `TrigExpand(tan(x + y))`

in *Evaluate* mode gives

in *Numeric* mode gives .

The following commands are only available in the CAS View:

TrigExpand(<Expression>, <Target Function>, <Target Variable>)

Transforms a trigonometric expression into an expression using only simple variables as arguments, preferring the given target function and target variable.

Examples:

- `TrigExpand(sin(x), sin(x), x/2)` gives
- `TrigExpand(sin(x)/(1+cos(x)), tan(x), x/2)` gives .

TrigExpand(<Expression>, <Target Function>, <Target Variable>, <Target Variable>)

Transforms a trigonometric expression into an expression using only simple variables as arguments, preferring the given target function and target variables.

Example: `TrigExpand(csc(x) - cot(x) + csc(y) - cot(y), tan(x), x/2, y/2)` gives .

Note: See also TrigSimplify Command and TrigCombine Command.

TrigCombine Command

TrigCombine(<Expression>)

Transforms a trigonometric expression including products of trigonometric functions into an expression without products involving sums of variables.

Examples:

- `TrigCombine(sin(x) cos(3x))` gives
- `TrigCombine(sin(x) + cos(x))` gives .

TrigCombine(<Expression>, <Target Function>)

Transforms a trigonometric expression including products of trigonometric functions into an expression without products involving sums of variables, preferring the given target function.

Example: `TrigCombine(sin(x) + cos(x), sin(x))` gives .

Notes:

- In the CAS View undefined variables can be used as well.
- **Example:** `TrigCombine(sin(p) cos(3p))` gives .
- See also TrigExpand Command and TrigSimplify Command.

TrigSimplify Command

TrigSimplify(<Expression>)

Simplifies the given trigonometric expression.

Examples:

- `TrigSimplify(1 - sin(x)^2)` gives $\cos^2(x)$.
- `TrigSimplify(sin(x)^2 - cos(x)^2 + 1)` gives $2 \sin^2(x)$.

Note: This command works only for variables "x", "y" and "z" in the Algebra View

CAS Syntax

TrigSimplify(<Expression>)

Simplifies the given trigonometric expression.

Examples:

- `TrigSimplify(1 - sin(x)^2)` gives $\cos^2(x)$
- `TrigSimplify(sin(x)^2 - cos(x)^2 + 1)` gives $2 \sin^2(x)$.

Note:

- This command works for all variables in the CAS View
- See also TrigExpand Command and TrigCombine Command.

InflectionPoint Command

InflectionPoint(<Polynomial>)

Yields all inflection points of the polynomial as points on the function graph.

Example:

InflectionPoint [x^3] yields (0, 0).

CAS Syntax

InflectionPoint(<Function>)

Yields all inflection points of the function (where possible) as a list.

Example:

InflectionPoint [x exp (-x)] yields .

UpperSum Command

UpperSum(<Function>, <Start x-Value>, <End x-Value>, <Number of Rectangles>)

Calculates the *upper sum* of the function on the interval [*Start x-Value*, *End x-Value*] using *n* rectangles.

Example: UpperSum (x^2, -2, 4, 6) yields 35.

Notes: This command draws the rectangles of the upper sum as well. This command is designed as a visual aid so won't give accurate answers if the number of rectangles is too large. See also the commands: LowerSum CommandLowerSum, LeftSum CommandLeftSum, RectangleSum CommandRectangleSum, and TrapezoidalSum CommandTrapezoidalSum.

Conic Commands

<links/>

See also Conic section tools.

Axes Command

`Axes(<Conic>)`

Returns the equations of the major and minor axes of a conic section.

Note: See also MajorAxis and MinorAxis commands.

`Axes(<Quadric>)`

Creates the 3 axes of the given quadric.

Example:

`Axes(x^2 + y^2 + z^2 = 3)` returns the three lines

$a: X = (0, 0, 0) + \lambda (1, 0, 0)$, $b: X = (0, 0, 0) + \lambda (0, 1, 0)$ and $c: X = (0, 0, 0) + \lambda (0, 0, 1)$

Notes: Specifically:if the given quadric is a cylinder, the command yields the two axes of the bottom circle and the rotation axis.if the given quadric is a sphere, the command yields the three axes parallel to the coordinate system axes.

Center Command

`Center(<Conic>)`

Returns the center of a circle, ellipse, or hyperbola.

Example: `Center(x^2 + 4 y^2 + 2x - 8y + 1 = 0)`

`(, : Centre(x^2 + 4 y^2 + 2x - 8y + 1 = 0))`

returns point $A = (-1, 1)$

Note: See also Midpoint or Center (, : Midpoint or Centre) tool .

`Center(<Quadric>)`

Creates the center of a quadric (e.g. sphere, cone, etc.).

Example:

`Center(x^2 + (y-1)^2 + (z-2)^2 = 1)` yields $(0, 1, 2)$

Circle Command

`Circle(<Point>, <Radius Number>)`

Yields a circle with given center and radius.

`Circle(<Point>, <Segment>)`

Yields a circle with given center and radius equal to the length of the given segment.

`Circle(<Point>, <Point>)`

Yields a circle with given center through a given point.

`Circle(<Point>, <Point>, <Point>)`

Yields a circle through the three given points (if they do not lie on the same line).

Note: See also Compass, Circle with Center through Point, Circle with Center and Radius, and Circle through 3 Points tools.

`Circle(<Line>, <Point>)`

Creates a circle with line as axis and through the point.

`Circle(<Point>, <Radius>, <Direction>)`

Creates a circle with center, radius, and axis parallel to direction, which can be a line, vector or plane.

Example:

`Circle(<Point>, <Radius>, <Plane>)` yields a circle parallel to the plane and with perpendicular vector of the plane as axis.

`Circle(<Point>, <Point>, <Direction>)`

Creates a circle with center, through a point, and axis parallel to direction.

Note: If you use eg $x = 0$ or $y = 0$ for the *Direction* it will be interpreted as a plane, not a line

Conic Command

Conic(<Point>, <Point>, <Point>, <Point>, <Point>)

Returns a conic section through the five given points.

Example: `Conic((0, -4), (2, 4), (3, 1), (-2, 3), (-3, -1))` yields $151x^2 - 37x y + 72y^2 + 14x - 42y = 1320$.

Note: If four of the points lie on one line, then the conic section is not defined.

Conic(<Number a>, <Number b>, <Number c>, <Number d>, <Number e>, <Number f>)

Returns a conic section .

Example: `Conic(2, 3, -1, 4, 2, -3)` yields $2x^2 + 4x y + 3y^2 + 2x - 3y = 1$.

Conic(<List>)

Returns a conic section .

Example: `Conic({2, 3, -1, 4, 2, -3})` yields $2x^2 + 4x y + 3y^2 + 2x - 3y = 1$.

Note: See also Conic through 5 Points tool and Coefficients command.

ConjugateDiameter Command

ConjugateDiameter(<Line>, <Conic>)

Returns the ([w:Conjugate diameters|conjugate diameter]) of the diameter that is parallel to the given line (relative to the conic section).

Example: `ConjugateDiameter(-4x + 5y = -2, x^2 + 4 y^2 + 2x - 8y + 1 = 0)`
yields line $5x + 16y = 11$

ConjugateDiameter(<Vector>, <Conic>)

Returns the ([w:Conjugate diameters|conjugate diameter]) of the diameter that is parallel to the given vector (relative to the conic section).

Example: Let $u = (4,1)$ be a vector. Then `ConjugateDiameter(u, x^2 + 4 y^2 + 2x - 8y + 1 = 0)` yields line $x + y = 0$

Directrix Command

Directrix(<Conic>)

Yields the directrix of the conic.

Example: `Directrix(x^2 - 3x + 3y = 9)` yields the line $y = 4.5$

Note: See also the Focus command.

Eccentricity Command

Eccentricity(<Conic>)

Calculates the eccentricity of the conic section.

Example: `Eccentricity(x^2/9 + y^2/4 = 1)` returns $a = 0.75$

Ellipse Command

Ellipse(<Focus>, <Focus>, <Semimajor Axis Length>)

Creates an ellipse with two focal points and semimajor axis length.

Example: `Ellipse((0, 1), (1, 1), 1)` yields $12x^2 + 16y^2 - 12x - 32y = -7$.

Note: If the condition: $2 * \text{semimajor axis length} > \text{Distance between the focus points}$ isn't met, you will get an hyperbola.

Ellipse(<Focus>, <Focus>, <Segment>)

Creates an ellipse with two focal points, where the length of the semimajor axis equals the length of the given segment.

Example: Let $s = \text{Segment}((0,1), (2,1))$: `Ellipse((0, 1), (2, 1), s)` yields $3x^2 + 4y^2 - 6x - 8y = 5$.

Ellipse(<Focus>, <Focus>, <Point>)

Creates an ellipse with two focal points passing through a given point.

Example: `Ellipse((0, 1), (2, 1), (1, 2))` yields $1x^2 + 2y^2 - 2x - 4y = -1$.

Note: See also Ellipse tool .

LinearEccentricity Command

LinearEccentricity(<Conic>)

Calculates the linear eccentricity of the conic section.

For ellipses or hyperolas the command gives the distance between the conic's center and one of its foci, for circles it gives 0, and for parabolas gives the distance between its focus and the vertex.

Example: `LinearEccentricity(4x^2 - y^2 + 16x + 20 = 0)` returns 2.24

MajorAxis Command

MajorAxis(<Conic>)

Returns the equation of the major axis of the conic section.

Example: `MajorAxis(x^2 / 9 + y^2 / 4 = 1)` returns $y = 0$.

Note: See also MinorAxis command.

SemiMajorAxisLength Command

SemiMajorAxisLength(<Conic>)

Returns the length of the semimajor axis (half of the major axis) of the conic section.

Example:

`SemiMajorAxisLength((x - 1)^2 + (y - 2)^2 = 4)` yields 2.

Note: See also SemiMinorAxisLength command.

Focus Command

Focus(<Conic>)

Yields (all) foci of the conic section.

Example: Focus($4x^2 - y^2 + 16x + 20 = 0$) returns the two foci of the given hyperbola: $A=(-2, -2.24)$ and $B=(-2, 2.24)$.

Note: See also the Directrix command.

Hyperbola Command

Hyperbola(<Focus>, <Focus>, <Semimajor Axis Length>)

Creates a hyperbola with given focus points and semimajor axis length.

Example: Hyperbola((0, -4), (2, 4), 1) yields $-8xy - 15y^2 + 8y = -16$.

Note: If the condition: $0 < 2 * \text{semimajor axis length} < \text{Distance between the focus points}$ isn't met, you will get an ellipse.

Hyperbola(<Focus>, <Focus>, <Segment>)

Creates a hyperbola with given focus points where the length of the semimajor axis equals the length of the segment.

Example: Let $a = \text{Segment}((0, 1), (2, 1))$. Hyperbola((4, 1), (-2, 1), a) yields $-5x^2 + 4y^2 + 10x - 8y = -19$.

Hyperbola(<Focus>, <Focus>, <Point>)

Creates a hyperbola with given focus points passing through a given point.

Example: Hyperbola((1, 1), (2, 1), (-2, -4)) yields $-2.69x^2 + 1.30y^2 + 8.07x - 2.62y = 4.52$.

Note: See also Hyperbola tool .

Incircle Command

`Incircle(<Point>, <Point>, <Point>)`

Returns Incircle of the triangle formed by the three Points.

Example: Let $O=(0, 0)$, $A=(3, 0)$ and $B=(0, 5)$ be three points: `Incircle(O, A, B)` yields $(x - 1.08)^2 + (y - 1.08)^2 = 1.18$ in *Algebra View* and draws the corresponding circle in *Graphics View*.

Parabola Command

`Parabola(<Point>, <Line>)`

Returns a parabola with focal point and the line as directrix.

Example: Let $a = \text{Line}((0,1), (2,1))$. `Parabola((3, 3), a)` yields $x^2 - 6x - 4y = -17$.

Note: See also Parabola tool .

Parameter Command

`Parameter(<Parabola>)`

Returns the parameter of the parabola, which is the distance between the directrix and the focus.

Example: `Parameter(y = x^2 - 3x + 5)` returns 0.5.

Polar Command

`Polar(<Point>, <Conic>)`

Creates the polar line of the given point relative to the conic section.

Example: `Polar((0,2), y = x^2 - 3x + 5)` creates the line $1.5x + 0.5y = 4$

Note: See also Polar or Diameter Line tool.

`Polar(<Line>, <Conic>)`

Creates the pole, given a polar line and a conic.

Example: `Polar(1.5x+0.5y=4, y = x^2 - 3x + 5)` creates the point $(0, 2)$

MinorAxis Command

MinorAxis(<Conic>)

Returns the equation of the minor axis of the conic section.

Example: `MinorAxis(x^2 / 9 + y^2 / 4 = 1)` returns $x = 0$.

Note: See also MajorAxis command.

SemiMinorAxisLength Command

SemiMinorAxisLength(<Conic>)

Returns the length of the semiminor axis (half of the minor axis) of the conic section.

Example:

`SemiMinorAxisLength(x^2 + 2y^2 - 2x - 4y = 5)` yields 2.

Note: See also SemiMajorAxisLength command.

Semicircle Command

Semicircle(<Point>, <Point>)

Creates a semicircle above the segment between the two points and displays its length in *Algebra View*.

Note: See also Semicircle tool.

List Commands

<links/>

Append Command

Append(<List>, <Object>)

Appends the object to the list and yields the results in a new list.

Example: Append({1, 2, 3}, 4) creates the list {1, 2, 3, 4}.

Append(<Object>, <List>)

Appends the list to the object and yields the results in a new list.

Example: Append(4, {1, 2, 3}) creates he list {4, 1, 2, 3}.

Element Command

Element(<List>, <Position of Element n>)

Yields the n^{th} element of the list.

Example:

Element({1, 3, 2}, 2) yields 3, the second element of {1, 3, 2}.

Note: In the CAS View undefined variables can be used as well. Example: Element({a, b, c}, 2) yields b, the second element of {a, b, c}.

Element(<Matrix>, <Row>, <Column>)

Yields the element of the matrix in the given row and column.

Example:

Element({{1, 3, 2}, {0, 3, -2}}, 2, 3) yields -2, the third element of the second row of .

Note: In the CAS View undefined variables can be used as well. Example: Element({{a, b, c}, {d, e, f}}, 2, 3) yields f, the third element of the second row of \begin{pmatrix} a&b&c \\ d&e&f \end{pmatrix}.

Element(<List>, <Index1>, <Index2>, ...)

Provided list is n -dimensional list, one can specify up to n indices to obtain an element (or list of elements) at given coordinates.

Example:

Let $L = \{\{1, 2\}, \{3, 4\}\}, \{\{5, 6\}, \{7, 8\}\}$.

Then Element(L, 1, 2, 1) yields 3, Element(L, 2, 2) yields {7, 8}.

Note: This command only works, if the list or matrix contains elements of one object type (e. g. only numbers or only points). See also First Command, Last Command and RandomElement Command.

Flatten Command

Flatten(<List>)

Flattens lists to one list.

Example: Flatten({2, 3, {5, 1}, {{2, 1, {3}}}}) yields *list1* = {2, 3, 5, 1, 2, 1, 3}.

First Command

First(<List>)

Gives a new list that contains the first element of the given list.

Example:

First({1, 4, 3}) yields {1}.

Note: To get the first element use Element({1, 4, 3}, 1).

First(<List>, <Number n of elements>)

Gives a new list that contains just the first *n* elements of the given list.

Example:

First({1, 4, 3}, 2) yields {1, 4}.

First(<Text>)

Gives first character of the text.

Example:

First("Hello") yields "H".

First(<Text> , <Number n of elements>)

Gives the first *n* characters of the text.

Example:

First("Hello", 2) yields "He".

First(<Locus>, <Number n of elements>)

This command is useful for

- loci generated by NSolveODE Command - It returns list points that were created in the first *n* steps of the numeric ODE-solving algorithm.
- loci generated using ShortestDistance Command, TravelingSalesman Command, Voronoi Command, MinimumSpanningTree Command and ConvexHull Command Commands - It returns vertices of the graph.

Note:

See also Last Command.

IndexOf Command

`IndexOf(<Object>, <List>)`

Returns position of first occurrence of Object in List.

Examples: `IndexOf(5, {1, 3, 5, 2, 5, 4})` returns 3.

Note: When the object is not found, result is *undefined*.

`IndexOf(<Object>, <List>, <Start Index>)`

Same as above, but the search starts at given index.

Examples:

- `IndexOf(5, {1, 3, 5, 2, 5, 4}, 3)` returns 3.
- `IndexOf(5, {1, 3, 5, 2, 5, 4}, 4)` returns 5.
- `IndexOf(5, {1, 3, 5, 2, 5, 4}, 6)` returns *undefined*.

`IndexOf(<Text>, <Text>)`

Specifies the position at which the short text appears for the first time in the whole text.

Example: `IndexOf("Ge", "GeoGebra")` returns 1.

`IndexOf(<Text>, <Text>, <Start Index>)`

Same as above, but the search starts at given index.

Example: `IndexOf("Ge", "GeoGebra", 2)` returns 4.

Insert Command

`Insert(<Object>, <List>, <Position>)`

Inserts the object in the list at the given position.

Example:

`Insert(x^2, {1, 2, 3, 4, 5}, 3)` places x^2 at the third position and creates the list {1, 2, x^2 , 3, 4, 5}.

Note: If the position is a negative number, then the position is counted from the right.

Example:

`Insert(x^2, {1, 2, 3, 4, 5}, -1)` places x^2 at the end of the list and creates the list {1, 2, 3, 4, 5, x^2 }.

`Insert(<List>, <List>, <Position>)`

Inserts all elements of the first list in the second list at the given position.

Example:

`Insert({11, 12}, {1, 2, 3, 4, 5}, 3)` places the elements of the first list at the third (and following) position(s) of the second list and creates the list {1, 2, 11, 12, 3, 4, 5}.

Note: If the position is a negative number, then the position is counted from the right.

Example:

`Insert({11, 12}, {1, 2, 3, 4, 5}, -2)` places the elements of the first list at the end of the second list before its last element and creates the list {1, 2, 3, 4, 11, 12, 5}.

Intersection Command

Intersection(<List>, <List>)

Gives you a new list containing all elements that are part of both lists.

Example:

Let `list1 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}` and `list2 = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30}` be two lists.
Intersection(`list1, list2`) yields a new list `list3 = {2, 4, 6, 8, 10, 12, 14}`.

Join Command

Join(<List>, <List>, ...)

Joins the two (or more) lists.

Note: The new list contains all elements of the initial lists even if they are the same. The elements of the new list are not re-ordered.

Example:

`Join({5, 4, 3}, {1, 2, 3})` creates the list `{5, 4, 3, 1, 2, 3}`.

Join(<List of Lists>)

Joins the sub-lists into one longer list.

Note: The new list contains all elements of the initial lists even if they are the same. The elements of the new list are not re-ordered.

Examples: `Join({{1, 2}})` creates the list `{1, 2}`. `Join({{1, 2, 3}, {3, 4}, {8, 7}})` creates the list `{1, 2, 3, 3, 4, 8, 7}`.

Last Command

Last(<List>)

Gives a new list that contains the last element of the initial list.

Example:

 Last({1, 4, 3}) yields {3}.

Note: To get the last element use Element({1, 4, 3}, 3).

Last(<List>, <Number of elements>)

Gives a new list that contains just the last n elements of the initial list.

Example:

 Last({1, 4, 3}, 2) yields {4, 3}.

Last(<Text>)

Gives last character of the text.

Example:

 Last("Hello") yields "o".

Last(<Text>, <Number of elements>)

Gives the last n characters of the text.

Example:

 Last("Hello", 2) yields "lo".

Note:

See also First Command.

OrdinalRank Command

OrdinalRank(<List>)

Returns a list, whose i -th element is the rank of i -th element of list L (rank of element is its position in Sort(L)). If there are more equal elements in L which occupy positions from k to l in Sort(L), ranks from k to l are associated with these elements.

Example:

- `OrdinalRank({4, 1, 2, 3, 4, 2})` returns $\{5, 1, 2, 4, 6, 3\}$
- `OrdinalRank({3, 2, 2, 1})` returns $\{4, 2, 3, 1\}$

Note: Also see command: TiedRank

PointList Command

PointList(<List>)

Creates list of points from a list of two-element lists.

Example: `PointList({{1,2}, {3,4}})` returns $\{(1,2), (3,4)\}$.

Product Command

Product(<List of Raw Data>)

Calculates the product of all numbers in the list.

Example:

`Product({2, 5, 8})` yields 80.

Product(<List of Numbers>, <Number of Elements>)

Calculates the product of the first n elements in the list.

Example:

`Product({1, 2, 3, 4}, 3)` yields 6.

Product(<List of Numbers>, <List of Frequencies>)

Calculates the product of all elements in the *list of numbers* raised to the value given in the *list of frequencies* for each one of them.

Examples:

`Product({20, 40, 50, 60}, {4, 3, 2, 1})` yields 1536000000000000

`Product({sqrt(2), cbrt(3), sqrt(5), cbrt(-7)}, {4, 3, 2, 3})` yields -420

Note: The two lists must have the same length.

CAS Syntax

`Product(<List of Expressions>)`

Calculates the product of all elements in the list.

Example:

`Product({1, 2, x})` yields $2x$.

`Product(<Expression>, <Variable>, <Start Index>, <End Index>)`

Calculates the product of the expressions that are obtained by replacing the given variable with every integer from *start* to *end*.

Example:

`Product(x + 1, x, 2, 3)` yields 12 .

RandomElement Command

`RandomElement(<List>)`

Returns randomly chosen element from the list (with uniform probability). All elements in the list must be of the same type.

Example:

`RandomElement({3, 2, -4, 7})` yields one of $\{-4, 2, 3, 7\}$.

Hint: In the CAS View this command also works with symbolic input. Example: `RandomElement({a,b,c,d})` yields one of $\{a, b, c, d\}$.

Note:

See also Element Command, SetSeed Command, RandomBetween Command, RandomBinomial Command, RandomNormal Command, RandomPoisson Command and RandomUniform Command.

Remove Command

Remove(<List>, <List>)

Removes objects from the first list each time they appear in the second list.

Example:

Remove({1, 3, 4, 4, 9}, {1, 4, 5}) yields list {3, 4, 9}.

Note: See also RemoveUndefined Command. You can also type {1,3,4,4,9} \ {1,4,5} if you want the set-theoretic difference .

RemoveUndefined Command

RemoveUndefined(<List>)

Removes undefined objects from a list.

Example:

RemoveUndefined(Sequence((-1)^i, i, -3, -1, 0.5)) removes the second and fourth element of the sequence since expressions and are undefined and yields list {-1, 1, -1}.

Note:

See also Remove Command.

Reverse Command

Reverse(<List>)

Reverses the order of a list.

Example:

Reverse(list1) reverses list1 = {(1, 2), (3, 4), (5, 6)} to create list2 = {(5, 6), (3, 4), (1, 2)}

CAS Syntax

Reverse(<List>)

Reverses the order of a list.

Example:

Reverse({1, 2, 3, 4}) reverses the list to create {4, 3, 2, 1}

SelectedElement Command

`SelectedElement(<List>)`

Returns the selected element in a drop-down list.

Note: See also SelectedIndex command

SelectedIndex Command

`SelectedIndex(<List>)`

Returns the index of the selected element of a drop-down list.

Note: See also SelectedElement command

Note: `SetValue(<Action_Objects#Drop-down listsdrop-down list>, <Number n >)` Set *n* as SelectedIndex_Command the index of the selected element in the drop-down list.

Sequence Command

`Sequence(<Expression>, <Variable k>, <Start Value a>, <End Value b>)`

Yields a list of objects created using the given expression and the index *k* that ranges from start value *a* to end value *b*.

Examples:

- `Sequence((2, k), k, 1, 5)` creates a list of points whose y-coordinates range from 1 to 5: $\{(2, 1), (2, 2), (2, 3), (2, 4), (2, 5)\}$
- `Sequence(x^k, k, 1, 10)` creates the list $\{x, x^2, x^3, x^4, x^5, x^6, x^7, x^8, x^9, x^{10}\}$

`Sequence(<Expression>, <Variable k>, <Start Value a>, <End Value b>, <Increment>)`

Yields a list of objects created using the given expression and the index *k* that ranges from start value *a* to end value *b* with given increment.

Examples:

- `Sequence((2, k), k, 1, 3, 0.5)` creates a list of points whose y-coordinates range from 1 to 3 with an increment of 0.5: $\{(2, 1), (2, 1.5), (2, 2), (2, 2.5), (2, 3)\}$
- `Sequence(x^k, k, 1, 10, 2)` creates the list $\{x, x^3, x^5, x^7, x^9\}$.

Note: Since the parameters *a* and *b* are dynamic you could use slider variables in both cases above as well.

`Sequence(<End Value >)`

Creates a list of integers from 1 to the given end value.

Examples:

- `Sequence(4)` creates the list $\{1, 2, 3, 4\}$.
- `2^Sequence(4)` creates the list $\{2, 4, 8, 16\}$.

`Sequence(<Start value k >, <End value n >)`

Creates a list of integers from *k* to *n* (increasing or decreasing).

Examples:

- `Sequence(7, 13)` creates the list $\{7, 8, 9, 10, 11, 12, 13\}$
- `Sequence(18, 14)` creates the list $\{18, 17, 16, 15, 14\}$

- `Sequence (-5, 5)` creates the list $\{-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5\}$.

Note: This syntax can be further simplified: instead of using e.g. the formal `Sequence (7, 13)` it is possible to obtain the same result by typing in the input bar `7..13`.

Note: See Lists for more information on list operations.

Sort Command

Sort(<List>)

Sorts a list of numbers, text objects, or points.

Note: Lists of points are sorted by *x*-coordinates.

Examples:

- `Sort ({3, 2, 1})` gives you the list $\{1, 2, 3\}$.
- `Sort ({"pears", "apples", "figs"})` gives you the list elements in alphabetical order.
- `Sort ({(3, 2), (2, 5), (4, 1)})` gives you $\{(2, 5), (3, 2), (4, 1)\}$.

Sort(<Values>, <Keys>)

Sorts the first list *Values* according to the corresponding second list *Keys*.

Examples:

- In order to sort a list of polynomials `list1 = {x^3, x^2, x^6}` according to degree, create the dependent list of degrees `list2 = Zip(Degree(a), a, list1)`. After that, `Sort(list1, list2)` yields the requested $list3 = \{x^2, x^3, x^6\}$.
- In order to draw the polygon having as vertices the complex roots of , sorted by their arguments, create `list1 = {ComplexRoot(x^10-1)}`, then use the command `Polygon(Sort(list1, arg(list1)))`. This command yields $poly1 = 2.94$.

Note: There is a workaround to sort lists of arbitrary objects which is explained in the Tutorial:Advanced List Sorting.

Take Command

Take(<List>, <Start Position>)

Returns a list containing the elements from *Start Position* to the end of the initial list.

Example:

Take({2, 4, 3, 7, 4}, 3) yields {3, 7, 4}.

Take(<Text>, <Start Position>)

Returns a text containing the elements from *Start Position* to the end of the initial text.

Example:

Take("GeoGebra", 3) yields the text *oGebra*.

Take(<List>, <Start Position>, <End Position>)

Returns a list containing the elements from *Start Position* to *End Position* of the initial list.

Example:

Take({2, 4, 3, 7, 4}, 3, 4) yields {3, 7}.

Take(<Text>, <Start Position>, <End Position>)

Returns a text containing the elements from *Start Position* to *End Position* of the initial text.

Example:

Take("GeoGebra", 3, 6) yields the text *oGeb*.

TiedRank Command

TiedRank(<List>)

Returns a list, whose *i*-th element is the rank of *i*-th element of the given list *L* (rank of element is its position in Sort(*L*)). If there are more equal elements in *L* which occupy positions from *k* to *l* in Sort[*L*], the mean of the ranks from *k* to *l* are associated with these elements.

Examples: TiedRank({4, 1, 2, 3, 4, 2}) returns {5.5, 1, 2.5, 4, 5.5, 2.5}. TiedRank({3, 2, 2, 1}) returns {4, 2.5, 2.5, 1}.

Note: Also see OrdinalRank Command

Union Command

Union(<List>, <List>)

Joins the two lists and removes elements that appear multiple times.

Example:

`Union({1, 2, 3, 4, 5}, {3, 2, 1, 7})` yields `{1, 2, 3, 4, 5, 7}`.

Union(<Polygon>, <Polygon>)

Finds the union of the two polygons. Works only for where the polygons are not self-intersecting, and where the union is a single polygon.

Unique Command

Unique(<List>)

Returns list of elements of the given list in ascending order, repetitive elements are included only once. Works for both a list of numbers and a list of text.

Examples:

- `Unique({1, 2, 4, 1, 4})` yields `{1, 2, 4}`.
- `Unique({"a", "b", "Hello", "Hello"})` yields `{"Hello", "a", "b"}`.

Note: See also Frequency command.

CAS Syntax

Unique(<List>)

Returns a list where each element of the given list occurs only once.

Example:

`Unique({1, x, x, 1, a})` yields `{1, x, a}`.

Zip Command

`Zip(<Expression>, <Var1>, <List1>, <Var2>, <List2>, ...)`

Creates list of objects obtained by substitution of variables in the expression by elements of corresponding lists. If the number of variables matches the number of lists, each variable is taken from the following list. If the number of variables exceeds number of lists by one, the last variable takes values from 1, 2, 3, ..., k where k is the length of the shortest list. Length of the resulting list is minimum of lengths of input lists.

Example: Let P, Q, R, S be some points. `Zip(Midpoint(A, B), A, {P, Q}, B, {R, S})` returns a list containing midpoints of segments PR and QS .

Example: Let $list1=\{x^2, x^3, x^6\}$ be a list of polynomials. `Zip(Degree(a), a, list1)` returns the list $\{2, 3, 6\}$.

Example: Let $list1=\{1, 2, 5\}$ be a list of numbers. `Zip(Simplify(a*x^(b-1)), a, list1, b)` returns the list $\{1, 2x, 5x^2\}$.

Note: In each list the elements must be of the same type.

New possibility > 5.0.258 , functions allowed as variables

Example: `Zip(f(2), f, {x+1, x+3})` returns the list $\{3, 5\}$.

Vector & Matrix Commands

<links/>

ApplyMatrix Command

`ApplyMatrix(<Matrix>, <Object>)`

Transforms the object O so that point P of O is mapped to:

- point $M*P$, if P is a 2D point and M is a 2×2 matrix

Example: Let $M = \{\{\cos(\pi/2), -\sin(\pi/2)\}, \{\sin(\pi/2), \cos(\pi/2)\}\}$ be the transformation matrix and $u = (2, 1)$ a given vector (object). `ApplyMatrix(M, u)` yields the vector $u' = (-1, 2)$, i.e. the result of a mathematically positive rotation by 90° of vector u .

- point $project(M*(x(P), y(P), 1))$, if P is a 2D point and M a 3×3 matrix: *project* is a projection, mapping point (x, y, z) to $(x/z, y/z)$.

Example: Let $M = \{\{1, 1, 0\}, \{0, 1, 1\}, \{1, 0, 1\}\}$ be a matrix and $u = (2, 1)$ a given vector. `ApplyMatrix(M, u)` yields vector $u' = (1, 0.67)$. In effect $=$, and $(3/3 = 1, 2/3 \approx 0.67)$ (rounding to 2 decimal places)

- point $M*P$, if P is a 3D point and M a 3×3 matrix
- point $N*P$, if P is a 3D point and M a 2×2 matrix: the matrix N is the *completion or order 3* of M : given $M =$ then $N =$

Note: This command also works for images.

Determinant Command

`Determinant(<Matrix>)`

Gives the determinant of the matrix.

Example:

`Determinant(\{ \{1, 2\}, \{3, 4\} \})` yields $a = -2$.

CAS Syntax

`Determinant(<Matrix>)`

Gives the determinant of the matrix. If the matrix contains undefined variables, it yields a formula for the determinant.

Example:

`Determinant(\{ \{1, a\}, \{b, 4\} \})` yields $-a b + 4$.

Identity Command

`Identity(<Number>)`

Gives the identity matrix of the given order.

Example:

`Identity(3)` yields the matrix .

Note: If A is a square matrix of order n , A^0 yields the same as `Identity(n)`.

Invert Command

`Invert(<Matrix>)`

Inverts the given matrix.

Example: `Invert({{1, 2}, {3, 4}})` yields , the inverse matrix of .

Note: In the CAS View undefined variables are allowed too.

Example: `Invert({{a, b}, {c, d}})` yields $\begin{pmatrix} \frac{d}{ad - bc} & \frac{-b}{ad - bc} \\ \frac{-c}{ad - bc} & \frac{a}{ad - bc} \end{pmatrix}$, the inverse matrix of $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$.

`Invert(<Function>)`

Gives the inverse of the function.

Example:

`Invert(sin(x))` yields $\text{asin}(x)$.

Note:

The function must contain just one x and no account is taken of domain or range, for example for $f(x) = x^2$ or $f(x) = \sin(x)$.

If there is more than one x in the function another command might help you:

Example:

Both `Invert(PartialFractions((x + 1) / (x + 2)))` and `Invert(CompleteSquare(x^2 + 2 x + 1))` yield the inverse functions.

Note:

- In the CAS View, the command also works if the function contains more than one x .
- See also Eigenvalues Command, Eigenvectors Command, SVD Command, Transpose Command, JordanDiagonalization Command

PerpendicularVector Command

PerpendicularVector(<Line>)

Returns the perpendicular vector of the line.

Example:

Let $\text{Line}((1, 4), (5, -3))$ be the line j . $\text{PerpendicularVector}(j)$ yields the perpendicular vector $u=(7, 4)$ of the line j .

Note: A line with equation $ax + by = c$ has the perpendicular vector (a, b) .

PerpendicularVector(<Segment>)

Returns the perpendicular vector of the segment with the same length.

Example:

Let $\text{Segment}((3, 2), (14, 5))$ be the segment k . $\text{PerpendicularVector}(k)$ yields the perpendicular vector $u=(-3, 11)$ of the segment k .

PerpendicularVector(<Vector>)

Returns the perpendicular vector of the given vector.

Example:

Let $\text{Vector}((-12, 8))$ be the vector u . $\text{PerpendicularVector}(u)$ yields the perpendicular vector $v=(-8, -12)$ of the vector u .

Note: In the CAS View undefined variables are allowed as well.

Example: $\text{PerpendicularVector}((a, b))$ yields the vector $\{-b, a\}$.

PerpendicularVector(<Plane>)

Creates a vector orthogonal to the plane.

Example:

$\text{PerpendicularVector}(\text{xOyPlane})$ yields the perpendicular vector $u=(0, 0, 1)$ of the xOy plane.

Note:

See also UnitPerpendicularVector Command.

ReducedRowEchelonForm Command

ReducedRowEchelonForm(<Matrix>)

Returns the reduced echelon form of the matrix.

Examples:

- ReducedRowEchelonForm({{1, 6, 4}, {2, 8, 9}, {4, 5, 6}}) yields the matrix .
- ReducedRowEchelonForm({{2, 10, 11, 4}, {2, (-5), (-6), 12}, {2, 5, 3, 2}}) yields the matrix .

CAS Syntax

ReducedRowEchelonForm(<Matrix>)

Returns the reduced echelon form of the matrix.

Examples:

- ReducedRowEchelonForm({{1, 6, 4}, {2, 8, 9}, {4, 5, 6}}) yields the matrix .
- ReducedRowEchelonForm({{2, 10, 11, 4}, {2, (-5), (-6), 12}, {2, 5, 3, 2}}) yields the matrix .

Transpose Command

Transpose(<Matrix>)

Transposes the matrix.

Example:

Transpose({{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}) yields the matrix .

CAS Syntax

Transpose(<Matrix>)

Transposes the matrix.

Example:

Transpose({{a, b}, {c, d}}) yields the matrix .

Note:

- See also Eigenvalues Command, Eigenvectors Command, SVD Command, Invert Command, JordanDiagonalization Command

UnitPerpendicularVector Command

UnitPerpendicularVector(<Line>)

Returns the perpendicular vector with length 1 of the given line.

Example:

`UnitPerpendicularVector(3x + 4y = 5)` yields .

UnitPerpendicularVector(<Segment>)

Returns the perpendicular vector with length 1 of the given segment.

Example:

Let `s = Segment((1,1), (4,5))`.

`UnitPerpendicularVector(s)` yields .

UnitPerpendicularVector(<Vector>)

Returns the perpendicular vector with length 1 of the given vector. The vector must be defined first.

Example:

Let `v=UnitPerpendicularVector(v)` yields .

Note: In the CAS View vectors with undefined variables are also valid input.

Example:`UnitPerpendicularVector((a, b))` yields $(\frac{-b}{\sqrt{a^2 + b^2}}, \frac{a}{\sqrt{a^2 + b^2}})$.

CAS Syntax

UnitPerpendicularVector(<Plane>)

Creates a unit vector orthogonal to the plane.

Note:

See also PerpendicularVector Command.

UnitVector Command

UnitVector(<Vector>)

Yields a vector with length 1, which has the same direction and orientation as the given vector. The vector must be defined first.

Example:

Let $v = \text{UnitVector}(v)$ yields .

UnitVector(<Line>)

Yields the direction vector of the given line with length 1.

Example:

$\text{UnitVector}(3x + 4y = 5)$ yields .

UnitVector(<Segment>)

Yields the direction vector of the given segment with length 1.

Example:

Let $s = \text{Segment}((1, 1), (4, 5))$.

$\text{UnitVector}(s)$ yields .

Hint: In the CAS View three-dimensional vectors and vectors with undefined variables are also valid inputs. Examples: $\text{UnitVector}((a, b))$ yields $(\frac{a}{\sqrt{a^2 + b^2}}, \frac{b}{\sqrt{a^2 + b^2}})$. $\text{UnitVector}((2, 4, 4))$ yields $(\frac{1}{3}, \frac{2}{3}, \frac{2}{3})$.

Vector Command

Vector(<Point>)

Returns the position vector of the given point.

Example:

$\text{Vector}((3, 2))$ yields $u =$.

Vector(<Start Point>, <End Point>)

Creates a vector from *Start Point* to *End Point*.

Example:

$\text{Vector}((1, 1), (3, 4))$ yields $u =$.

Note: See also Vector tool.

Transformation Commands

- Dilate (Enlarge)
- Reflect
- Rotate
- Shear
- Stretch
- Translate

See also Transformation tools

AttachCopyToView Command

AttachCopyToView(<Object>, <View 0|1|2>)

If *View* = 0, a copy of given object is created. For *View* = 1 or *View* = 2 this command creates a dependent copy of given object whose size in given Graphics View is constant.

Example:

Let poly = Polygon((0, 0), (1, 0), (1, 1), (0, 1)).

If Graphics View 1 is active, AttachCopyToView(poly, 1) creates a square with the same size at the same position.

Note: Once the copy is created, three other arguments are added to the command definition.

AttachCopyToView(<Object>, <View 0|1|2>, <Point 1>, <Point 2>, <Screen Point 1>, <Screen Point 2>)

If *View* = 0, a copy of given object is created. For *View* = 1 or *View* = 2 this command creates a dependent copy of given object whose size in given Graphics View is transformed using the affine transform that maps *Point 1* to a point whose screen coordinates (in pixels) are equal to *Screen Point 1*, and *Point 2* to a point with screen coordinates equal to *Screen Point 2*.

Example:

Let poly = Polygon((0, 0), (1, 0), (1, 1), (0, 1)).

If Graphics View 1 is active, AttachCopyToView(poly, 1, (0, 0), (1, 1), (0, 0), (100, 100)) creates a 100px x 100px square in the top left corner of the Graphics View.

Note: All points of the object are copied, even if they lie outside the view.

Dilate Command

`Dilate(<Object>, <Dilation Factor>)`

Dilates the object from a point of origin using the given factor.

`Dilate(<Object>, <Dilation Factor>, <Dilation Center Point>)`

Dilates the object from a point, which is the dilation center point, using the given factor.

Note: When dilating polygons, GeoGebra creates also all the transformed vertices and segments.

Note: See also Dilate from Point by Factor tool.

Reflect Command

`Reflect(<Object>, <Point>)`

Reflects the geometric object through a given point.

Note: When reflecting polygons through a point, the transformed vertices and segments are created as well.

`Reflect(<Object>, <Line>)`

Reflects an object (e.g. an image) across a given line.

Note: When reflecting polygons across a line, the transformed vertices and segments are created as well.

`Reflect(<Object>, <Circle>)`

Inverts the geometric object with respect to a circle.

`Reflect(<Object>, <Plane>)`

Reflects an object about a plane.

Note: See also Reflect about Point, Reflect about Line, Reflect about Plane, and Reflect about Circle tools.

Rotate Command

Rotate(<Object>, <Angle>)

Rotates the geometric object by the angle around the axis origin.

Rotate(<Object>, <Angle>, <Point>)

Rotates the geometric object by the angle around the given point.

Rotate(<Object>, <Angle>, <Axis of Rotation>)

Rotates the geometric object by the angle around the given axis of rotation.

Rotate(<Object>, <Angle>, <Point on Axis>, <Axis Direction or Plane>)

Note: Vectors are not rotated around axis origin, but around their initial point. When a polygon, segment, arc, etc. is rotated, also images of the vertices / endpoints and sides (in case of polygon) are created. This command also rotates images. For text rotation use RotateText Command. See also Rotate around Point ToolRotate around Point and Rotate around Line ToolRotate around Line tools.

Shear Command

Shear(<Object>, <Line>, <Ratio>)

Shears the object so that

- points on the line stay fixed.
- points at distance d from the line are shifted by $d \text{ ratio}$ in direction of the line (direction of the shift is different for halfplanes with respect to the line).

A sheared plane figure maintains its original area.

Stretch Command

Stretch(<Object>, <Vector>)

The object is stretched **parallel** to the given vector by the ratio given by the **magnitude** of the vector (i.e. points on the line perpendicular to the vector (through its startpoint) stay on their place and distance of other points from the line is multiplied by given ratio.)

Stretch(<Object>, <Line>, <Ratio>)

The object is stretched **perpendicular** to the line by the given ratio (i.e. points on the line aren't moved and the distance of other points from the line is multiplied by given ratio.)

Translate Command

Translate(<Object>, <Vector>)

Translates the geometric object by the vector.

Note: When translating a polygon, the transformed new vertices and segments are created as well.

Translate(<Vector>, <Start Point>)

Translates the vector to the start point.

Note: See also Translate by Vector tool.

Chart Commands

<links/>

BarChart Command

BarChart(<List of Data>, <List of Frequencies>)

Creates a bar chart using the list of data with corresponding frequencies.

Note: The numbers in the list of raw data need to be arranged in increasing order.

Example:

- `BarChart({10, 11, 12, 13, 14}, {5, 8, 12, 0, 1})`
- `BarChart({5, 6, 7, 8, 9}, {1, 0, 12, 43, 3})`
- `BarChart({0.3, 0.4, 0.5, 0.6}, {12, 33, 13, 4})`

BarChart(<List of Raw Data>, <Width of Bars>, <Vertical Scale Factor (optional)>)

Creates a bar chart using the given raw data; the bars have the given width and the height of the bars depends on the vertical scale factor.

Example: `BarChart({1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 5, 5, 5}, 1)BarChart({1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 5, 5, 5}, 1, 2)`

BarChart(<List of Data>, <List of Frequencies>, <Width of Bars>)

Creates a bar chart using the list of data and corresponding frequencies; the bars have width w .

Example:

- `BarChart({10, 11, 12, 13, 14}, {5, 8, 12, 0, 1}, 0.5)` leaves gaps between bars.
- `BarChart({10, 11, 12, 13, 14}, {5, 8, 12, 0, 1}, 0)` produces a line graph.

BarChart(<Start Value>, <End Value>, <List of Heights>)

Creates a bar chart over the given interval: the number of bars is determined by the length of the list, whose elements are the heights of the bars.

Example: `BarChart(10, 20, {1, 2, 3, 4, 5})` gives you a bar chart with five bars of specified height in the interval [10, 20].

BarChart(<Start Value>, <End Value>, <Expression>, <Variable>, <From Number>, <To Number>)

Creates a bar chart over the given interval (Start Value, End Value), that calculates the bars' heights using the expression whose variable k varies from number c to number d .

Example: If $p = 0.1$, $q = 0.9$, and $n = 10$ are numbers, then `BarChart(-0.5, n + 0.5, BinomialCoefficient(n, k) * p^k * q^(n-k), k, 0, n)` gives you a bar chart in the interval [-0.5, $n+0.5$]. The heights of the bars depend on the probabilities calculated using the given expression.

BarChart(<Start Value>, <End Value>, <Expression>, <Variable>, <From Number>, <To Number>, <Step Width>)

Creates a bar chart over the given interval (Start Value, End Value), the bars' heights are calculated using the given expression in which the variable k varies from number c to number d using step width s .

Note: It is possible to specify a different color/filling for each bar in the Object Properties.

BoxPlot Command

`BoxPlot(yOffset, yScale, List of Raw Data)`

Creates a box plot using the given raw data and whose vertical position in the coordinate system is controlled by variable *yOffset* and whose height is influenced by factor *yScale*.

Example: `BoxPlot(0, 1, {2, 2, 3, 4, 5, 5, 6, 7, 7, 8, 8, 8, 9})`

`BoxPlot(yOffset, yScale, Start Value, Q1, Median, Q3, End Value)`

Creates a box plot for the given statistical data in interval (*Start Value, End Value*).

`BoxPlot(<yOffset>, <yScale>, <List of Raw Data>, <Boolean Outliers>)`

This allows outliers to be plotted as "X"s rather than included in the boxplot. For this command, outliers are data lying below $Q1 - 1.5 * (Q3 - Q1)$ or above $Q3 + 1.5 * (Q3 - Q1)$ (see IQR).

`BoxPlot(<yOffset>, <yScale>, <List of Data>, <List of Frequencies>, <Boolean Outliers>)`

This allows data from a frequency table to be easily plotted as a boxplot.

ContingencyTable Command

`ContingencyTable(<List of Text>, <List of Text>)`

Draws a Contingency Table created from the two given lists. Unique values from the first list are used as row values in the table. Unique values from the second list are used as column values in the table.

`ContingencyTable(<List of Text>, <List of Text>, <Options>)`

Draws a Contingency Table created from the two given lists as described above. The text *Options* controls the display of optional calculations within the table.

Note: Possible values for *Options* are "|", "|_", "+", "e", "k", "=".
"|" = show column percentages
_| = show row percentages
"+" = show total percentages
"e" = show expected counts
"k" = show Chi Squared contributions
"=" = show results of a Chi Squared test

`ContingencyTable(<List of Row Values>, <List of Column Values>, <Frequency Table>)`

Draws a Contingency Table using the given list of row values, column values and corresponding frequency table.

Example: `ContingencyTable({"Males", "Females"}, {"Right-handed", "Left-handed"}, {{43, 9}, {44, 4}})` yields the corresponding Contingency Table.

`ContingencyTable(<List of Row Values>, <List of Column Values>, <Frequency Table>, <Options>)`

Draws a Contingency Table using the given list of row values, column values and corresponding frequency table. The text *Options* controls the display of optional calculations within the table as described above.

Example: `ContingencyTable({"Males", "Females"}, {"Right-handed", "Left-handed"}, {{43, 9}, {44, 4}}, "_")` yields the corresponding Contingency Table showing the row percentages.

DotPlot Command

DotPlot(<List of Raw Data>)

Returns a dot plot for the given list of numbers, as well as the list of the dot plot points. If a number n appears in the list of raw data k times, the returned list contains points $(n, 1), (n, 2), \dots, (n, k)$.

Example:

`DotPlot({2, 5, 3, 4, 3, 5, 3})` yields $\{(2, 1), (3, 1), (3, 2), (3, 3), (4, 1), (5, 1), (5, 2)\}$.

DotPlot(<List of Raw Data>, <Stack Adjacent Dots (optional)>, <Scale Factor (optional)>)

Returns a dot plot for the given list of data, as well as the list of the dot plot points. If a data n appears in the list of raw data k times, the returned list contains points $(n, 1), (n, 2), \dots, (n, k)$.

If you choose a *Scale Factor s*, the returned list contains points $(n, 1s), (n, 2s), \dots, (n, ks)$.

Stack Adjacent Dots means a Boolean Value (true or false): If you choose *true*, points (which are close to each other) are stacked. If you choose *false*, the result will be the same as without *<Stack Adjacent Dots (optional)>*.

The command DotPlot will also work with a list of text.

Example:

`DotPlot({"Red", "Red", "Red", "Blue", "Blue"})` yields $\{(1, 1), (1, 2), (2, 1), (2, 2), (2, 3)\}$.

Note:

If you use a list of text the DotPlot command will put the result in alphabetical order. (e.g. *Blue* appears two times, *Red* three times and B comes before R in the alphabet, so you get $(1, 1), (1, 2)$ for *Blue* and $(2, 1), (2, 2), (2, 3)$ for *Red*.)

FrequencyPolygon Command

Note: Frequency polygon is a line graph drawn by joining all the midpoints of the top of the bars of a histogram. Therefore usage of this command is the same as usage of Histogram Command.

FrequencyPolygon(<List of Class Boundaries>, <List of Heights>)

Creates a frequency polygon with vertices in given heights. The class boundaries determine the x-coordinate of each vertex.

Example: FrequencyPolygon({0, 1, 2, 3, 4, 5}, {2, 6, 8, 3, 1}) creates the corresponding line graph.

FrequencyPolygon(<List of Class Boundaries>, <List of Raw Data>, <Boolean Use Density>, <Density Scale Factor (optional)>)

Creates a frequency polygon using the raw data. The class boundaries determine the x-coordinates of vertices and are used to determine how many data elements lie in each class. The y-coordinate of a vertex is determined as follows

- If *Use Density* = *true*, height = (Density Scale Factor) * (class frequency) / (class width)
- If *Use Density* = *false*, height = class frequency

By default, *Use Density* = *true* and *Density Scale Factor* = 1.

FrequencyPolygon(<Boolean Cumulative>, <List of Class Boundaries>, <List of Raw Data>, <Boolean Use Density> , <Density Scale Factor (optional)>)

If *Cumulative* is true this creates a frequency polygon where each vertex y-coordinate equals the frequency of the class plus the sum of all previous frequencies.

Note: For further examples see Histogram Command.

Histogram Command

`Histogram(<List of Class Boundaries>, <List of Heights>)`

Creates a histogram with bars of the given heights. The class boundaries determine the width and position of each bar of the histogram.

Example: `Histogram({0, 1, 2, 3, 4, 5}, {2, 6, 8, 3, 1})` creates a histogram with 5 bars of the given heights. The first bar is positioned at the interval [0, 1], the second bar is positioned at the interval [1, 2], and so on.

`Histogram(<List of Class Boundaries>, <List of Raw Data>, <Boolean Use Density>, <Density Scale Factor>(optional))`

Creates a histogram using the raw data. The class boundaries determine the width and position of each bar of the histogram and are used to determine how many data elements lie in each class. Bar height is determined as follows

- If *Use Density = true*, height = (Density Scale Factor) * (class frequency) / (class width)
- If *Use Density = false*, height = class frequency

By default, *Use Density = true* and *Density Scale Factor = 1*. This creates a histogram with total area = n, the number of data values.

Note: All elements of Raw Data must be within the interval of the class boundaries, otherwise “undefined” will be returned.

Note: By convention this uses the $a \leq x < b$ rule for each class except for the last class which is $a \leq x \leq b$

Example: (*Default Histogram*)`Histogram({10, 20, 30, 40}, {10, 11, 11, 12, 18, 20, 25, 40}, true)` creates a histogram with 3 bars, with the heights 0.5 (first bar), 0.2 (second bar), and 0.1 (third bar). This histogram has total area = $0.5*10 + 0.2*10 + 0.1*10 = 8$.

Example: (*Count Histogram*)`Histogram({10, 20, 30, 40}, {10, 11, 11, 12, 18, 20, 25, 40}, false)` creates a histogram with 3 bars, with the heights 5 (first bar), 2 (second bar), and 1 (third bar). This histogram does not use density scaling and gives bar heights that equal the count of values in each class.

Example: (*Relative Frequency Histogram*)`Histogram({10, 20, 30, 40}, {10, 11, 11, 12, 18, 20, 25, 40}, true, 10/ 8)` creates a histogram with 3 bars, with the heights 0.625 (first bar), 0.25 (second bar), and 0.125 (third bar). This histogram uses density scaling to give bar heights that equal the proportion of values in each class. If n is the number of data values, and the classes have constant width w, then Density Scale Factor = w/n creates a relative histogram.

Example: (*Normalized Histogram*)`Histogram({10, 20, 30, 40}, {10, 11, 11, 12, 18, 20, 25, 40}, true, 1/8)` creates a histogram with 3 bars, with the heights .0625 (first bar), .025 (second bar), and .0125 (third bar). This histogram has total area = $.0625*10 + .025*10 + .0125*10 = 1$. If n is the number of data values, then Density Scale Factor = 1/n creates a normalized histogram with total area = 1. This is useful for fitting a histogram with a density curve.

`Histogram(<Boolean Cumulative>, <List of Class Boundaries>, <List of Raw Data>, <Boolean Use Density>, <Density Scale Factor> (optional))`

If Cumulative is true this creates a histogram where each bar height equals the frequency of the class plus the sum of all previous frequencies.

Example: `:Histogram(true, {10, 20, 30, 40}, {10, 11, 11, 12, 18, 20, 25, 40}, true)` creates a histogram with 3 bars, with the heights 0.5 (first bar), 0.7 (second bar), and 0.8 (third bar).

HistogramRight Command

HistogramRight(<List of Class Boundaries>, <List of Heights>)

Same as ([Histogram Command]Histogram[<List of Class Boundaries>, <List of Heights>])

HistogramRight(<List of Class Boundaries>, <List of Raw Data>, <Boolean Use Density>, <Density Scale Factor> (optional))

Same as Histogram(<List of Class Boundaries>, <List of Raw Data>, <Boolean Use Density>, <Density Scale Factor>), except that if a datum is equal to the right border of a class, it is counted in this class and not in the next one.

HistogramRight(<Boolean Cumulative>, <List of Class Boundaries>, <List of Raw Data>, <Boolean Use Density>, <Density Scale Factor> (optional))

Same as Histogram(<Boolean Cumulative>, <List of Class Boundaries>, <List of Raw Data>, <Boolean Use Density>, <Density Scale Factor>), except that if a datum is equal to the right border of a class, it is counted in this class and not in the next one.

Note: By convention this uses the $a < x \leq b$ rule for each class except for the first class which is $a \leq x \leq b$

NormalQuantilePlot Command

NormalQuantilePlot(<List of Raw Data>)

Creates a normal quantile plot from the given list of data and draws a line through the points showing the ideal plot for exactly normal data. Points are formed by plotting data values on the x axis against their expected normal score (Z-score) on the y axis.

ResidualPlot Command

ResidualPlot(<List of Points>, <Function>)

Returns a list of points whose x-coordinates are equal to the x-coordinates of the elements of the given list, and y-coordinates are the residuals with respect to f .

If the i -th element of the given list is a point (a, b) then i -th element of the result is $(a, b - f(a))$.

Example:

Let $\text{list} = \{(-1, 1), (-0.51, 2), (0, 0.61), (0.51, -1.41), (0.54, 1.97), (1.11, 0.42), (1.21, 2.53), (-0.8, -0.12)\}$ be the list of points and $f(x) = x^5 + x^4 - x - 1$ the function. The `ResidualPlot(list, f)` command yields $\text{list1} = \{(-1, 1), (-0.51, 2.46), (0, 1.61), (0.51, 0), (0.54, 3.38), (1.11, -0.66), (1.21, 0), (-0.8, 0)\}$ and creates the corresponding points in Graphics View.

StemPlot Command

StemPlot(<List>)

Returns a stem plot of the given list of numbers. Outliers are removed from the plot and listed separately.

An outlier is defined as a value outside the interval $[Q1 - 1.5(Q3 - Q1), Q3 + 1.5(Q3 - Q1)]$.

StemPlot(<List>, <Adjustment -1|0|1>)

Returns a stem plot of the given list of numbers.

If $\text{Adjustment} = -1$ the default stem unit is divided by 10

If $\text{Adjustment} = 0$ nothing is changed

If $\text{Adjustment} = 1$ the default stem unit is multiplied by 10

StepGraph Command

StepGraph(<List of Points>)

Draws a step graph of the given list of points. Each point is connected to the next point in the list by a horizontal line segment.

Example: StepGraph({(1, 1), (3, 2), (4, 5), (5, 7)})

StepGraph(<List of Points>, <Boolean Join>)

Draws a step graph of the given list of points. If *Join* = *false*, then a horizontal line segment is drawn towards the x-coordinate of the next point, but a vertical line segment is not drawn. If *Join* = *true*, then each point is connected to the next point in the list by a horizontal and a vertical line segment.

Example: StepGraph({(1, 1), (3, 2), (4, 5), (5, 7)}, true)

StepGraph(<List of x-coordinates>, <List of y-coordinates>)

Draws a step graph of a list of points created from the given lists of coordinates. Each point is connected to the next point in the list by a horizontal line segment.

Example: StepGraph({1, 3, 4, 5}, {1, 2, 5, 7})

StepGraph(<List of x-coordinates>, <List of y-coordinates>, <Boolean Join>)

Draws a step graph of a list of points created from the given lists of coordinates. If *Join* = *false*, then a horizontal line segment is drawn towards the x-coordinate of the next point, but a vertical line segment is not drawn. If *Join* = *true*, then each point is connected to the next point in the list by a horizontal and a vertical line segment.

Example: StepGraph({1, 3, 4, 5}, {1, 2, 5, 7}, true)

StepGraph(<List of x-coordinates>, <List of y-coordinates>, <Boolean Join>, <Point Style>)

Draws a step graph as described above.

Point style values of -2, -1, 0, 1, -1 determine how points are drawn as follows:

0 = no points are drawn

1 = solid points on the right

2 = solid points on the right, open points on the left

-1 = solid points on the left

-2 = solid points on the left, open points on the right

Example: StepGraph({1, 3, 4, 5}, {1, 2, 5, 7}, false, 1)

StepGraph(<List of Points>, <Boolean Join>, <Point Style>)

Draws a step graph as described above.

Example: StepGraph({(1, 1), (3, 2), (4, 5), (5, 7)}, false, 1)

StickGraph Command

StickGraph(<List of Points>)

Draws a stick graph of the given points. For each point a vertical line segment is drawn from the x-axis to the point.

Example: StickGraph({(1, 1), (3, 2), (4, 5), (5, 7)})

StickGraph(<List of Points>, <Boolean Horizontal>)

Draws a stick graph of the given points. If *Horizontal* = *true*, then horizontal line segments are drawn from the y-axis to each point. If *Horizontal* = *false*, then vertical line segments are drawn from the x-axis to each point.

Example: StickGraph({(1, 1), (3, 2), (4, 5), (5, 7)}, false)

StickGraph(<List of x-coordinates>, <List of y-coordinates>)

Draws a stick graph of points created from the two lists of coordinates. For each point a vertical line segment is drawn from the x-axis to the point.

Example: StickGraph({1, 3, 4, 5}, {1, 2, 5, 7})

StickGraph(<List of x-coordinates>, <List of y-coordinates>, <Boolean Horizontal>)

Draws a stick graph of points created from the two lists of coordinates. If *Horizontal* = *true*, then horizontal line segments are drawn from the y-axis to each point. If *Horizontal* = *false*, then vertical line segments are drawn from the x-axis to each point.

Example: StickGraph({1, 3, 4, 5}, {1, 2, 5, 7}, true)

Statistics Commands

<links/>

See also Probability Calculator.

ANOVA Command

ANOVA(<List>, <List>, ...)

Performs a one-way ANOVA test on the given lists of numbers.

Results are returned in list form as {P value, F test statistic}.

Classes Command

Classes(<List of Data>, <Start>, <Width of Classes>)

Gives a list of class boundaries. The first boundary (min) is equal to *Start*, the last boundary (max) will be at least the maximum of the *List* and the boundaries will be equally spaced between min and max.

Example: Classes({0.1, 0.2, 0.4, 1.1}, 0, 1) gives {0, 1, 2}

Classes(<List of Data>, <Number of Classes>)

Gives a list of class boundaries. The first boundary (min) is equal to the minimum of the *List*, the last boundary (max) will be the maximum of the *List* and the boundaries will be equally spaced between min and max.

Example: Classes({1, 3, 5, 7, 8, 9, 10}, 3) gives {1, 4, 7, 10}

Note: By convention this uses the $a \leq x < b$ rule for each class except for the last class which is $a \leq x \leq b$

CorrelationCoefficient Command

CorrelationCoefficient(<List of x-coordinates>, <List of y-coordinates>)

Calculates the product moment correlation coefficient using the given x- and y-coordinates.

Example: CorrelationCoefficient({1, 3, 2, 1, 5, 2}, {1, 6, 4, 3, 3, 2}) yields 0.36.

CorrelationCoefficient(<List of Points>)

Calculates the product moment correlation coefficient using the coordinates of the given points.

Example: CorrelationCoefficient({(1, 1), (3, 6), (2, 4), (1, 3), (5, 3), (2, 2)}) yields 0.36.

Covariance Command

Covariance(<List of Numbers>, <List of Numbers>)

Calculates the covariance between the elements of the specified lists.

Example: `Covariance({1, 2, 3}, {1, 3, 7})` yields 2, the covariance of {1, 2, 3} and {1, 3, 7}.

Covariance(<List of Points>)

Calculates the covariance between the x and y coordinates of the specified points.

Example: `Covariance({(1, 1), (2, 3), (3, 7)})` yields 2, the covariance of {1, 2, 3} and {1, 3, 7}.

Fit Command

Fit(<List of Points>, <List of Functions>)

Calculates a linear combination of the *functions* that best fit the *points* in the list.

Example:

- `Fit({(-2, 3), (0, 1), (2, 1), (2, 3)}, {x^2, x})` yields $0.625x^2 - 0.25x$.

- Let $L = \{A, B, C, \dots\}$, $f(x) = 1$, $g(x) = x$, $h(x) = e^x$, $F = \{f, g, h\}$.

`Fit(L, F)` calculates a function of the form $a + b x + c e^x$ that fits the points in the list.

Fit(<List of points>, <Function>)

Calculates a minimum squared error function to the points in the list. The *function* must depend on one or more sliders, that are taken as start values of parameters to be optimized. The non-linear iteration might not converge, but adjusting the sliders to a better starting point might help.

Example: Let a be slider with interval from -5 to 5 and increment 1. `Fit({(-2, 3), (0, 1), (2, 1), (2, 3)}, a + x^2)` yields $-I + x^2$.

Note:

- See also FitExp, FitGrowth, FitLine, FitLineX, FitLog, FitLogistic, FitPoly, FitPow and FitSin
- If you work with big/small numbers, you should consider normalizing them for a more accurate result, see Normalize Command.

FitExp Command

FitExp(<List of Points>)

Calculates the exponential regression curve in the form $a \cdot b^x$.

Example: FitExp({ (0, 1), (2, 4) }) yields $1 \cdot 1.69^x$.

Note:

- If you want the answer in the form then use the FitGrowth Command.
- Euler's number e can be obtained by pressing ALT + e.
- See also Fit, FitGrowth, FitLine, FitLineX, FitLog, FitLogistic, FitPoly, FitPow and FitSin.
- If you work with big/small numbers, you should consider normalizing them for a more accurate result, see Normalize Command.

FitGrowth Command

FitGrowth(<List of Points>)

Calculates a function of the form to the points in the list. (Very similar to FitExp[<List of Points>], just in a slightly different form).

Example: FitGrowth({ (0, 1), (2, 3), (4, 3), (6, 4) }) yields $1.31 \cdot 1.23^x$.

Note:

- See also Fit, FitExp, FitLine, FitLineX, FitLog, FitLogistic, FitPoly, FitPow and FitSin
- If you work with big/small numbers, you should consider normalizing them for a more accurate result, see Normalize Command.

FitImplicit Command

FitImplicit(<List of Points>, <Order>)

Attempts to find a best-fit implicit curve of order $n \geq 2$ through the points. You need at least points.

Note: See also the ImplicitCurve, FitExp, FitGrowth, FitLine, FitLineX, FitLog, FitLogistic, FitPoly, FitPow and FitSin commands.

FitLineX Command

FitLineX(<List of Points>)

Calculates the x on y regression line of the points.

Example: FitLineX({(-1, 3), (2, 1), (3, 4), (5, 3), (6, 5)}) yields $1.1x - 0.1$.

CAS Syntax

FitLineX(<List of Points>)

Calculates the x on y regression line of the points.

Example: FitLineX({(-1, 3), (2, 1), (3, 4), (5, 3), (6, 5)}) yields $1.1x - 0.1$.

Note:

- See also Best Fit Line tool and FitLine Command
- If you work with big/small numbers, you should consider normalizing them for a more accurate result, see Normalize Command.

FitLine Command

FitLine(<List of Points>)

Calculates the y on x regression line of the points.

Example: FitLine({ (-2, 1), (1, 2), (2, 4), (4, 3), (5, 4) }) yields $0.4x + 2$.

CAS Syntax

FitLine(<List of Points>)

Calculates the y on x regression line of the points.

Example: FitLine({ (-2, 1), (1, 2), (2, 4), (4, 3), (5, 4) }) yields $0.4x + 2$.

Note:

- See also Best Fit Line tool and FitLineX Command
- If you work with big/small numbers, you should consider normalizing them for a more accurate result, see Normalize Command.

FitLog Command

FitLog(<List of Points>)

Calculates the logarithmic regression curve.

Example: FitLog({ (e, 1), (e^2, 4) }) yields $-2 + 3 \ln(x)$.

CAS Syntax

FitLog(<List of Points>)

Calculates the logarithmic regression curve.

Example: FitLog({ (e, 1), (e^2, 4) }) yields $3 \ln(x) - 2$.

Note:

- Euler's number e can be obtained by pressing ALT + e.
- See also FitExp Command, FitPoly Command, FitPow Command and FitSin Command.
- If you work with big/small numbers, you should consider normalizing them for a more accurate result, see Normalize Command.

FitLogistic Command

FitLogistic(<List of Points>)

Calculates the regression curve in the form $a/(1 + b e^{-kx})$.

Example: FitLogistic({{-6, 2}, {0, 2}, {3, 4}, {3.4, 8}}) yields .

Note:

- The first and last data points should be fairly close to the curve. The list should have at least 3 points, preferably more.
- See also Fit, FitExp, FitGrowth, FitLine, FitLineX, FitLog, FitPoly, FitPow and FitSin.
- If you work with big/small numbers, you should consider normalizing them for a more accurate result, see Normalize Command.

FitPoly Command

FitPoly(<List of Points>, <Degree of Polynomial>)

Calculates the regression polynomial of degree n .

Example: FitPoly({{-1, -1}, {0, 1}, {1, 1}, {2, 5}}, 3) yields $f(x) = x^3 - 1 x^2 + 1$.

FitPoly(<Freehand Function>, <Degree of Polynomial>)

Calculates the regression polynomial of degree n for a function drawn by the Freehand Shape Tool.

CAS Syntax

FitPoly(<List of Points>, <Degree of Polynomial>)

Calculates the regression polynomial of degree n .

Example: FitPoly({{-1, -1}, {0, 1}, {1, 1}, {2, 5}}, 3) yields $x^3 - x^2 + 1$.

Note:

- For order n there must be at least $n + 1$ Points in the list.
- See also FitExp Command, FitLog Command, FitPow Command and FitSin Command.
- If you work with big/small numbers, you should consider normalizing them for a more accurate result, see Normalize Command.

FitPow Command

FitPow(<List of Points>)

Calculates the regression curve in the form $a x^b$.

Example: FitPow({ (1, 1), (3, 2), (7, 4) }) creates the regression curve $f(x) = 0.97 x^{0.71}$.

CAS Syntax

FitPow(<List of Points>)

Calculates the regression curve in the form $a x^b$.

Example: FitPow({ (1, 1), (3, 2), (7, 4) }) yields $0.97 x^{0.71}$.

Note:

- All points used need to be in the first quadrant of the coordinate system.
- See also FitExp Command, FitLog Command, FitPoly Command, and FitSin Command.
- If you work with big/small numbers, you should consider normalizing them for a more accurate result, see Normalize Command.

FitSin Command

FitSin(<List of Points>)

Calculates the regression curve in the form $a + b \sin(c x + d)$.

Example: FitSin({ (1, 1), (2, 2), (3, 1), (4, 0), (5, 1), (6, 2) }) yields $f(x) = 1 + 1 \sin(1.57 x - 1.57)$.

Note:

- The list should have at least four points, preferably more. The list should cover at least two extremal points. The first two local extremal points should not be too different from the absolute extremal points of the curve.
- See also FitExp Command, FitLog Command, FitPoly Command and FitPow Command.
- If you work with big/small numbers, you should consider normalizing them for a more accurate result, see Normalize Command.

Frequency Command

Frequency(<List of Raw Data>)

Returns a list with a count of the occurrences of each unique value in the given list of data. This input list can be numbers or text. The list is sorted in ascending order of the unique values. To get a list of the corresponding unique values use the Unique Command.

Example: Enter `list1 = { "a", "a", "x", "x", "x", "b" }`. `Frequency(list1)` returns the list `{ 2, 1, 3 }`. `Unique(list1)` returns the list `{ "a", "b", "x" }`.

Frequency(<Boolean Cumulative>, <List of Raw Data>)

If *Cumulative* = *false*, returns the same list as Frequency(<List of Raw Data>)

If *Cumulative* = *true*, returns a list of cumulative frequencies for Frequency(<List of Raw Data>).

Example: Enter `list1 = { 0, 0, 0, 1, 1, 2 }`. `Frequency(true, list1)` returns the list `{ 3, 5, 6 }`. `Frequency(false, list1)` returns the list `{ 3, 2, 1 }`. `Unique(list1)` returns the list `{ 0, 1, 2 }`.

Frequency(<List of Class Boundaries>, <List of Raw Data>)

Returns a list of the counts of values from the given data list that lie within intervals of the form $[a, b]$, where a and b are all the couples of consecutive numbers in the given class boundaries list. The highest interval has the form $[a, b]$.

Example: `Frequency({1, 2, 3}, {1, 1, 2, 3})` returns the list `{ 2, 2 }`.

Frequency(<List of Text>, <List of Text>)

Returns a contingency matrix containing counts of paired values from the two lists. The rows of the matrix correspond to the unique values in the first list, and the columns correspond to the unique values in the second list. To get a list of the unique values for each list use the command Unique Command.

Example: Let `list1 = {"a", "b", "b", "c", "c", "c", "c"} and list2 = {"a", "b", "a", "a", "c", "c", "d"}`. Then `Frequency(list1, list2)` returns the matrix

Note: See also the ContingencyTable command.

Frequency(<Boolean Cumulative>, <List of Class Boundaries>, <List of Raw Data>)

If *Cumulative* = *false*, returns the same list as Frequency(<List of Class Boundaries>, <List of Raw Data>)

If *Cumulative* = *true*, returns a list of cumulative frequencies for Frequency(<List of Class Boundaries>, <List of Raw Data>)

Frequency(<List of Class Boundaries>, <List of Raw Data>, <Use Density> , <Density Scale Factor> (optional))

Returns a list of frequencies for the corresponding Histogram Command.

If *Use density* = *false*, returns the same list as Frequency(<List of Class Boundaries>, <List of Raw Data>)

If *Use density* = *true*, returns the list of frequencies of each class.

Example: Let `data = {1, 2, 2, 2, 3, 3, 4, 4, 4}` be the list of raw data and `classes={0, 2, 5}` the list of class boundaries. Then `Frequency(classes, data, false)` and `Frequency(classes, data)` both return the list `{1, 9}`, while `Frequency(classes, data, true)` returns the list `{0.5, 3}`.

Frequency(<Boolean Cumulative>, <List of Class Boundaries>, <List of Raw Data>, <Use Density> , <Density Scale Factor> (optional))

Returns a list of frequencies for the corresponding Histogram Command.

FrequencyTable Command

FrequencyTable(<List of Raw Data>)

Returns a table (as text) whose first column contains sorted list of unique elements of list L and second column contains the count of the occurrences of value in the first column. List L can be numbers or text.

FrequencyTable(<Boolean Cumulative>, <List of Raw Data>)

If $Cumulative = \text{false}$, returns the same table as Frequency(<List of Raw Data>)

If $Cumulative = \text{true}$, returns a table whose first column is the same as in FrequencyTable(<List of Raw Data>) and the second contains cumulative frequencies of values in the first column.

FrequencyTable(<List of Class Boundaries>, <List of Raw Data>)

Returns a table (as text) whose first column contains intervals (classes) and second column contains the count of numbers in *List of Raw Data*, which belong to the interval in the first column. All intervals except the highest interval are of the form $[a, b)$. The highest interval has the form $[a, b]$.

FrequencyTable(<Boolean Cumulative>, <List of Class Boundaries>, <List of Raw Data>)

If $Cumulative = \text{false}$, returns the same table as FrequencyTable(<List of Class Boundaries>, <List of Raw Data>)

If $Cumulative = \text{true}$, returns a table whose first column is the same as in FrequencyTable(<List of Raw Data>) and the second contains cumulative frequencies of values in the first column.

FrequencyTable(<List of Class Boundaries>, <List of Raw Data>, <Use Density> , <Density Scale Factor (optional)>)

Returns a table (as text) whose first column contains intervals (classes) and second contains frequencies for the corresponding Histogram Command.

FrequencyTable(<Boolean Cumulative>, <List of Class Boundaries>, <List of Raw Data>, <Use Density> , <Density Scale Factor (optional)>)

Returns a table (as text) whose first column contains intervals (classes) and second contains frequencies for the corresponding Histogram Command.

FrequencyTable(<List of Raw Data>, <Scale Factor (optional)>)

Returns a table (as text) whose first column **Value** contains a sorted list of unique elements of the <*List of Raw Data*> and second column **Frequency** contains the count of the occurrences of value in the first column multiplied by the <*Scale Factor*>. The list can be numbers or text.

Example:

FrequencyTable({1, 1, 1, 2, 2, 3, 3, 4, 5}, 2) returns a table with first column *Value* with entries 1, 2, 3, 4, 5 and second column *Frequency* with entries 6, 4, 4, 2, 2.

Note: In the list there appears 1 three-times, so the count of the occurrences of 1 (=3) has to be multiplied by the scale factor 2 to get entry 6 in the second column.

Example:

FrequencyTable({"red", "red", "green", "green", "blue"}, 5) returns a table with first column *Value* with entries blue, green, red (alphabetical order) and second column *Frequency* with entries 5, 10, 10.

Note: This command is similar to Frequency Command and Histogram Command. Articles about these commands contain some related examples.

GeometricMean Command

GeometricMean(List of Numbers)

Returns the geometric mean of given list of numbers.

Example: GeometricMean({13, 7, 26, 5, 19}) yields 11.76.

HarmonicMean Command

HarmonicMean(<List of Numbers>)

Returns the harmonic mean of given list of numbers.

Example: HarmonicMean({13, 7, 26, 5, 19}) yields 9.79.

Mean Command

Mean(<List of Raw Data>)

Calculates the arithmetic mean of list elements.

Example:

- Mean({1, 2, 3, 2, 4, 1, 3, 2}) yields $a = 2.25$ and
- Mean({1, 3, 5, 9, 13}) yields $a = 6.2$.

Mean(<List of Numbers>, <List of Frequencies>)

Calculates the weighted mean of the list elements.

Example:

- Mean({1, 2, 3, 4}, {6, 1, 3, 6}) yields $a = 2.56$ and
- Mean({1, 2, 3, 4}, {1, 1, 3, 6}) yields $a = 3.27$.

Note:

See also MeanX, MeanY, and SD commands.

MeanX Command

MeanX(<List of Points>)

Calculates the mean of the x -coordinates of the points in the list.

Example: MeanX({ (0, 0), (3, 2), (5, 1), (2, 1), (2, 4) }) yields 2.4

MeanY Command

MeanY(<List of Points>)

Calculates the mean of the y -coordinates of the points in the list.

Example: MeanY({ (0, 0), (3, 2), (5, 1), (2, 1), (2, 4) }) yields 1.6

Median Command

Median(<List of Raw Data>)

Determines the median of the list elements.

Examples:

- Median({1, 2, 3}) yields 2.
- Median({1, 1, 8, 8}) yields 4.5.

Median(<List of Numbers>, <List of Frequencies>)

Calculates the weighted median of the list elements.

Example:

- Median({1, 2, 3}, {4, 1, 3}) yields 1.5.
- Median({1, 2, 3, 4}, {6, 1, 3, 6}) yields 3.

Note:

- If the length of the given list is even, the arithmetic mean of the two center elements is returned.
- See also Mean command.

Mode Command

Mode(<List of Numbers>)

Determines the mode(s) of the list elements.

Examples: Mode({1, 2, 3, 4}) returns an empty list {}. Mode({1, 1, 1, 2, 3, 4}) returns the list {1}. Mode({1, 1, 2, 2, 3, 3, 4}) returns the list {1, 2, 3}.

Normalize Command

Normalize(<List of Numbers>)

Returns a list containing the *normalized* form of the given numbers.

Example: Normalize({1, 2, 3, 4, 5}) returns {0, 0.25, 0.5, 0.75, 1}.

Normalize(<List of Points>)

Returns a list containing the *normalized* form of the given points.

Example: Normalize({(1, 5), (2, 4), (3, 3), (4, 2), (5, 1)}) returns {(0, 1), (0.25, 0.75), (0.5, 0.5), (0.75, 0.25), (1, 0)}.

Notes: If you are doing calculations using big or small numbers (eg using FitGrowth CommandFitGrowth) then normalizing them might avoid rounding/overflow errors. This command is not applicable to 3D points. The operation of normalization maps a value x to the interval $[0, 1]$ using the linear function $x \mapsto \frac{x - \text{Min}[list]}{\text{Max}[list] - \text{Min}[list]}$.

Percentile Command

Percentile(<List of Numbers>, <Percent>)

Let P equal the given *Percent*.

Returns the value that cuts off the first P percent of the *list of numbers*, when the list is sorted in ascending order. *Percent* must be a number in the interval $0 < P \leq 1$.

Example: Percentile({1, 2, 3, 4}, 0.25) yields 1.25.

Note: The commands Quartile and Percentile use different rules and do not always return matching results. Example: Q1({1, 2, 3, 4}) yields 1.5 whereas Percentile({1, 2, 3, 4}, 0.25) yields 1.25.

Q1 Command

Quartile1(<List of Raw Data>)

Determines the lower quartile of the list elements.

Example: Quartile1({1, 2, 3, 4}) yields 1.5.

Quartile1(<List of Numbers>, <List of Frequencies>)

Determines the lower quartile of the list elements considering the frequencies.

Example: Quartile1({1, 2, 3, 4}, {3, 2, 4, 2}) yields 1.

Note: GeoGebra uses the **Moore & McCabe (2002)** method to calculate quartiles, see <http://mathworld.wolfram.com/Quartile.html>

Q3 Command

Quartile3(<List of Raw Data>)

Determines the upper quartile of the list elements.

Example: Quartile3({1, 2, 3, 4}) yields 3.5.

Quartile3(<List of Numbers>, <List of Frequencies>)

Determines the upper quartile of the list elements considering the frequencies.

Example: Quartile3({1, 2, 3, 4}, {3, 2, 4, 2}) yields 3.

Note: GeoGebra uses the **Moore & McCabe (2002)** method to calculate quartiles, see <http://mathworld.wolfram.com/Quartile.html>

RSquare Command

RSquare(<List of Points>, <Function>)

Calculates the coefficient of determination $R^2 = 1 - SSE / Syy$, between the y -values of the points in the list and the function values of the x -values in the list.

Example:

`RSquare({{-3, 2}, {-2, 1}, {-1, 3}, {0, 4}, {1, 2}, {2, 4}, {3, 3}, {4, 5}, {6, 4}}, 0.5x + 2.5)` yields 0.28.

RootMeanSquare Command

RootMeanSquare(<List of Numbers>)

Returns the root mean square of given list of numbers.

Example: `RootMeanSquare({3, 4, 5, 3, 2, 3, 4})` yields 3.5456.

SD Command

SD(<List of Numbers>)

Calculates the standard deviation of the numbers in the list.

Example: `SD({1, 2, 3, 4, 5})` yields 1.41

SD(<List of Numbers>, <List of Frequencies>)

Calculates the weighted *standard deviation* of the given numbers.

Example: `SD({20, 40, 41, 42, 40, 54}, {20, 6, 4, 5, 2, 1})` yields 10.98

CAS Syntax

SD(<List of Numbers>)

Calculates the *standard deviation* of the numbers in the list.

Example:

- `SD({1, 2, 3, 4, 5})` yields .
- `SD({-3 + 2 x, -1 - 4 x, -2 + 5 x^2})` is evaluated as .

Note: See also Mean Command.

SDX Command

SDX(<List of Points>)

Returns standard deviation of x-coordinates of points from the given list.

Example: SDX({(1, 1), (2, 2), (3, 1), (3, 3), (4, 2), (3, -1)}) yields $a = 0.94$.

SDY Command

SDY(<List of Points>)

Returns standard deviation of y-coordinates of points from the given list.

Example: SDY({(1, 1), (2, 2), (3, 1), (3, 3), (4, 2), (3, -1)}) yields $a = 1.25$.

Sxx Command

Sxx(<List of Numbers>)

Calculates the statistic .

Sxx(<List of Points>)

Calculates the statistic using the x -coordinates of the given points.

Sxy Command

Sxy(<List of Points>)

Calculates the statistic using the coordinates of the given points.

Sxy(<List of Numbers>, <List of Numbers>)

Calculates the statistic , where x are the values in the first list, and y are the values in the second given list.

Syy Command

Syy(<List of Points>)

Calculates the statistic using the *y*-coordinates of the given points.

Sample Command

Sample(<List>, <Size>)

Returns list of *n* randomly chosen elements of a list; elements can be chosen several times.

Example: `Sample({1, 2, 3, 4, 5}, 5)` yields for example *list1* = {1, 2, 1, 5, 4}.

Sample(<List>, <Size>, <With Replacement>)

Returns list of *n* randomly chosen elements of a list. Elements can be chosen several times if and only if the last parameter is true.

Example: `Sample({1, 2, 3, 4, 5}, 5, true)` yields for example *list1* = {2, 3, 3, 4, 5}.

Hint: In the CAS View the input list can contain different types of objects: Examples: `Sample({-5, 2, a, 7, c}, 3)` yields for example {a, 7, -5}. The list can include lists as well: Let List1 be {1, 2, 3}; `Sample({List1, 4, 5, 6, 7, 8}, 3, false)` yields for example {6, {1, 2, 3}, 4}.

SampleSD Command

SampleSD(<List of Numbers>)

Returns sample standard deviation of given list of numbers.

Example: `SampleSD({1, 2, 3})` yields 1.

SampleSD(<List of Numbers>, <List of Frequencies>)

Returns the *standard deviation* of the sample of numbers having the given frequencies.

Example: `SampleSD({1, 2, 3, 4}, {1, 1, 1, 2})` yields 1.08.

Hint: If the list contains undefined variables in the CAS View, the command yields a formula for the *sample standard deviation*. Example: `SampleSD({1, 2, a})` yields $\frac{\sqrt{a^2 - 3a + 3}}{\sqrt{3}}$.

SampleSDX Command

SampleSDX(<List of Points>)

Returns sample standard deviation of x-coordinates of points from the given list.

Example: SampleSDX({(2, 3), (1, 5), (3, 6), (4, 2), (1, 1), (2, 5)}) yields $a = 1.17$.

SampleSDY Command

SampleSDY(<List of Points>)

Returns sample standard deviation of y-coordinates of points from the given list.

Example: SampleSDY({(2, 3), (1, 5), (3, 6), (4, 2), (1, 1), (2, 5)}) yields $a = 1.97$.

SampleVariance Command

SampleVariance(<List of Raw Data>)

Returns the sample variance of given list of numbers.

Example: SampleVariance({1, 2, 3, 4, 5}) yields $a = 2.5$.

SampleVariance(<List of Numbers>, <List of Frequencies>)

Returns the sample variance of given list of numbers considering the frequencies.

Example: SampleVariance({1, 2, 3, 4, 5}, {3, 2, 4, 4, 1}) yields $a = 1.67$.

Hint: If the list in the CAS View contains undefined variables, this command yields a formula for the sample variance. Example: SampleVariance({a, b, c}) yields $\frac{1}{3} a^2 - \frac{1}{3} ab - \frac{1}{3} ac + \frac{1}{3} b^2 - \frac{1}{3} bc + \frac{1}{3} c^2$.

Shuffle Command

Shuffle(<List>)

Returns list with same elements, but in random order.

Note: You can recompute the list via *Recompute all objects* in View Menu (or pressing F9).

See also RandomElement Command and RandomBetween Command.

CAS Syntax

Shuffle(<List>)

Returns list with same elements, but in random order.

Examples: Shuffle({3, 5, 1, 7, 3}) yields for example {5, 1, 3, 3, 7}. Shuffle(Sequence(20)) gives the first 20 whole numbers in a random order.

SigmaXX Command

SigmaXX(<List of Points>)

Calculates the sum of squares of the x -coordinates of the given points.

Example: Let $\text{list1} = \{(-3, 4), (-1, 4), (-2, 3), (1, 3), (2, 2), (1, 5)\}$ be a list of points. SigmaXX(list1) yields the value 20.

SigmaXX(<List of Raw Data>)

Calculates the sum of squares of the given numbers.

Example: In order to work out the variance of a list you may use SigmaXX(list) / Length(list) – Mean(list) 2 .

SigmaXX(<List of Numbers>, <List of Frequencies>)

Calculates the weighted sum of squares of the given numbers.

Example: Let $\text{list1} = \{3, 2, 4, 3, 3, 2, 1, 1, 2, 3, 3, 4, 5, 3, 2, 1, 1, 2, 3\}$ be a list of numbers. Unique(list1) yields $\text{list2} = \{1, 2, 3, 4, 5\}$ and Frequency(list1) yields $\text{list3} = \{4, 5, 7, 2, 1\}$. Command SigmaXX(list2, list3) yields the value 144.

SigmaXY Command

`SigmaXY(<List of Points>)`

Calculates the sum of the products of the x - and y -coordinates.

Example: You can work out the covariance of a list of points using `SigmaXY(list) / Length(list) - MeanX(list) * MeanY(list)`.

`SigmaXY(<List of x-coordinates>, <List of y-coordinates>)`

Calculates the sum of the products of the x - and y -coordinates.

Example: Let $A = (-3, 4)$, $B = (-1, 4)$, $C = (-2, 3)$ and $D = (1, 3)$ be points. `{x(A), x(B), x(C), x(D)}` yields the x -coordinates of the points in a list $list1 = \{-3, -1, -2, 1\}$ and `{y(A), y(B), y(C), y(D)}` yields the y -coordinates of the points in a list $list2 = \{4, 4, 3, 3\}$. Command `SigmaXY(list1, list2)` yields $a = -19$.

SigmaYY Command

`SigmaYY(<List of Points>)`

Calculates the sum of squares of y -coordinates of the given points.

Example: Let $list = \{(-3, 4), (-1, 4), (-2, 3), (1, 3), (2, 2), (1, 5)\}$ be a list of points. `SigmaYY(list)` yields $a = 79$.

Spearman Command

`Spearman(<List of Points>)`

Returns Spearman's rank correlation coefficient of x -coordinates and y -coordinates of points of a list.

Example: Let $list = \{(-3, 4), (-1, 4), (-2, 3), (1, 3), (2, 2), (1, 5)\}$ be a list of points. `Spearman(list)` yields $a = -0.37$.

`Spearman(<List of Numbers>, <List of Numbers>)`

Returns Spearman's rank correlation coefficient of two lists.

Example: Let $list1 = \{3, 2, 4, 5, 1, 6, 8, 9\}$ and $list2 = \{5, 6, 8, 2, 1, 3, 4, 7\}$ be two lists. `Spearman(list1, list2)` yields $a = 0.24$.

Sum Command

Sum(<List>)

Calculates the sum of all list elements.

Examples:

- Sum({1, 2, 3}) yields the number $a = 6$.
- Sum({x^2, x^3}) yields $f(x) = x^2 + x^3$.
- Sum(Sequence(i, i, 1, 100)) yields the number $a = 5050$.
- Sum({(1, 2), (2, 3)}) yields the point $A = (3, 5)$.
- Sum({"a", "b", "c"}) yields the text "abc".

Sum(<List>, <Number of Elements>)

Calculates the sum of the first n list elements.

Example:

Sum({1, 2, 3, 4, 5, 6}, 4) yields the number $a = 10$.

Sum(<List>, <List of Frequencies>)

Returns the sum of given list of numbers considering the frequencies.

Example:

Sum({1, 2, 3, 4, 5}, {3, 2, 4, 4, 1}) yields $a = 40$.

Note: This command works for numbers, points, vectors, text, and functions.

CAS Syntax

The following command only works in the CAS View.

Sum(<Expression>, <Variable>, <Start Value>, <End Value>)

Computes sum . End value might be infinity.

Examples:

- Sum(n^2, n, 1, 3) yields 14.
- Sum(r^k, k, 0, n) yields .
- Sum((1/3)^n, n, 0, Infinity) yields .

SumSquaredErrors Command

SumSquaredErrors(<List of Points>, <Function>)

Calculates the sum of squared errors, SSE, between the y-values of the points in the list and the function values of the x-values in the list.

Example: If we have a list of points $L=\{(1, 2), (3, 5), (2, 2), (5, 2), (5, 5)\}$ and have calculated for example: $f(x)=\text{FitPoly}(L, 1)$ and $g(x)=\text{FitPoly}(L, 2)$. $\text{SumSquaredErrors}(L, f)$ yields 9 and $\text{SumSquaredErrors}(L, g)$ yields 6.99, and therefore we can see, that $g(x)$ offers the best fit, in the sense of the least sum of squared errors (Gauss).

TMean2Estimate Command

TMean2Estimate(<List of Sample Data 1>, <List of Sample Data 2>, <Level>, <Boolean Pooled>)

Calculates a t confidence interval estimate of the difference between two population means using the given sample data sets and confidence *Level*.

If *Pooled* = true, then population variances are assumed equal and sample standard deviations are combined in calculation.

If *Pooled* = false, then population variances are not assumed equal and sample standard deviations are not combined.

Results are returned in list form as *{lower confidence limit, upper confidence limit}*.

TMean2Estimate(<Sample Mean 1>, <Sample Standard Deviation 1>, <Sample Size 1>, <Sample Mean 2>, <Sample Standard Deviation 2>, <Sample Size 2>, <Level>, <Boolean Pooled>)

Calculates a t confidence interval estimate of the difference between two population means using the given sample statistics and confidence *Level*. *Pooled* is defined as above. Results are returned in list form as *{lower confidence limit, upper confidence limit}*.

TMeanEstimate Command

TMeanEstimate(<List of Sample Data>, <Level>)

Calculates a t confidence interval estimate of a population mean using the given sample data and confidence level. Results are returned in list form as *{lower confidence limit, upper confidence limit}*.

TMeanEstimate(<Sample Mean>, <Sample Standard Deviation>, <Sample Size>, <Level>)

Calculates a t confidence interval estimate of a population mean using the given sample statistics and confidence level. Results are returned in list form as *{lower confidence limit, upper confidence limit}*.

TTest Command

TTest(<List of Sample Data>, <Hypothesized Mean>, <Tail>)

Performs a one-sample t-test of a population mean using the given list of sample data. *Hypothesized Mean* is the population mean assumed in the null hypothesis. *Tail* has possible values "<", ">" , "≠". These specify the alternative hypothesis as follows.

"<" = population mean < *Hypothesized Mean*

">" = population mean > *Hypothesized Mean*

"≠" = population mean ≠ *Hypothesized Mean*

Results are returned in list form as *{Probability value, t-test statistic}*.

Example: TTest ({1, 2, 3, 4, 5}, 3, "<") yields {0.5, 0}.

TTest(<Sample Mean>, <Sample Standard Deviation>, <Sample Size>, <Hypothesized Mean>, <Tail>)

Performs a one-sample t-test of a population mean using the given sample statistics. *Hypothesized Mean* and *Tail* are defined as above. Results are returned in list form as *{Probability value, t-test statistic}*.

Example: TTest (4, 1, 12, 4, "≠") yields {1, 0}.

TTest2 Command

`TTest2(<List of Sample Data 1>, <List of Sample Data 2>, <Tail>, <Boolean Pooled>)`

Performs a t-test of the difference between two population means using the given lists of sample data. *Tail* has possible values "<", ">" , "≠" that determine the following alternative hypotheses:

"<" = difference in population means < 0

">" = difference in population means > 0

"≠" = difference in population means ≠ 0

If *Pooled* = true, then population variances are assumed equal and sample standard deviations are combined in calculation.

If *Pooled* = false, then population variances are not assumed equal and sample standard deviations are not combined.

Results are returned in list form as *{Probability value, t-test statistic}*.

`TTest2(<Sample Mean 1>, <Sample Standard Deviation 1>, <Sample Size 1>, <Sample Mean 2>, <Sample Standard Deviation 2>, <Sample Size 2>, <Tail>, <Boolean Pooled>)`

Performs a t-test of the difference between two population means using the given sample statistics. *Tail* and *Pooled* are defined as above. Results are returned in list form as *{Probability value, t-test statistic}*.

TTestPaired Command

`TTestPaired(<List of Sample Data 1>, <List of Sample Data 2>, <Tail>)`

Performs a paired t-test using the given lists of paired sample data. *Tail* has possible values "<", ">" , "≠" that determine the following alternative hypotheses:

"<" = $\mu < 0$

">" = $\mu > 0$

"≠" = $\mu \neq 0$

(μ is the mean paired difference of the population)

Results are returned in list form as *{Probability value, t-test statistic}*.

Example: `TTestPaired({1, 2, 3, 4, 5}, {1, 1, 3, 5, 5}, "<")` yields *{0.5, 0}*.

Variance Command

Variance(<List of Raw Data>)

Calculates the variance of list elements.

Example: `Variance({1, 2, 3})` yields *0.67*.

Variance(<List of Numbers>, <List of Frequencies>)

Calculates the variance of list elements, considering the frequencies.

Example: `Variance({1, 2, 3}, {1, 2, 1})` yields *0.5*.

CAS Syntax

Variance(<List of Numbers>)

Calculates the variance of list elements. If the list contains undefined variables, it yields a formula for the variance.

Example: `Variance({1, 2, a})` yields .

Probability Commands

<links/>

Bernoulli Command

Bernoulli(<Probability p>, <Boolean Cumulative>)

For *Cumulative = false* returns the bar graph of Bernoulli distribution where probability of success is equal to *p*.

For *Cumulative = true* returns the bar graph of cumulative Bernoulli distribution.

BinomialCoefficient Command

`BinomialCoefficient(<Number>, <Number>)`

Calculates the binomial coefficient . The first Number represents all elements n and the second Number represents the selected elements r .

Example: `BinomialCoefficient(5, 3)` yields 10.

Hint: If your input in the CAS View contains undefined variables, then this command yields a formula for the binomial coefficient. Example: `BinomialCoefficient(n, 3)` yields $\frac{n^3 - 3n^2 + 2n}{6}$.

Note: See also `NPr` command.

BinomialDist Command

`BinomialDist(<Number of Trials>, <Probability of Success>)`

Returns a bar graph of a Binomial distribution.

The parameter *Number of Trials* specifies the number of independent Bernoulli trials and the parameter *Probability of Success* specifies the probability of success in one trial.

`BinomialDist(<Number of Trials>, <Probability of Success>, <Boolean Cumulative>)`

Returns a bar graph of a Binomial distribution when *Cumulative* = false.

Returns a graph of a cumulative Binomial distribution when *Cumulative* = true.

First two parameters are same as above.

`BinomialDist(<Number of Trials>, <Probability of Success>, <Variable Value>, <Boolean Cumulative>)`

Let X be a Binomial random variable and let v be the variable value.

Returns $P(X = v)$ when *Cumulative* = false.

Returns $P(X \leq v)$ when *Cumulative* = true.

First two parameters are same as above.

Note: A simplified syntax is available to calculate $P(u \leq X \leq v)$: e.g. `BinomialDist(10, 0.2, 1..3)` yields 0.77175, that is the same as `BinomialDist(10, 0.2, {1, 2, 3})`. This syntax also works in the CAS View

CAS Specific Syntax

In CAS View only one syntax is allowed:

`BinomialDist(<Number of Trials>, <Probability of Success>, <Variable Value>, <Boolean Cumulative>)`

Let X be a Binomial random variable and let v be the variable value.

Returns $P(X = v)$ when *Cumulative* = false.

Returns $P(X \leq v)$ when *Cumulative* = true.

Example:

Assume transferring three packets of data over a faulty line. The chance an arbitrary packet transferred over this line becomes corrupted is , hence the probability of transferring an arbitrary packet successfully is .

- `BinomialDist(3, 0.9, 0, false)` yields , the probability of none of the three packets being transferred successfully.
- `BinomialDist(3, 0.9, 1, false)` yields , the probability of exactly one of three packets being transferred successfully.

- `BinomialDist(3, 0.9, 2, false)` yields , the probability of exactly two of three packets being transferred successfully.
- `BinomialDist(3, 0.9, 3, false)` yields , the probability of all three packets being transferred successfully.
- `BinomialDist(3, 0.9, 0, true)` yields , the probability of none of the three packets being transferred successfully.
- `BinomialDist(3, 0.9, 1, true)` yields , the probability of at most one of three packets being transferred successfully.
- `BinomialDist(3, 0.9, 2, true)` yields , the probability of at most two of three packets being transferred successfully.
- `BinomialDist(3, 0.9, 3, true)` yields I , the probability of at most three of three packets being transferred successfully.
- `BinomialDist(3, 0.9, 4, false)` yields 0 , the probability of exactly four of three packets being transferred successfully.
- `BinomialDist(3, 0.9, 4, true)` yields I , the probability of at most four of three packets being transferred successfully.

Cauchy Command

`Cauchy(<Median>, <Scale>, x)`

Creates cumulative density function (cdf) of Cauchy distribution.

`Cauchy(<Median>, <Scale>, x, <Boolean Cumulative>)`

If *Cumulative* is true, creates cumulative distribution function of Cauchy distribution, otherwise creates pdf of Cauchy distribution.

`Cauchy(<Median>, <Scale>, <Variable Value>)`

Calculates the value of cumulative distribution function of Cauchy distribution at *Variable Value* v , i.e. the probability $P(X \leq v)$ where X is a random variable with Cauchy given by parameters *Median* and *Scale*.

Note: Returns the probability for a given x -coordinate's value (or area under the Cauchy distribution curve to the left of the given x -coordinate).

Example: `Cauchy(1, 2, 3)` yields 0.75 in the Algebra View and in the CAS View.

ChiSquared Command

`ChiSquared(<Degrees of Freedom>, x)`

Creates cumulative density function (cdf) of Chi squared distribution with the appropriate degrees of freedom.

`ChiSquared(<Degrees of Freedom>, x, <Boolean Cumulative>)`

If the logical value is *true*, creates cumulative distribution function of Chi squared distribution, otherwise creates pdf of Chi squared distribution.

`ChiSquared(<Degrees of Freedom>, <Variable Value>)`

Calculates the value of cumulative distribution function of Chi squared distribution at *Variable Value v*, i.e. the probability $P(X \leq v)$ where X is a random variable with Chi squared distribution with the appropriate degrees of freedom.

Note: Returns the probability for a given *x*-coordinate's value (or area under the Chi squared distribution curve to the left of the given *x*-coordinate).

Example: `ChiSquared(4, 3)` yields , which is approximately 0.44.

ChiSquaredTest Command

`ChiSquaredTest(<Matrix>)`

Performs a chi-squared test that compares the given matrix of observed counts against the matrix of expected counts determined by the hypothesis of independence.

The matrix of expected counts is calculated internally. Each expected count is found from the row and column totals of the given matrix of observed counts using the rule:

Results are returned in list form as *{Probability value, chi-squared test statistic}*.

Example: `ChiSquaredTest({{1, 2, 1}, {3, 2, 3}})` yields {0.69, 0.75}.

`ChiSquaredTest(<Observed List>, <Expected List>)`

Performs a Goodness of Fit test that compares the given list of observed counts against the given list of expected counts.

Results are returned in list form as *{Probability value, chi-squared test statistic}*.

Example: `ChiSquaredTest({1, 2, 3, 4}, {3, 2, 4, 2})` yields {0.31, 3.58}.

`ChiSquaredTest(<Observed Matrix>, <Expected Matrix>)`

Performs a chi-squared test that compares the given matrix of observed counts against the given matrix of expected counts.

Results are returned in list form as *{Probability value, chi-squared test statistic}*.

Example: `ChiSquaredTest({{1, 2, 1}, {3, 2, 3}}, {{2, 3, 2}, {4, 2, 3}})` yields {0.45, 1.58}.

Erlang Command

`Erlang(<Shape>, <Rate>, x)`

Creates cumulative density function (cdf) of Erlang distribution with parameters shape k and rate λ .

`Erlang(<Shape>, <Rate>, x, <Boolean Cumulative>)`

If *Cumulative* is true, creates cumulative distribution function of Erlang distribution, otherwise creates pdf of Erlang distribution.

`Erlang(<Shape>, <Rate>, <Variable Value>)`

Calculates the value of cumulative distribution function of Erlang distribution at variable value v , i.e. the probability $P(X \leq v)$ where X is a random variable with Erlang distribution given by parameters shape k and rate λ .

Note: Returns the probability for a given x -coordinate's value (or area under the Erlang distribution curve to the left of the given x -coordinate).

Exponential Command

`Exponential(<Lambda>, x)`

Creates cumulative density function (cdf) of exponential distribution with parameter *lambda*.

`Exponential(<Lambda>, x, <Boolean Cumulative>)`

If *Cumulative* is true, creates cumulative distribution function (cdf) of exponential distribution, otherwise creates pdf of Exponential distribution.

`Exponential(<Lambda>, <Variable Value>)`

Calculates the value of cumulative distribution function of Exponential distribution at variable value v , i.e. the probability $P(X \leq v)$ where X is a random variable with Exponential distribution with parameter *lambda*.

Note: Returns the probability for a given x -coordinate's value (or area under the Exponential distribution curve to the left of the given x -coordinate).

CAS Syntax

`Exponential(<Lambda>, <Variable Value>)`

Calculates the value of cumulative distribution function of exponential distribution at variable value v , i.e. the probability $P(X \leq v)$ where X is a random variable with Exponential distribution with parameter *lambda*.

Example: `Exponential(2, 1)` yields , which is approximately 0.86.

FDistribution Command

FDistribution(<Numerator Degrees of Freedom>, <Denominator Degrees of Freedom>, x)

Creates cumulative density function (cdf) of F-distribution with parameters n, d (n for *Numerator Degrees of Freedom*, d for *Denominator Degrees of Freedom*).

FDistribution(<Numerator Degrees of Freedom>, <Denominator Degrees of Freedom>, x, <Boolean Cumulative>)

If *Cumulative* is true, creates cumulative distribution function of F-distribution, otherwise creates pdf of F-distribution.

FDistribution(<Numerator Degrees of Freedom>, <Denominator Degrees of Freedom>, <Variable Value>)

Calculates the value of cumulative distribution function of F-distribution at *Variable Value v*, i.e. the probability $P(X \leq v)$ where X is a random variable with F-distribution given by parameters n, d (n for *Numerator Degrees of Freedom*, d for *Denominator Degrees of Freedom*).

Note: Returns the probability for a given x -coordinate's value (or area under the F-distribution curve to the left of the given x -coordinate).

Note: This command is also available in the CAS View.

Gamma Command

Gamma(<Number α >, <Number β >, x)

Creates cumulative density function (cdf) of gamma distribution with parameters α, β .

Gamma(<Number α >, <Number β >, x, <Boolean Cumulative>)

If *Cumulative* is true, creates cumulative distribution function of gamma distribution, otherwise creates pdf of gamma distribution.

Gamma(<Number α >, <Number β >, <Variable Value v>)

Calculates the value of cumulative distribution function of gamma distribution at v , i.e. the probability $P(X \leq v)$ where X is a random variable with gamma distribution given by parameters α, β .

Note:

Returns the probability for a given x -coordinate's value (or area under the gamma distribution curve to the left of the given x -coordinate).

HyperGeometric Command

`HyperGeometric(<Population Size>, <Number of Successes>, <Sample Size>)`

Returns a bar graph of a Hypergeometric distribution.

Parameters:

Population size: number of balls in the urn

Number of Successes: number of white balls in the urn

Sample Size: number of balls drawn from the urn

The bar graph shows the probability function of the number of white balls in the sample.

`HyperGeometric(<Population Size>, <Number of Successes>, <Sample Size>, <Boolean Cumulative>)`

Returns a bar graph of a Hypergeometric distribution when *Cumulative* = false.

Returns the graph of a cumulative Hypergeometric distribution when *Cumulative* = true.

First three parameters are same as above.

`HyperGeometric(<Population Size>, <Number of Successes>, <Sample Size>, <Variable Value>, <Boolean Cumulative>)`

Let X be a Hypergeometric random variable and v the variable value.

Returns $P(X = v)$ when *Cumulative* = false.

Returns $P(X \leq v)$ when *Cumulative* = true.

First three parameters are same as above.

CAS Syntax

In the CAS View you can use only the following syntax:

`HyperGeometric(<Population Size>, <Number of Successes>, <Sample Size>, <Variable Value>, <Boolean Cumulative>)`

Let X be a Hypergeometric random variable and v the variable value.

Returns $P(X = v)$ when *Cumulative* = false.

Returns $P(X \leq v)$ when *Cumulative* = true.

The first three parameters are the same as above.

Example:

Assume you select two balls out of ten balls, two of which are white, without putting any back.

- `HyperGeometric(10, 2, 2, 0, false)` yields , the probability of selecting zero white balls,
- `HyperGeometric(10, 2, 2, 1, false)` yields , the probability of selecting one white ball,
- `HyperGeometric(10, 2, 2, 2, false)` yields , the probability of selecting both white balls,
- `HyperGeometric(10, 2, 2, 3, false)` yields 0, the probability of selecting three white balls.
- `HyperGeometric(10, 2, 2, 0, true)` yields , the probability of selecting zero (or less) white balls,
- `HyperGeometric(10, 2, 2, 1, true)` yields , the probability of selecting one or less white balls,
- `HyperGeometric(10, 2, 2, 2, true)` yields 1, the probability of selecting two or less white balls and
- `HyperGeometric(10, 2, 2, 3, true)` yields 1, the probability of selecting three or less white balls.

InverseBinomial Command

`InverseBinomial(<Number of Trials>, <Probability of Success>, <Probability>)`

Returns least integer n such that $P(X \leq n) \geq p$, where p is the probability and X is binomial random variable given by *Number of Trials* and *Probability of Success*.

Note: See also BinomialDist Command.

InverseCauchy Command

`InverseCauchy(<Median>, <Scale>, <Probability>)`

Computes the inverse of cumulative distribution function of Cauchy distribution at probability p , where the Cauchy distribution is given by median m and scale s .

In other words, finds t such that $P(X \leq t) = p$, where X is Cauchy random variable.

Probability p must be from [0,1].

InverseChiSquared Command

`InverseChiSquared(<Degrees of Freedom>, <Probability>)`

Computes the inverse of cumulative distribution function of Chi squared distribution at probability p , where the Chi squared distribution has given d degrees of freedom.

In other words, finds t such that $P(X \leq t) = p$, where X is Chi squared random variable.

Probability p must be from [0,1].

InverseExponential Command

`InverseExponential(<Lambda>, <Probability>)`

Computes the inverse of cumulative distribution function of exponential distribution at probability p , where the exponential distribution is given by `Lambda`.

In other words, finds t such that $P(X \leq t) = p$, where X is exponential random variable.

Probability p must be from $[0,1]$.

InverseFDistribution Command

`InverseFDistribution(<Numerator Degrees of Freedom>, <Denominator Degrees of Freedom>, <Probability>)`

Computes the inverse of cumulative distribution function of F-distribution at probability p , where the F-distribution is given by the degrees of freedom.

In other words, finds t such that $P(X \leq t) = p$, where X is random variable with F-distribution.

Probability p must be from $[0,1]$.

InverseGamma Command

`InverseGamma(<Alpha>, <Beta>, <Probability>)`

Computes the inverse of cumulative distribution function of gamma distribution at probability p , where the gamma distribution is given by parameters `Alpha` and `Beta`.

In other words, finds t such that $P(X \leq t) = p$, where X is random variable with gamma distribution.

Probability p must be from $[0,1]$.

InverseHyperGeometric Command

`InverseHyperGeometric(<Population Size>, <Number of Successes>, <Sample Size>, <Probability>)`

Returns least integer n such that $P(X \leq n) \geq p$, where p is the probability and X is hypergeometric random variable given by *Population Size*, *Number of Successes* and *Sample Size*.

Note: See also HyperGeometric Command.

InverseLogNormal Command

`InverseLogNormal(<Mean>, <Standard Deviation>, <Probability>)`

Computes the inverse of cumulative distribution function of the log-normal distribution at probability p , where the log-normal distribution is given by mean μ and standard deviation σ .

In other words, it finds t such that $P(X \leq t) = p$, where X is a log-normal random variable.

Probability p must be from $[0, 1]$.

Examples:

- `InverseLogNormal(10, 20, 1/3)` computes 3.997.
- `InverseLogNormal(1000, 2, 1)` computes .

InverseLogistic Command

`InverseLogistic(<Mean>, <Scale>, <Probability>)`

Computes the inverse of cumulative distribution function of the Logistic distribution at probability p , where the Logistic distribution is given by mean μ and scale s .

In other words, finds t such that $P(X \leq t) = p$, where X is a Logistic random variable.

Probability p must be from $[0,1]$.

Example: `InverseLogistic(100, 2, 1)` yields .

InverseNormal Command

`InverseNormal(<Mean>, <Standard Deviation>, <Probability>)`

Calculates the function with given probability P , mean μ and standard deviation σ , where is the inverse of the cumulative distribution function Φ for $N(0,1)$.

Note: Returns the x -coordinate with the given probability to the left under the normal distribution curve.

InversePascal Command

`InversePascal(<n>, <p>, <Probability>)`

Returns least integer n such that $P(X \leq n) \geq p$, where p is the probability and X is Pascal random variable given by n and p .

Note: See also Pascal Command.

InversePoisson Command

`InversePoisson(<Mean>, <Probability>)`

Returns least integer n such that $P(X \leq n) \geq p$, where p is the probability and X is Poisson random variable with mean λ .

Note: See also Poisson Command.

InverseTDistribution Command

`InverseTDistribution(<Degrees of Freedom>, <Probability>)`

Computes the inverse of cumulative distribution function of t-distribution at p , where the t-distribution has d degrees of freedom. In other words, finds r such that $P(X \leq r) = p$, where X is random variable with t-distribution. Probability p must be from [0,1].

InverseWeibull Command

`InverseWeibull(<Shape>, <Scale>, <Probability>)`

Computes the inverse of cumulative distribution function of Weibull distribution at p , where the Weibull distribution is given by shape parameter k and scale parameter λ .

In other words, finds t such that $P(X \leq t) = p$, where X is random variable with Weibull distribution. Probability p must be from [0,1].

InverseZipf Command

`InverseZipf(<Number of Elements>, <Exponent>, <Probability>)`

Returns least integer n such that $P(X \leq n) \geq p$, where X is Zipf random variable given by *Number of Elements* and *Exponent* and p is the probability.

Note: See also Zipf Command.

LogNormal Command

LogNormal(<Mean>, <Standard Deviation>, x)

Creates cumulative density function (cdf) of log-normal distribution with parameters mean μ and standard deviation σ .

LogNormal(<Mean>, <Standard Deviation>, x, <Boolean Cumulative>)

If *Cumulative* is true, creates cumulative density function of log-normal distribution, otherwise creates pdf of log-normal distribution.

LogNormal(<Mean>, <Standard Deviation>, <Variable Value>)

Calculates the value of cumulative distribution function of log-normal distribution at variable value v , i.e. the probability $P(X \leq v)$ where X is a random variable with log-normal distribution given by parameters mean μ and standard deviation σ .

Note: Returns the probability for a given x -coordinate's value (or area under the log-normal distribution curve to the left of the given x -coordinate).

Logistic Command

Logistic(<Mean>, <Scale>, x)

Creates cumulative density function (cdf) of logistic distribution with parameters mean μ and scale s .

Logistic(<Mean>, <Scale>, x, <Boolean Cumulative>)

If *Cumulative* is true, creates cumulative distribution function of logistic distribution, otherwise creates pdf of logistic distribution.

Logistic(<Mean>, <Scale>, <Variable Value>)

Calculates the value of cumulative distribution function of logistic distribution at variable value v , i.e. the probability $P(X \leq v)$ where X is a random variable with logistic distribution given by parameters mean μ and scale s .

Note: Returns the probability for a given x -coordinate's value (or area under the logistic distribution curve to the left of the given x -coordinate).

Normal Command

`Normal(<Mean>, <Standard Deviation>, x)`

Creates cumulative density function (cdf) of normal distribution.

`Normal(<Mean>, <Standard Deviation>, x, <Boolean Cumulative>)`

If *Cumulative* is true, creates cumulative distribution function of normal distribution with mean μ and standard deviation σ , otherwise creates pdf of normal distribution.

`Normal(<Mean μ >, <Standard Deviation σ >, <Variable Value v>)`

Calculates the function at v where Φ is the cumulative distribution function for $N(0,1)$ with mean μ and standard deviation σ .

Note: Returns the probability for a given x -coordinate's value (or area under the normal distribution curve to the left of the given x -coordinate).

Example: `Normal(2, 0.5, 1)` yields 0.02 in the Algebra View and in the CAS View.

Pascal Command

`Pascal(<n>, <p>)`

Returns a bar graph of a Pascal distribution.

The Pascal distribution models the number of failures before the n th success in repeated mutually independent Bernoulli trials, each with probability of success p .

`Pascal(<n>, <p>, <Boolean Cumulative>)`

Returns a bar graph of a Pascal distribution when *Cumulative* = false.

Returns a graph of a cumulative Pascal distribution when *Cumulative* = true.

First two parameters are same as above.

`Pascal(<n>, <p>, <Variable Value>, <Boolean Cumulative>)`

Let X be a Pascal random variable and v the variable value.

Returns $P(X = v)$ when *Cumulative* = false.

Returns $P(X \leq v)$ when *Cumulative* = true.

First two parameters are same as above.

Note: This command also works in the CAS View.

Example:

If the number of independent Bernoulli trials that must be successful is $n = 1$, the probability of success in one trial is $p =$, then the probability of 2 failures before the success is given by

`Pascal(1, 1/6, 2, false)` which yields 0.12 in the Algebra View and 25/216 in the CAS View.

Poisson Command

`Poisson(<Mean>)`

Returns a bar graph of a Poisson distribution with given mean λ .

`Poisson(<Mean>, <Boolean Cumulative>)`

Returns a bar graph of a Poisson distribution when *Cumulative = false*.

Returns a graph of a cumulative Poisson distribution when *Cumulative = true*.

The first parameter is same as above.

`Poisson(<Mean>, <Variable Value v>, <Boolean Cumulative>)`

Let X be a Poisson random variable.

Returns $P(X = v)$ when *Cumulative = false*.

Returns $P(X \leq v)$ when *Cumulative = true*.

First parameter is same as above.

Examples:

- `Poisson(3, 1, true)` yields 0.2 in the *Algebra View* and in the *CAS View*.
- `Poisson(3, 1, false)` yields 0.15 in the *Algebra View* and in the *CAS View*.

Note: A simplified syntax is available to calculate $P(u \leq X \leq v)$: e.g. `Poisson(1, 1..5)` yields 0.63153, that is the same as `Poisson(1, {1, 2, 3, 4, 5})`.

RandomBetween Command

`RandomBetween(<Minimum Integer> , <Maximum Integer>)`

Generates a random integer between *minimum* and *maximum* (inclusive).

Example: `RandomBetween(0, 10)` yields a number between 0 and 10 (inclusive)

`RandomBetween(<Minimum Integer> , <Maximum Integer> , <Boolean Fixed>)`

If *Boolean Fixed = "true"*, it generates a random integer between *minimum* and *maximum* (inclusive), which is updated just once (when file is loaded and also on undo/redo).

Example: `RandomBetween(0, 10, true)` yields a number between 0 and 10 (inclusive)

Note: Press F9 to see the difference between those two syntaxes.

Note: See also `SetSeed` command, `RandomElement` command, `RandomBinomial` command, `RandomNormal` command, `RandomPoisson` command, `RandomUniform` command.

RandomBinomial Command

RandomBinomial(<Number of Trials>, <Probability>)

Generates a random number from a binomial distribution with n trials and probability p .

Example:

RandomBinomial(3, 0.1) gives $j \in \{0, 1, 2, 3\}$, where the probability of getting j is the probability of an event with probability 0.1 occurring j times in three tries.

Note: See also SetSeed command, RandomBetween command, RandomElement command, RandomNormal command, RandomPoisson command, RandomUniform command.

RandomNormal Command

RandomNormal(<Mean>, <Standard Deviation>)

Generates a random number from a normal distribution with given mean and standard deviation.

Example: RandomNormal(3, 0.1) yields a random value from a normal distribution with a mean of 3 and standard deviation of 0.1.

Note: See also SetSeed command, RandomBetween command, RandomElement command, RandomBinomial command, RandomPoisson command, RandomUniform command.

RandomPoisson Command

RandomPoisson(<Mean>)

Generates a random number from a Poisson distribution with given mean.

Example: RandomPoisson(3) yields a random value from a Poisson distribution with a mean of 3.

Note: See also SetSeed command, RandomBetween command, RandomElement command, RandomBinomial command, RandomNormal command, RandomUniform command.

RandomUniform Command

`RandomUniform(<Min>, <Max>)`

Returns random real number from uniform distribution on interval [*min, max*].

Example: `RandomUniform(0, 1)` returns a random number between 0 and 1

`RandomUniform(<Min>, <Max>, <Number of Samples n>)`

Returns a list of *n* random real numbers from uniform distribution on interval [*min, max*].

Example: `RandomUniform(0, 1, 3)` returns a list of three random numbers between 0 and 1

Notes: `RandomUniform(0,1)` is equivalent to `random()` (see Predefined Functions and Operators). See also `SetSeed` Command, `SetSeed`, `RandomBetween` Command, `RandomElement` Command, `RandomBinomial` Command, `RandomBinomial`, `RandomNormal` Command, `RandomNormal`, `RandomPoisson` Command, `RandomPoisson` commands.

TDistribution Command

`TDistribution(<Degrees of Freedom>, x)`

Creates cumulative density function (cdf) of t-distribution with given degrees of freedom.

`TDistribution(<Degrees of Freedom>, x, <Boolean Cumulative>)`

If *Cumulative* is true, creates cumulative distribution function of t-distribution, otherwise creates pdf of t-distribution.

`TDistribution(<Degrees of Freedom>, <Variable Value>)`

Calculates the value of cumulative distribution function of t-distribution at *variable value v*, i.e. the probability $P(X \leq v)$ where *X* is a random variable with t-distribution with given degrees of freedom.

Example: `TDistribution(10, 0)` yields 0.5.

Note: Returns the probability for a given *x*-coordinate's value (or area under the t-distribution curve to the left of the given *x*-coordinate).

CAS Syntax

`TDistribution(<Degrees of Freedom>, <Variable Value>)`

Calculates the value of cumulative distribution function of t-distribution at *variable value v*, i.e. the probability $P(X \leq v)$ where *X* is a random variable with t-distribution with given degrees of freedom.

Example: `TDistribution(10, 0)` yields 0.5.

Triangular Command

Triangular(<Lower Bound>, <Upper Bound>, <Mode>, x)

Creates cumulative density function of triangular distribution with parameters *lower Bound*, *upper Bound*, *Mode*.

Triangular(<Lower Bound>, <Upper Bound>, <Mode>, x, <Boolean Cumulative>)

If *Cumulative* is true, creates cumulative distribution function of triangular distribution, otherwise creates probability density function of triangular distribution.

Triangular(<Lower Bound>, <Upper Bound>, <Mode>, <Variable Value>)

Calculates the value of cumulative distribution function of triangular distribution at *Variable Value v*, i.e. the probability $P(X \leq v)$ where X is a random variable with triangular distribution given by parameters *lower Bound*, *upper Bound*, *Mode*.

Note: Returns the probability for a given *x*-coordinate's value (or area under the triangular distribution curve to the left of the given *x*-coordinate).

Example: Triangular(0, 5, 2, 2) yields 0.4.

Uniform Command

Uniform(<Lower Bound>, <Upper Bound>, x)

Returns the cumulative density function of uniform distribution on interval [*lower bound*, *upper bound*].

Uniform(<Lower Bound>, <Upper Bound>, x, <Boolean Cumulative>)

For *Cumulative=false* returns the probability density function of uniform distribution on interval [*lower bound*, *upper bound*].

For *Cumulative=true* returns the cumulative distribution function of the same distribution.

Uniform(<Lower Bound>, <Upper Bound>, <Variable Value>)

Returns the value of cumulative distribution function at the given variable value *v* (i.e. $P(X < v)$) for uniform distribution on interval [*lower bound*, *upper bound*]

Weibull Command

`Weibull(<Shape>, <Scale>, x)`

Creates cumulative density function (cdf) of Weibull distribution with parameters shape k and scale λ

`Weibull(<Shape>, <Scale>, x, <Boolean Cumulative>)`

If *Cumulative* is true, creates cumulative distribution function of Weibull distribution, otherwise creates pdf of Weibull distribution.

`Weibull(<Shape>, <Scale>, <Variable Value>)`

Calculates the value of cumulative distribution function of Weibull distribution at variable value v , i.e. the probability $P(X \leq v)$ where X is a random variable with Weibull distribution given by parameters shape k and scale λ .

Note: Returns the probability for a given x -coordinate's value (or area under the Weibull distribution curve to the left of the given x -coordinate).

Examples:

- `Weibull(0.5, 1, 0)` yields 0.
- `Weibull(0.5, 1, 1)` yields .

Zipf Command

`Zipf(<Number of Elements>, <Exponent>)`

Returns a bar graph of a Zipf distribution.

Parameters:

Number of Elements: number of elements whose rank we study

Exponent: exponent characterizing the distribution

`Zipf(<Number of Elements>, <Exponent> , <Boolean Cumulative>)`

Returns a bar graph of a *Zipf distribution* when *Cumulative* = false.

Returns a graph of a cumulative Zipf distribution when *Cumulative* = true.

First two parameters are same as above.

`Zipf(<Number of Elements>, <Exponent> , <Variable Value v>, <Boolean Cumulative>)`

Let X be a Zipf random variable.

Returns $P(X = v)$ when *Cumulative* = false.

Returns $P(X \leq v)$ when *Cumulative* = true.

First two parameters are same as above.

Example: `Zipf(10, 1, 5, false)` yields 0.07 in the *Algebra View* and in the *CAS View*.

ZMean2Estimate Command

ZMean2Estimate(<List of Sample Data 1>, <List of Sample Data 2>, < σ_1 >, < σ_2 >, <Level>)

Calculates a Z confidence interval estimate of the difference between two population means using the given sample data sets, population standard deviations and confidence *Level*.

Results are returned in list form as {lower confidence limit, upper confidence limit}.

Example:

Two sample data $list1 = \{1, 4, 5, 4, 1, 3, 4, 2\}$, $list2 = \{2, 1, 3, 1, 2, 5, 2, 4\}$ are given. The standard deviation of $list1$ is $\sigma_1 = \text{sqrt}(2)$, the standard deviation of $list2$ is $\sigma_2 = \text{sqrt}(1.75)$ and the confidence level is 0.75.

`ZMean2Estimate(list1, list2, σ1, σ2, 0.75)` yields $list3 = \{-0.29, 1.29\}$.

ZMean2Estimate(<Sample Mean 1>, < σ_1 >, <Sample Size 1>, <Sample Mean 2 >, < σ_2 >, <Sample Size 2>, <Level>)

Calculates a Z confidence interval estimate of the difference between two population means using the given sample means, population standard deviations and confidence *Level*.

Results are returned in list form as {lower confidence limit, upper confidence limit}.

ZMean2Test Command

ZMean2Test(<List of Sample Data 1>, < σ_1 >, <List of Sample Data 2>, < σ_2 >, <Tail>)

Performs a Z test of the difference between two population means using the given lists of sample data and the population standard deviations. Tail has possible values "<", ">" , "≠" that determine the following alternative hypotheses:

"<" = difference in population means < 0

">" = difference in population means > 0

"≠" = difference in population means $\neq 0$

Results are returned in list form as {Probability value, Z test statistic}.

ZMean2Test(<Sample Mean 1 >, < σ_1 >, <Sample Size 1>, <Sample Mean 2 >, < σ_2 >, <Sample Size 2>, <Tail>)

Performs a Z test of the difference between two population means using the given sample statistics and population standard deviations. *Tail* is defined as above.

ZProportion2Estimate Command

ZProportion2Estimate(<Sample Proportion 1>, <Sample Size 1>, <Sample Proportion 2>, <Sample Size 2>, <Level>)

Calculates a Z confidence interval estimate of the difference between two proportions using the given sample statistics and confidence *Level*.

Results are returned in list form as {*lower confidence limit*, *upper confidence limit*}.

ZProportion2Test Command

ZProportion2Test(<Sample Proportion 1>, <Sample Size 1>, <Sample Proportion 2>, <Sample Size 2>, <Tail>)

Performs a test of the difference between two population proportions using the given sample statistics. *Tail* has possible values "<", ">" , "≠". These specify the alternative hypothesis as follows.

"<" = difference in population proportions < 0

">" = difference in population proportions > 0

"≠" = difference in population proportions ≠ 0

Results are returned in list form as {Probability value, Z test statistic}.

ZProportionEstimate Command

ZProportionEstimate (<Sample Proportion>, <Sample Size>, <Level>)

Calculates a Z confidence interval estimate of a population proportion using the given sample statistics and confidence level. Results are returned in list form as {*lower confidence limit*, *upper confidence limit*}.

ZProportionTest Command

ZProportionTest(<Sample Proportion>, <Sample Size>, <Hypothesized Proportion>, <Tail>)

Performs a one sample Z test of a proportion using the given sample statistics. *Hypothesized Proportion* is the population proportion assumed in the null hypothesis. *Tail* has possible values "<", ">" , "≠". These specify the alternative hypothesis as follows.

"<" = population proportion < *Hypothesized Proportion*

">" = population proportion > *Hypothesized Proportion*

"≠" = population proportion ≠ *Hypothesized Proportion*

Results are returned in list form as {Probability value, Z test statistic}.

ZMeanEstimate Command

ZMeanEstimate(<List of Sample Data>, < σ >, <Level>)

Calculates a Z confidence interval estimate of a population mean using the given sample data, the population standard deviation and confidence *Level*.

Results are returned in list form as {lower confidence limit, upper confidence limit}.

ZMeanEstimate(<Sample Mean>, < σ >, <Sample Size>, <Level>)

Calculates a Z confidence interval estimate of a population mean using the given sample statistics, the population standard deviation and confidence level.

Results are returned in list form as {lower confidence limit, upper confidence limit}.

ZMeanTest Command

ZMeanTest(<List of Sample Data>, < σ >, <Hypothesized Mean>, <Tail>)

Performs a one sample Z test of a population mean using the given list of sample data and the population standard deviation. *Hypothesized Mean* is the population mean assumed in the null hypothesis. *Tail* has possible values "<", ">" , "≠". These specify the alternative hypothesis as follows.

"<" = population mean < *Hypothesized Mean*

">" = population mean > *Hypothesized Mean*

"≠" = population mean ≠ *Hypothesized Mean*

Results are returned in list form as {Probability value, Z test statistic}.

ZMeanTest(<Sample Mean>, < σ >, <Sample Size>, <Hypothesized Mean>, <Tail>)

Performs a *one sample Z test* of a population mean using the given sample statistics and the population standard deviation. *Hypothesized Mean* and *Tail* are defined as above.

Results are returned in list form as {Probability value, Z test statistic}.

Spreadsheet Commands

These commands are designed for referencing data from Spreadsheet View and copying data into it. <links/>

Cell Command

Cell(<Column>, <Row>)

Returns copy of spreadsheet cell in given column and row.

Example: Cell(2, 1) returns copy of B1.

Note:

- By default the cells in spreadsheet cells are auxiliary and in such case this command returns auxiliary object as well.
- You must make sure that the cells you refer to are **earlier** in the Construction Protocol than this command.

CellRange Command

CellRange(<Start Cell>, <End Cell>)

Creates a list containing the cell values in this cell range.

Example:

Let $A1 = 1, A2 = 4, A3 = 9$ be spreadsheet cells values. Then CellRange (A1, A3) returns the list {1, 4, 9}.

Note: A1 : A3 is a shorter syntax.

Column Command

Column(<Spreadsheet Cell>)

Returns the column of the cell as a number (starting at 1).

Example: q = Column (B3) returns $q = 2$ since column B is the second column of the spreadsheet.

ColumnName Command

ColumnName(<Spreadsheet Cell>)

Returns the column name of the cell as a text.

Example: r = ColumnName (A1) creates $r = A$ and shows such text - A - in the Graphics View.

FillCells Command

FillCells(<CellRange>, <Object>)

Copies the value/equation etc. of the object to the given cellrange. Resulting cells are free objects, i.e. independent of object.

Notes: CellRange has to be entered like this: e.g.: B2:D5. Object can be anything, e.g.: 3, RandomBetween(0, 10), Circle(A, B). Cells are labelled by column and row, e.g.: B2.

FillCells(<Cell>, <List>)

Copies values from the list to the first cells on the right of the given cell. Resulting cells are ([Free, Dependent and Auxiliary Objects|free objects]), i.e. independent of the list.

FillCells(<Cell>, <Matrix>)

Copies values from the matrix into the spreadsheet. The upper left corner of the matrix is matched to the given cell. Resulting cells are free objects, i.e. independent of the matrix.

Note: See also FillRow and FillColumn commands.

Note: You can use `FillCell(cell, Transpose({list}))` to fill vertically. The extra braces convert the list into a matrix thus `{list}`

FillColumn Command

FillColumn(<Column>, <List>)

Copies values from the list to the first cells of the column given by number (1 for A, 2 for B, etc.). Resulting cells are free objects, i.e. independent of the list.

Note: See also the FillRow and FillCells commands.

FillRow Command

FillRow(<Row>, <List>)

Copies values from the list to the first cells of the row given by number. Resulting cells are free objects, i.e. independent of the list.

Note: See also the FillColumn and FillCells commands.

Row Command

Row(<Spreadsheet Cell>)

Returns the row number of the ([Spreadsheet View]spreadsheet) cell (starting at 1).

Example: `r = Row(B3)` yields $r = 3$.

Scripting Commands

These commands are substitutes for features accessible e.g. via Properties Dialog and are meant to simplify scripting in GeoGebra.

Note: These commands don't return any object, therefore cannot be nested in other commands.

<links/>

Button Command

Button()

Creates a new button.

Button(<Caption>)

Creates a new button with given caption.

Example: `Button("Ok")` creates a button in the left upper corner of the Graphics View with the caption *Ok*.

CenterView Command

CenterView(<Center Point>)

Translates the Graphics View so that the specified point is in the center.

Example: CenterView((0, 0)) moves the origin to the center of the screen.

Note: If multiple *Graphics Views* are present, the active one is used.

Checkbox Command

Checkbox()

Creates a checkbox.

Checkbox(<Caption>)

Creates a checkbox with given caption.

Checkbox(<List>)

Creates a checkbox which, when unchecked, hides listed objects.

Example: Let *A* and *B* be points. *c* = Checkbox({*A*, *B*}) creates checkbox *c*. When *c* is checked, *A* and *B* are visible, otherwise they are hidden.

Checkbox(<Caption>, <List>)

Creates checkbox with given caption which, when unchecked, hides listed objects.

CopyFreeObject Command

CopyFreeObject(<Object>)

Creates a free copy of the object. Preserves all basic Object Properties and copy of Auxiliary Object is auxiliary as well.

Delete Command

Delete(<Object>)

Deletes the object and all its dependent objects.

Example: `Delete(a)` clears *a*.

CAS Syntax

Delete(<Object>)

Deletes the object and all its dependent objects in GeoGebra and removes any value assigned to the object in the CAS.

Example: `Delete(a)` clears *a*.

Note: See also Delete tool.

Execute Command

Execute(<List of Texts>)

Executes list of commands entered as texts.

Note: Please note that you always need to use English (US) commands within this list of texts, no matter which language option you selected for GeoGebra.

Examples: `Execute({ "A=(1,1)", "B=(3,3)", "C = Midpoint(A, B)" })` creates points A, B and their midpoint C. `Execute(Join({ "f_{1} = 1", "f_{2} = 1" }, Sequence("f_{i+(i+2)+}" = f_{i+(i+1)+} + f_{i+i+}", i, 1, 10)))` creates first 10 elements of Fibonacci sequence.

Execute(<List of Texts>, <Parameter>, ... , <Parameter>)

Replaces %1 for the first parameter, %2 for the second parameter and so on in each text in list. Up to 9 parameters can be specified. After the replacement, resulting scripts are executed.

Example:

`Execute({ "Segment (%1, %2)", "Midpoint (%1, %2)" }, A, B)` creates the segment *AB* and its midpoint.

Note: Command names **must be in English (US)** in the texts for this command to work.

Note: If you need to use a quote ("") then you can use `UnicodeToLetter(34)`

GetTime Command

GetTime()

Returns a list with the current time and date in this order:

milliseconds, seconds, minutes, hours (0-23), date, month (1-12), year, month (as text), day (as text), day (1 = Sunday, 2 = Monday, etc)

Example: GetTime () returns a list such as {647, 59, 39, 23, 28, 2, 2011, "February", "Monday", 2}.

GetTime("<Format>")

Creates a text using *Format* as a template replacing any of the following characters when prefixed by a backslash (\):

d, D, j, l, N, S, w, z, W, F, m, M, n, t, L, Y, y, a, A, g, G, h, H, i, s, U - the explanation to these characters are here <http://php.net/manual/en/function.date.php>

Example: GetTime ("The date is \l the \j\S of \F \Y") might give *The date is Thursday the 5th of July 2012*

HideLayer Command

HideLayer(<Number>)

Makes all objects in given layer invisible. Does not override Conditional Visibility.

Pan Command

Pan(<x>, <y>)

Shifts the current view by *x* pixels to the right and *y* pixels upwards.

Pan(<x>, <y>, <z>)

Shifts the current view by (*x*, *y*, *z*) if it's a 3D View, or just by (*x,y*) for a 2D View

Notes: If multiple Graphics ViewGraphics Views are present, the active one is used. See also ZoomIn CommandZoomIn, ZoomOut CommandZoomOut, SetActiveView CommandSetActiveView commands.

ParseToFunction Command

`ParseToFunction(<Text>)`

Parses the text containing the function definition and creates the corresponding function.

Examples:

- `ParseToFunction("x^2")` creates the function $f(x) = x^2$.
- `ParseToFunction("t+2/t")` creates the function $f(t) = t + 2/t$.

`ParseToFunction(<Function>, <Text>)`

Parses the string and stores the result to a function f , which must be defined and free before the command is used.

Example: Define $f(x) = 3x^2 + 2$ and `text1 = "f(x) = 3x + 1"`. `ParseToFunction(f, text1)` returns $f(x) = 3x + 1$.

`ParseToFunction(<Text>, <List of variables>)`

Parses the text containing the function definition and creates the corresponding function of the variables defined in the list.

Example: `ParseToFunction("2u+3v", {"u", "v"})` creates the function $a(u,v) = 2u + 3v$.

Note: See also ParseToNumber command.

ParseToNumber Command

`ParseToNumber(<Number>, <Text>)`

Parses the text and stores the result to a number a , which must be defined and free before the command is used.

Example: Define `a = 3` and `text1 = "6"`. `ParseToNumber(a, text1)` returns $a = 6$.

Note: This is a scripting command which only sets the value of a number once. To convert a text `text1` into a number which is updated dynamically, use `FromBase(text1,10)`.

`ParseToNumber(<Text>)`

Parses the text and stores the result to a number.

Example: `ParseToNumber("1+2+5-pi")` creates the number $a = 4.86$.

Note: See also ParseToFunction command.

Repeat Command

`Repeat(<Number>, <Scripting Command>, <Scripting Command>, ...)`

Repeats the execution of scripting commands *n* times, where *n* is the given *Number*.

Example: `Turtle().Click the "Play" button displayed at bottom left. Repeat(8, TurtleForward(turtle1, 1), TurtleRight(turtle1, 45°))` The turtle moves and draws a regular octagon.

PlaySound Command

`PlaySound(<URL>)`

Plays an MP3 (.mp3) file

Examples:

- `PlaySound("http://static.geogebra.org/welcome_to_geogebra.mp3")`
- `PlaySound("#J2sQQfwQ")` plays an .mp3 that has been uploaded to GeoGebra [1]
- `PlaySound("https://drive.google.com/uc?id=0B7xCmZaU3oU2eXFNUzd6ZlZJS0U&authuser=0&e")` plays an .mp3 from Google Drive
- `PlaySound("https://www.dropbox.com/s/27skpv82odjp7ej/material-1264825.mp3?dl=1")` plays an .mp3 from DropBox

Note: To work on iOS (and also if you want immediate playback) then you can encode the .mp3 as an inline base64-encoded data: URL, see <https://www.geogebra.org/m/wztkqxuv> for an example. It must start with the *exact* string `data:audio/mp3;base64,` to work in GeoGebra Classic 5. You can use this utility to convert .MP3s into the syntax needed in GeoGebra <https://test.geogebra.org/~mike/utils/base64Encode.html>

`PlaySound(<Boolean Play>)`

Pause or resume play (not MP3 files)

`PlaySound(true) = play, PlaySound(false) = pause.`

`PlaySound(<Function>, <Min Value>, <Max Value>)`

Plays a sound generated by Function, a time-valued function with range [-1,1]. The time units are seconds and the sound is played from time Min Value to Max Value. Sound is generated by 8-bit samples taken at a rate of 8000 samples per second.

Example: `PlaySound(sin(440 2Pi x), 0, 1)` This plays a pure sine wave tone at 440 Hz (musical note A) for one second.

`PlaySound(<Function>, <Min Value>, <Max Value>, <Sample Rate>, <Sample Depth>)`

Plays a sound generated by Function, a time-valued function with range [-1,1]. The time units are seconds and the sound is played from time Min Value to Max Value. The sampling method is specified by "Sample Depth" and "Sample Rate".

"Sample Rate" is the number of sample function values taken each second. Allowable values are 8000, 11025, 16000, 22050, or 44100

"Sample Depth" is the data size of a sample in bits. Allowable values are 8 and 16.

`PlaySound(<Note>, <Duration>, <Instrument>)` (GeoGebra Classic 5 only)

Plays a MIDI note.

Note is an integer from 0 to 127 that represents a musical note given by the table below. When note = 60 a Middle C is played.

Duration is the time to play the note in seconds.

Instrument is an integer that represents the synthesized instrument used to play the note. See technical specifications [2] for possible instruments.

Most instruments are supported, but there are differences between computer platforms.

MIDI Notes

References

[1] <http://www.geogebra.org/m/J2sQQfwQ>

[2] https://web.archive.org/web/20130919034922/http://www.classicalmidiconnection.com/General_Midi.html

Rename Command

Rename(<Object>, <Name>)

Sets the label of given object to the given name.

Example: Let $c : x^2 + 2y^2 = 2$. Rename (c, ell) sets the label to *ell*.

RunClickScript Command

RunClickScript(<Object>)

Runs the click script associated with the Object (if it has one).

Example: Let *A* and *B* be points. The *OnClick* script for *B* is SetValue(B, (1, 1)). Setting the *OnClick* script of *A* as RunClickScript(B), moves point *B* to (1,1) when point *A* is clicked.

Note: See also RunUpdateScript command.

RunUpdateScript Command

RunUpdateScript(<Object>)

Runs the update script associated with the Object (if it has one).

Note: See also RunClickScript command.

Turtle Command

Turtle()

Creates a turtle at the coordinate origin.

Note: See also TurtleForward, TurtleBack, TurtleLeft, TurtleRight, TurtleUp and TurtleDown commands.

TurtleLeft Command

TurtleLeft(<Turtle>, <Angle>)

The turtle turns to the left by a given angle.

Example:

`TurtleLeft(turtle, 1)` turns the turtle to the left by 1 rad, if *Pause* button is displayed. Otherwise you must press the *Play* button so that the rotation is effected.

Example: If you enter `TurtleLeft(turtle, 1°)` the turtle turns to the left by 1 degree.

Note: See also Turtle, TurtleBack, TurtleForward and TurtleRight commands.

TurtleUp Command

TurtleUp(<Turtle>)

Instructs the named turtle not trace its movement from now.

Note: See also command TurtleDown

TurtleRight Command

TurtleRight(<Turtle>, <Angle>)

The turtle turns to the right by a given angle.

Example: TurtleRight(turtle, 1) turns the turtle to the right by 1 rad, if *Pause* button is displayed. Otherwise you must press the *Play* button so that the rotation is effected. Note: If you enter TurtleRight(turtle, 1°) the turtle turns to the right by 1 degree.

Note: See also Turtle, TurtleBack, TurtleForward and TurtleLeft commands.

TurtleDown Command

TurtleDown(<Turtle>)

Authorizes the turtle named to trace its movement from now.

Note: See also command TurtleUp

TurtleForward Command

`TurtleForward(<Turtle>, <Distance>)`

The turtle moves forward with given distance.

Example:

If the turtle is at the origin of the coordinates and the *Pause* button is displayed the command `TurtleForward(turtle, 2)` moves the turtle to the point (2, 0). Otherwise you must press the *Play* button so that the displacement is effected.

Note: See also `Turtle`, `TurtleBack`, `TurtleLeft` and `TurtleRight` commands.

TurtleBack Command

`TurtleBack(<Turtle>, <Distance>)`

The turtle moves back with given distance.

Example:

If the turtle is at the origin of the coordinates and the *Pause* button is displayed the command `TurtleBack(turtle, 2)` moves the turtle to the point (-2, 0). Otherwise you must press the *Play* button so that the displacement is effected.

Note: See also `Turtle`, `TurtleForward`, `TurtleLeft` and `TurtleRight` commands.

SelectObjects Command

`SelectObjects()`

Deselects all selected objects.

`SelectObjects(<Object>, <Object>, ...)`

Deselects all objects and selects objects passed as parameters. All parameters must be labeled objects.

Examples:

- Let *A*, *B* and *C* be points. `SelectObjects (A, B, C)` selects points *A*, *B* and *C*.
- The command `SelectObjects (Midpoint (A, B))` has no effect, besides deselecting all selected objects.

Note: This command now cancels any drag that is in progress (useful in scripts).

SetActiveView Command

SetActiveView(<View>)

Makes given View active.

- 1 or "G" for *Graphics View* and 2 or "D" for *Graphics View 2*
- -1 or "T" for *3D Graphics View*
- "A" for *Algebra View*
- "S" for *Spreadsheet View*
- "C" for *CAS View*

Notes: See also *ZoomIn* Command *ZoomIn*, *ZoomOut* Command *ZoomOut*, *Pan* Command *Pan*, *SetPerspective* Command *SetPerspective* commands.

SetAxesRatio Command

SetAxesRatio(<Number>, <Number>)

Changes the axes ratio of active Graphics View so that X units on *x-axis* correspond to the same number of pixels as Y units on *y-axis* and point (0,0) stays on its coordinates. If a unitary ratio is used, the related axis is fixed with the unit value, and the other one is adjusted as indicated.

Examples:

- *SetAxesRatio(1, 2)* fixes the *x-axis* and compresses the *y-axis*
- *SetAxesRatio(2, 1)* fixes the *y-axis* and shrinks the *x-axis*.

SetAxesRatio(<Number>, <Number>, <Number>)

Similar to above syntax, works with 3D Graphics View.

SetBackgroundColor Command

`SetBackgroundColor(<Object>, <Red>, <Green>, <Blue>)`

Changes the background color of given object. This is used for *Texts* and for objects in the *Spreadsheet*. The red, green and blue represent amount of corresponding color component, 0 being minimum and 1 maximum. Number t exceeding this interval is mapped to it using function .

`SetBackgroundColor(<Object>, <"Color">)`

Changes the background color of given object. This is used for *Texts* and for objects in the *Spreadsheet*. The color is entered as text, that may be:

- an English color name (see Reference:Colors). Some of them can be also used in national languages and are listed below.

Note: If you use this command in a GeoGebraScript, you must use the English color names

- an hexadecimal string of the type #AARRGGBB or #RRGGBB, where *AA* defines transparency (00 full transparency to FF full opacity), *RR* defines the red component, *GG* the green one and *BB* the blue one.

Example: `SetBackgroundColor(text1, "#80FF0000")` sets the background color of existing *text1* as Red, with a 50% transparency.

`SetBackgroundColor(<Red>, <Green>, <Blue>)`

Changes the background color of the active Graphics View

`SetBackgroundColor(<"Color">)`

Changes the background color of the active Graphics View

Note: If you use this command in a GeoGebraScript, you must use the English color names

- **Black**
- **Dark Gray**
- **Gray**
- **Dark Blue**
- **Blue**
- **Dark Green**
- **Green**
- **Maroon**
- **Crimson**
- **Red**
- **Magenta**
- **Indigo**
- **Purple**
- **Brown**
- **Orange**
- **Gold**
- **Lime**
- **Cyan**
- **Turquoise**
- **Light Blue**
- **Aqua**
- **Silver**
- **Light Gray**

- **Pink**
- **Violet**
- **Yellow**
- **Light Yellow**
- **Light Orange**
- **Light Violet**
- **Light Purple**
- **Light Green**
- **White**

SetCaption Command

`SetCaption(<Object>, <Text>)`

Changes the caption of the given object. *Text* must be enclosed in double quotes ".

SetColor Command

`SetColor(<Object>, <Red>, <Green>, <Blue>)`

Changes the color of given object. The red, green and blue represent amount of corresponding color component, 0 being minimum and 1 maximum. Number *t* exceeding this interval is mapped to it using function .

`SetColor(<Object>, <"Color">)`

Changes the color of given object. The color is entered as text, that may be:

- an English color name (see Reference:Colors). Some of them can be also used in national languages and are listed below.

Note: If you use this command in a GeoGebraScript, you must use the English color names

- an hexadecimal string of the type #AARRGGBB or #RRGGBB, where *AA* defines transparency (01 full transparency to FF full opacity), *RR* defines the red component, *GG* the green one and *BB* the blue one.

Example: `SetColor(text1, "#80FF0000")` sets the color of existing *text1* as red, with a 50% white transparency.

- **Black**
- **Dark Gray**
- **Gray**
- **Dark Blue**
- **Blue**
- **Dark Green**
- **Green**
- **Maroon**
- **Crimson**
- **Red**
- **Magenta**
- **Indigo**
- **Purple**
- **Brown**

- **Orange**
- **Gold**
- **Lime**
- **Cyan**
- **Turquoise**
- **Light Blue**
- **Aqua**
- **Silver**
- **Light Gray**
- **Pink**
- **Violet**
- **Yellow**
- **Light Yellow**
- **Light Orange**
- **Light Violet**
- **Light Purple**
- **Light Green**
- **White**

SetConditionToShowObject Command

SetConditionToShowObject(<Object>, <Condition>)

Sets the condition to show given object.

SetCoords Command

SetCoords(<Object>, <x>, <y>)

Changes cartesian coordinates of free objects. This command uses values of the coordinates, not their definitions, therefore the object stays free.

Note: This also works for Points on paths and in regions. The Point will be moved to the closest possible place

Note: This command now works for Sliders, Buttons, Checkboxes, Input Boxes, Images. If "Absolute Screen Position" is selected then x, y are in screen pixels.

SetDynamicColor Command

SetDynamicColor(<Object>, <Red>, <Green>, <Blue>)

Sets the dynamic color of the object.

SetDynamicColor(<Object>, <Red>, <Green>, <Blue>, <Opacity>)

Sets the dynamic color and opacity of the object.

Note: All numbers are on a scale from 0 (off/transparent) to 1 (on/opaque).

SetFilling Command

SetFilling(<Object>, <Number>)

Changes the opacity of given object. Number must be from interval [0,1], where 0 means transparent and 1 means 100% opaque. Other numbers are ignored.

SetFixed Command

SetFixed(<Object>, <true | false>)

Makes the object fixed (for true) or not fixed (for false).

SetFixed(<Object>, <true | false>, <true | false>)

Makes the object fixed (for true) or not fixed (for false) and the second parameter determines "Selection Allowed"

SetLabelMode Command

SetLabelMode(<Object>, <Number>)

Changes the label mode of the given object, according to the table below.

Notes:

- Integers distinct from the ones listed in table are treated as 0.
- The default option for the object's label is *Name*.
- For options 3 and 9, if the object's *Caption* box is empty, the *Name* of the object is used as caption.

Number	Mode
0	Name
1	Name + Value
2	Value
3	Caption
9	Caption + Value

SetLayer Command

SetLayer(<Object>, <Layer>)

Sets the layer for given object, where number of the layer must be an integer from {0, 1, ..., 9}.

SetLineStyle Command

SetLineStyle(<Line>, <Number>)

Changes the line style of given object according to following table (numbers out of range [0,4] are treated as 0).

Number	Style
0	Full
1	Dashed long
2	Dashed short
3	Dotted
4	Dash-dot

SetLineThickness Command

SetLineThickness(<Object>, <Number>)

Let N be the <Number>. The command SetLineThickness sets the line thickness for the given object to pixels.

SetPointSize Command

`SetPointSize(<Point>, <Number>)`

Changes the size of the point.

`SetPointSize(<Polygon>, <Number>)`

Changes the size of a polygon's points.

`SetPointSize(<Polyhedron>, <Number>)`

Changes the size of a polyhedron's points.

`SetPointSize(<Net>, <Number>)`

Changes the size of a net's points.

Notes: Also works for lists of (unlabeled) points, e.g. let list={(1, 2), (3, 4)}. Then `SetPointSize(list, 5)` changes the size of the listed points.

SetPointStyle Command

`SetPointStyle(<Point>, <Number>)`

Changes the point style of given point according to following table (numbers out of range [0,10] will be treated as 0).

Number	Style	Symbol
0	Full dot	
1	Cross	□
2	Empty dot	○
3	Plus sign	+
4	Full diamond	◆
5	Empty diamond	◇
6	Triangle north	▲
7	Triangle south	▼
8	Triangle east	▶
9	Triangle west	◀
10	Full dot (but with no outline)	

SetTooltipMode Command

`SetTooltipMode(<Object>, <Number>)`

Changes the tooltip mode for given object according to following table (values out of range [0,4] are treated as 0):

Number	Mode
0	Automatic
1	On
2	Off
3	Caption
4	Next cell

SetValue Command

`SetValue(<Boolean>, <0|1>)`

Sets the state of a boolean / check box : 1 = true, 0 = false

Example: If b is a boolean, `SetValue(b, 1)` sets the boolean b as *true*.

`SetValue(<Object>, <Object>)`

Let A be the first and B the second object. If A is a free object or a Point restricted to Path or Region, its value is set to current value of B (i.e. A doesn't change value if B is changed afterwards).

Example: If f is a function, `SetValue(f, RandomElement({cos(x), 3x+2, ln(x)}))` defines, at random, f as being one of the functions proposed in the list.

`SetValue(<List>, <Number>, <Object>)`

Let n be the *<Number>*. The command SetValue sets the n -th element of a free list to the current value of the object. Number n can be at most $1 + \text{length of L}$.

`SetValue(<Dependent Object>, ?)`

This is a special syntax that will set a dependent object to undefined without needing to fully redefine it using `=`.

`SetValue(<drop-down list>, <Number n >)`

Set n as the index of the selected element in the drop-down list.

SetVisibleInView Command

SetVisibleInView(<Object>, <View Number 1|2|-1>, <Boolean>)

Makes object visible or hidden in given Graphics View. Use -1 for the 3D View

Note: You can also use these special object names: **xAxis**, **yAxis**, **zAxis**, **xOyPlane**

SetViewDirection Command

SetViewDirection(<Direction>)

Sets the direction of the 3D view orientation depending on the given object.

Example: SetViewDirection(Vector((0, 0, 1)))

Example: SetViewDirection((0, 0, 1))

Example: SetViewDirection(x + y + z = 1)

SetViewDirection()

Sets the direction of the 3D view orientation to the default position.

Example: SetViewDirection()

SetViewDirection(<Direction>, <Boolean animated>)

Sets the direction of the 3D view orientation depending on the given object, with optional animation.

Example: In order to obtain the rotation of the 3D view, depending on the values of a previously created slider α , use the command SetViewDirection(Vector((1; α ; -30°)), false) in the *OnUpdate* scripting tab of slider α .

Notes: The view direction can be set towards a line, segment, plane, etc. If you do eg SetViewDirection(x + y + z = 1) twice then there are two possible outcomes and the second one will rotate the view 180°. To avoid ambiguity use eg SetViewDirection(Vector((0, 0, 1))) See also View in front of ToolView in front of tool.

SetSpinSpeed Command

`SetSpinSpeed(<Number>)`

Sets the rotational speed of *3D view* about the axis currently displayed in vertical position. The sign and value of the entered *Number* define the rotation as follows:

- if *Number* is *positive*, then the *3D view* rotates counter clockwise.
- if *Number* is *negative*, then the *3D view* rotates clockwise.
- if *Number* is 0 there will be no rotation of the *3D view*.

SetPerspective Command

`SetPerspective(<Text>)`

Changes the layout and visibility of *Views*. The text parameter is either the full description of the layout, description of a single view you want to change or ID of one of the standard perspectives.

Full layout description

To change the whole layout you can describe the view positions using an expression. *Views* are represented by variables (letters): the horizontal arrangement of *Views* is represented by the related letters juxtaposition, and their vertical arrangement by a division symbol /.

Letter	View
A	Algebra
B	Probability Calculator
C	CAS
D	Graphics 2
G	Graphics
L	Construction Protocol
P	Properties
R	Data analysis (Desktop only)
S	Spreadsheet
T	3D Graphics

Examples: `SetPerspective("G")` makes only the *Graphics View* visible `SetPerspective("AGS")` makes *Algebra*, *Graphics* and *Spreadsheet View* visible, aligned horizontally `SetPerspective("S/G")` makes *Spreadsheet* and *Graphics View* visible with *Spreadsheet* on top and *Graphics View* below `SetPerspective("S/(GA)")` is similar as above, the bottom part of the screen consists of *Graphics View* on the left and *Algebra View* on the right

Single view change

To open or close individual *Views*, add the symbols + or - before the *View* name (letter), respectively. In apps other than GeoGebra Classic (e.g. the Graphing Calculator) you can also use Tools and Table for the tools and table of values.

Examples: SetPerspective("+D") adds Graphics View 2 to the currently displayed ones, on the right SetPerspective("-D") removes Graphics View 2 from the currently displayed ones SetPerspective("+Tools") opens the sidebar in Graphing Calculator and switches it to tools tab SetPerspective("+Table") opens the sidebar in Graphing Calculator and switches it to table of values SetPerspective("-Tools") closes the sidebar in Graphing Calculator, no matter which tab is selected

Standard perspectives

You may also use a text containing a single digit to use a predefined perspective:

Text	Perspective
"1"	Algebra And Graphics
"2"	Geometry
"3"	Spreadsheet
"4"	CAS
"5"	3D Graphics
"6"	Probability

These roughly correspond to "AG", "G", "SG", "CG", "AT" and "B" respectively, but may also affect the display of Input Bar and content of Toolbar.

Notes: See also SetActiveView Command SetActiveView command.

SetSeed Command

`SetSeed(<Integer>)`

Seeds the random number generator so that subsequent random numbers will be determined by the seed.

Example: `SetSeed(33)`

Note: See also RandomBetween command, RandomElement command, RandomBinomial command, RandomNormal command, RandomPoisson command, RandomUniform command.

SetTrace Command

`SetTrace(<Object>, <true | false>)`

Turns Tracing on/off for the specified object.

Example: Create a point A, then type in `SetTrace(A, true)`. Select the Move Tool and drag the point, to show its trace.

Note: Use `ZoomIn(1)` to clear all traces.

ShowAxes Command

`ShowAxes()`

Shows the axes in the active View.

`ShowAxes(<Boolean>)`

Shows/hides the axes in the active View.

Example:

- `ShowAxes(true)` shows the axes in the active View.
- `ShowAxes(false)` hides the axes in the active View.

`ShowAxes(<View>, <Boolean>)`

Shows/hides the axes in the *Graphics View* specified by the number 1 or 2 (or 3 for *3D View*) .

Example:

- `ShowAxes(1, true)` shows the axes in *Graphics View*.
- `ShowAxes(2, false)` hides the axes in *Graphics 2 View*.

Note: See also ShowGrid command. To show / hide a single axis please use SetVisibleInView Command.

ShowGrid Command

ShowGrid()

Shows the grid in the active View.

ShowGrid(<Boolean>)

Shows/hides the grid in the active View.

Example:

- `ShowGrid(true)` shows the grid in the active View.
- `ShowGrid(false)` hides the grid in the active View.

ShowGrid(<View>, <Boolean>)

Shows/hides the grid in the *Graphics View* specified by the number 1 or 2 (or 3 for *3D View*).

Example:

- `ShowGrid(1, true)` shows the grid in *Graphics View*.
- `ShowGrid(2, false)` hides the grid in *Graphics 2 View*.

Note: See also ShowAxes command.

ShowLabel Command

ShowLabel(<Object>, <Boolean>)

Shows or hides the label in the Graphics View for the given object.

Example: Let $f(x) = x^2$. `ShowLabel(f, true)` shows the label of the function.

ShowLayer Command

ShowLayer(<Number>)

Makes all objects in given layer visible. Does not override Conditional Visibility.

Example: ShowLayer (2) makes all objects in the second layer visible.

Slider Command

Slider(<Min>, <Max>, <Increment>, <Speed>, <Width>,<Is Angle>, <Horizontal>, <Animating>, <Boolean Random>)

Creates a slider. The parameters settings can be as follows:

- *Min, Max*: set the range of the slider - These parameters are compulsory.
- *Increment*: set the increment of the slider's value - default: 0.1
- *Speed*: set the slider speed during animations - default: 1
- *Width*: sets the slider width in pixels - default: 100
- *Is Angle*: sets if the slider is related to an angle. This parameter can be *true* or *false* - default: *false*
- *Horizontal*: sets whether the slider is shown as an horizontal (*true*) or vertical (*false*) segment - default: *true*
- *Animating*: sets the automatic animation of the slider - default: *false*
- *Random*: sets if the slider assumes continuous values in the [Min, Max] range (*false*), or random values in the same interval (*true*) - default: *false*

Note: See also the Slider tool.

StartAnimation Command

StartAnimation()

Resumes all animations if they are paused.

StartAnimation(<Boolean>)

When the boolean is false, pauses all animations, otherwise resumes them.

StartAnimation(<Point or Slider>, <Point or Slider>,)

Starts animating given points and sliders, the points must be on paths.

StartAnimation(<Point or Slider>, <Point or Slider>,, <Boolean>)

Starts (for boolean = true) or permanently stops (for boolean = false) animating given points and sliders, the points must be on paths.

Note: See also Animation.

StartLogging Command

StartLogging Command

StopLogging Command

StopLogging Command

StartRecord Command

`StartRecord()`

Resumes all recording to spreadsheet if paused (and stores a value for each object).

`StartRecord(<Boolean>)`

When the boolean is false, pauses all recording to the spreadsheet, otherwise resumes it (and stores a value for each object).

InputBox Command

`InputBox()`

Create a new Input Box.

`InputBox(<Linked Object>)`

Create a new Input Box and associate a Linked Object with it.

Note: See also Input Box Tool.

UpdateConstruction Command

`UpdateConstruction()`

Recomputes all objects (random numbers are regenerated). Same as F9 or Ctrl + R.

Note: If you want to refresh the view (e.g. to remove traces from Graphics View) you can use `ZoomIn(1)` instead, which is the same as Ctrl + F. You may also need `SetActiveView(1)` or `SetActiveView(2)` first if you are using two Graphics Views.

`UpdateConstruction(<Number of times>)`

Performs the command `UpdateConstruction()` several times.

Example:

`UpdateConstruction(2)` updates the construction twice (e.g. to record several dice throws to the spreadsheet).

ZoomIn Command

ZoomIn()

Restores the Graphics View to the default initial position

ZoomIn(<Scale Factor>)

Zooms the Graphics View in by given factor with respect to current zoom, center of the screen is used as center point for the zoom.

Example:

`ZoomIn(1)` doesn't change the view, but does remove traces

`ZoomIn(2)` zooms the view in

`ZoomIn(0.5)` is equivalent to `ZoomOut(2)`, i.e. it zooms the view out.

ZoomIn(<Scale Factor>, <Center Point>)

Zooms the Graphics View in by given factor with respect to current zoom, second parameter specifies center point for the zoom.

Example:

`ZoomIn(2, (0, 0))`

ZoomIn(<Min x>, <Min y>, <Max x>, <Max y>)

Zooms the graphics view to the rectangle given by vertices (Min x, Min y), (Max x, Max y).

Example:

`ZoomIn(0, 1, 5, 6)`

Note: If any of these parameters are dependent or has label set, the bounds of the view become dynamic. To avoid this behavior, use `CopyFreeObject Command`.

Example:

If *a* is a slider, `ZoomIn(-a, -a, a, a)` makes the zoom of the view dependent on slider *a*.

ZoomIn(<Min x>, <Min y>, <Min z>, <Max x>, <Max y>, <Max z>)

Zooms the 3D graphics view to the cuboid given by vertices (Min x, Min y, Min z), (Max x, Max y, Max z).

Example:

`ZoomIn(-5, -5, -5, 5, 5, 5)`

Note: The dynamic behavior of the 2D version isn't supported

Notes: If multiple Graphics ViewGraphics Views are present, the active one is used. See also `ZoomOut Command`, `ZoomOut, SetActiveView Command`, `SetActiveView, Pan Command`, `Pan commands`.

ZoomOut Command

ZoomOut(<Scale Factor>)

Zooms the Graphics View out by given factor with respect to current zoom, center of the screen is used as center point for the zoom.

Example: `ZoomOut (2)` zooms the view out.

ZoomOut(<Scale Factor>, <Center Point>)

Zooms the Graphics View out by given factor with respect to current zoom, second parameter specifies center point for the zoom.

Example: `ZoomOut (2, (0, 0))`

Notes: `ZoomOut(t)` and `ZoomOut(t, A)` are equivalent to `ZoomIn(1/t)` and `ZoomIn(1/t, A)` respectively. If multiple Graphics Views are present, the active one is used. See also `ZoomIn` Command, `ZoomIn`, `SetActiveView` Command, `SetActiveView`, `Pan` Command, `Pan` commands.

Discrete Math Commands

<links/>

ConvexHull Command

ConvexHull(<List of Points>)

Creates convex hull of given set of points. Returned object is a locus, so it is auxiliary.

DelaunayTriangulation Command

DelaunayTriangulation(<List of Points>)

Creates a Delaunay Triangulation of the list of points. Returned object is a locus, so it is auxiliary.

Hull Command

Hull Command

MinimumSpanningTree Command

MinimumSpanningTree(<List of Points>)

Returns the minimum spanning tree of a complete graph on given vertices in which weight of edge (u,v) is the Euclidian distance between u and v . The resulting object is a locus.

ShortestDistance Command

ShortestDistance(<List of Segments>, <Start Point>, <End Point>, <Boolean Weighted>)

Finds shortest path between start point and endpoint in a graph given by list of segments. If weighted is false, weight of each edge is supposed to be 1 (i.e. we are looking for the path with least number of edges), otherwise it is the length of given segment (we are looking for the geometrically shortest path).

TravelingSalesman Command

TravelingSalesman(<List of Points>)

Returns the shortest closed path which goes through each point exactly once. Returned object is a locus, so it is auxiliary.

Voronoi Command

Voronoi(<List of Points>)

Draws the Voronoi diagram for given list of points. Returned object is a locus, so it is auxiliary.

GeoGebra Commands

<links/>

AxisStepX Command

AxisStepX()

Returns the current step width for the *x*-axis.

Note: See also AxisStepY Command AxisStepY command. Together with the Corner Command Corner and Sequence Command Sequence commands, the AxisStepX and AxisStepY commands allow you to create custom axes (also see section Customizing the Graphics View#Customizing Coordinate Axes and Grid Customizing Coordinate Axes and Grid).

AxisStepY Command

AxisStepY()

Returns the current step width for the y-axis.

Note: See also AxisStepX CommandAxisStepX command. Together with the Corner CommandCorner and Sequence CommandSequence commands, the AxisStepX and AxisStepY commands allow you to create custom axes (also see section Customizing the Graphics View#Customizing Coordinate Axes and GridCustomizing Coordinate Axes and Grid).

ClosestPointRegion Command

ClosestPointRegion(<Region>, <Point>)

Returns a new point on the region which is the closest to a selected point.

ConstructionStep Command

ConstructionStep()

Returns the current ([Construction Protocol]) step as a number.

ConstructionStep(<Object>)

Returns the ([Construction Protocol]) step for the given object as a number.

Corner Command

Corner(<Number of Corner>)

For number $n = 1, 2, 3, 4$ creates a point at the corner of the Graphics View, for $n = 5$ returns point (w, h) , where w and h are width and height of the Graphics View in pixels. Always uses first Graphics View, even if second is active.

Note: `Corner(<Number of Corner>)` won't work inside other commands. Instead create eg `C_1 = Corner(1)` and use that.

Corner(<Graphics View>, <Number of Corner>)

Creates a point at the corner of Graphics View (1, 2) which is never visible in that view. Supported values of number n are $1, 2, 3, 4$ and 5 as above.

Note: `Corner(<Graphics View>, <Number of Corner>)` won't work inside other commands. Instead create eg `C_1 = Corner(1, 1)` and use that.

Note: use -1 for the 3D Graphics View's corners (available values for *Number*: from 1 to 8); for $n = 9$ returns point $(w, h, 0)$, where w and h are width and height of the Graphics View in pixels; for $n = 10$ returns point $(w, h, 0)$, where w and h are width and height of the main window in pixels; for $n = 11$ returns view direction (for parallel projections) or eye position (for e.g. perspective projection); for $n = 12$ returns screen left-to-right direction; for $n = 13$ returns scales for x, y, z axes.

Corner(<Image>, <Number of Corner>)

Creates a point at the corner of the image (number $n = 1, 2, 3, 4$).

Corner(<Text>, <Number of Corner>)

Creates a point at the corner of the text (number $n = 1, 2, 3, 4$).

Note: `Corner(<Text>, <Number of Corner>)` won't work inside the Sequence or Zip commands. Also the *Absolute Position on Screen* property must be unchecked

Note: The numbering of the corners is counter-clockwise and starts at the lower left corner.

DynamicCoordinates Command

`DynamicCoordinates(<Point>, <x-Coordinate>, <y-Coordinate>)`

Creates a new point with given coordinates: this point is dependent, but can be moved. Whenever you try to move the new point to coordinates (x, y), the given point is moved there and coordinates for the new point are calculated. Works best if the given point is not visible and dragging is done with the mouse. At least one of the given coordinates should depend on the given point.

Examples:

- Let A be a point and B = `DynamicCoordinates(A, round(x(A)), round(y(A)))`. When you try to move B to (1.3, 2.1) using the Move Tool, point A becomes (1.3, 2.1) and B appears at (1,2).
- B = `DynamicCoordinates(A, x(A), min(y(A), sin(x(A))))` creates a point under $\sin(x)$.

Note: `PointIn(y < sin(x))` is the easier solution in this case.

The following examples show other ways to restrain the positions of a point C:

- Let A = `Point(xAxis)` and B = `Point(xAxis)`.

Now type in the Input Bar:

`DynamicCoordinates(B, Min(x(B), x(A)), 0)` and press Enter

`SetVisibleInView(B, 1, false)` and press Enter

`SetLayer(C, 1)` and press Enter

Now, C cannot be moved to the right of A.

- Define A=(1, 2).

Now, type in the Input Bar:

`SetVisibleInView(A, 1, false)` and press Enter

`B = DynamicCoordinates(A, If(x(A) > 3, 3, If(x(A) < -3, -3, If(x(A) < 0, round(x(A)), x(A)))), If(x(A) < 0, 0.5, If(y(A) > 2, 2, If(y(A) < 0, 0, y(A)))))` and press Enter

- This example makes A a sticky point when a point C is dragged near it. Define A = (1, 2) and B = (2, 3).

Now, type in the Input Bar:

`SetVisibleInView(B, 1, false)` and press Enter

`C = DynamicCoordinates(B, If(Distance(A, B) < 1, x(A), x(B)), If(Distance(A, B) < 1, y(A), y(B))).`

`DynamicCoordinates(<Point>, <x-Coordinate>, <y-Coordinate>, <z-Coordinate>)`

Creates a new 3D point with given coordinates: this point is dependent, but can be moved. Whenever you try to move the new point to coordinates (x, y, z), the given point is moved there and coordinates for the new point are calculated. Works best if the given point is not visible and dragging is done with the mouse. At least one of the given coordinates should depend on the given point.

Name Command

`Name(<Object>)`

Returns the name of an object as a text in the Graphics View.

Notes: This command works properly only in dynamic text for objects (so that they work after objects are renamed). The Name command is the opposite of the Object CommandObject command.

Object Command

`Object(<Name of Object as Text>)`

Returns the object for a given name. The result is always a dependent object.

Note: The **Object** command is the opposite of the Name command.

Example: If points $A1, A2, \dots, A20$ exist and also a slider $n = 2$, then `Object ("A" + n)` creates a copy of point $A2$.

Note: You must make sure that the objects you refer to are **earlier** in the Construction_Protocol than this command

Warning: Object command cannot be used in Custom Tools

SlowPlot Command

`SlowPlot(<Function>)`

Creates an animated graph of the given function, that is plotted from left to right. The animation is controlled by a slider, which is also created by this command.

`SlowPlot(<Function>, <Boolean Repeat>)`

Creates an animated graph of the given function, that is plotted from left to right. The animation is controlled by a slider, which is also created by this command: if *Repeat* is *false*, the graph is plotted only once - corresponding to the slider setting *Increasing (once)*, if *Repeat* is *true* or omitted, the graph is plotted continuously - corresponding to the slider setting *Increasing*.

ToolImage Command

ToolImage(<Number>)

Creates in the *Graphics View* a 32x32 pixel image of the ([:Category:IconsToolBar|tool icon]) with given number.

ToolImage(<Number>, <Point>)

Creates in the *Graphics view* a 32x32 pixel image of the tool icon, anchored to the given point.

ToolImage(<Number>, <Point>, <Point>)

Creates in the *Graphics view* an image of the tool icon. The two given points define two adjacent vertices of the side of the oriented square containing the image.

Note: See Reference:Toolbar for the icons numbering, or ToolsEN.

Optimization Commands

<links/>

Maximize Command

Maximize(<Dependent number>, <Free number>)

Calculates the free number which gives the maximal value of the dependent number. The free number must be a slider and the slider interval will be used as the search interval. The relationship should be continuous and have only one *local* maximum point in the interval. If the construction is complicated, this command might return ? to avoid using too much processor time.

Maximize(<Dependent Number>, <Point on Path>)

Note: See also Minimize command.

Minimize Command

Minimize(<Dependent number>, <Free number>)

Calculates the free number which gives the minimal value of the dependent number. The free number must be a slider and the slider interval will be used as the search interval. The relationship should be continuous and have only one *local* minimum in the interval. If the construction is complicated, this command might return ? to avoid using too much processor time.

Minimize(<Dependent Number>, <Point on Path>)

Note: See also Maximize command.

CAS Specific Commands

All of the following commands can be used in the CAS View.

- BinomialCoefficient
- BinomialDist
- CFactor
- CSolutions
- CSolve
- Cauchy
- ChiSquared
- Coefficients
- CommonDenominator
- Covariance
- Cross
- Degree
- Delete
- Denominator
- Derivative
- Determinant
- Dimension
- Div
- Division
- Divisors
- DivisorsList
- DivisorsSum
- Dot
- Element
- Eliminate
- Expand
- Exponential
- FDistribution
- Factor
- Factors
- First
- FitExp

- FitLog
 - FitPoly
 - FitPow
 - FitSin
 - GCD
 - Gamma
 - GroebnerDegRevLex
 - GroebnerLex
 - GroebnerLexDeg
 - HyperGeometric
 - Identity
 - ImplicitDerivative
 - Integral
 - IntegralBetween
 - Intersect
 - Invert
 - IsPrime
 - Last
 - LCM
 - LeftSide
 - Length
 - Limit
 - LimitAbove
 - LimitBelow
 - Max
 - Mean
 - Median
 - Min
 - MixedNumber
 - Mod
 - NIntegral
 - nPr
 - NSolutions
 - NSolve
 - NextPrime
 - Normal
 - Numerator
 - Numeric
 - PartialFractions
 - Pascal
 - PerpendicularVector
 - Poisson
 - PreviousPrime
 - PrimeFactors
 - Product
 - RandomBetween
 - RandomBinomial
-

- RandomElement
- RandomNormal
- RandomPoisson
- RandomPolynomial
- Rationalize
- ReducedRowEchelonForm
- RightSide
- Root
- SD
- Sample
- SampleSD
- SampleVariance
- Sequence
- Shuffle
- Simplify
- Solutions
- Solve
- SolveODE
- Substitute
- Sum
- TDistribution
- Take
- TaylorPolynomial
- ToComplex
- ToExponential
- ToPoint
- ToPolar
- Transpose
- Unique
- UnitPerpendicularVector
- UnitVector
- Variance
- Weibull
- Zipf

CFactor Command

CAS Syntax

`CFactor(<Expression>)`

Factorizes a given expression, allowing for complex factors.

Example: `CFactor(x^2 + 4)` yields $(x + 2i)(x - 2i)$, the factorization of $x^2 + 4$.

`CFactor(<Expression>, <Variable>)`

Factorizes an expression with respect to a given variable, allowing for complex factors.

Examples:

- `CFactor(a^2 + x^2, a)` yields $(ix + a)(-ix + a)$, the factorization of $a^2 + x^2$ with respect to a .
- `CFactor(a^2 + x^2, x)` yields $(x + ia)(x - ia)$, the factorization of $a^2 + x^2$ with respect to x .

Note: This command factors expressions over the Complex Rational Numbers. To factor over rational numbers, see the Factor Command.

CIFactor Command

CAS Syntax

`CIFactor(<Expression>)`

Factors over the complex irrationals.

Example:

`CIFactor[x^2 + x + 1]` returns

`CIFactor(<Expression>, <Variable>)`

Factors over the complex irrationals with respect to a given variable.

Example:

`CIFactor[a^2 + a + 1, a]` returns

Note: See also IFactor command.

CSolutions Command

CAS Syntax

`CSolutions(<Equation>)`

Solves a given equation for the main variable and returns a list of all solutions, allowing for complex solutions.

Example: `CSolutions(x^2 = -1)` yields $\{i, -i\}$, the complex solutions of $x^2 = -1$.

`CSolutions(<Equation>, <Variable>)`

Solves an equation for a given unknown variable and returns a list of all solutions, allowing for complex solutions.

Example: `CSolutions(a^2 = -1, a)` yields $\{i, -i\}$, the complex solutions of $a^2 = -1$.

`CSolutions(<List of Equations>, <List of Variables>)`

Solves a set of equations for a given set of unknown variables and returns a list of all solutions, allowing for complex solutions.

Example: `CSolutions({y^2 = x - 1, x = 2 * y - 1}, {x, y})` yields , the complex solutions of $y^2 = x - 1$ and $x = 2 * y - 1$.

Notes:

- The complex i is obtained by pressing ALT + i.
- See also CSolve Command and Solutions Command.

CSolve Command

CAS Syntax

`CSolve(<Equation>)`

Solves a given equation for the main variable and returns a list of all solutions, allowing for complex solutions.

Example: `CSolve(x^2 = -1)` yields $\{x = i, x = -i\}$, the complex solutions of $x^2 = -1$.

`CSolve(<Equation>, <Variable>)`

Solves an equation for a given unknown variable and returns a list of all solutions, allowing for complex solutions.

Example: `CSolve(a^2 = -1, a)` yields $\{a = i, a = -i\}$, the complex solutions of $a^2 = -1$.

`CSolve(<List of Equations>, <List of Variables>)`

Solves a set of equations for a given set of unknown variables and returns a list of all solutions, allowing for complex solutions.

Example: `CSolve({y^2 = x - 1, x = 2 * y - 1}, {x, y})` yields $\{x = 1 - 2i, y = 1 + i\}$, $\{x = 1 + 2i, y = 1 - i\}$, the complex solutions of $y^2 = x$ and $x = 2 * y - 1$.

Notes:

- The complex i is obtained by pressing ALT + i.
- See also CSolutions Command and Solve Command.

CommonDenominator Command

`CommonDenominator(<Expression>, <Expression>)`

Returns the function having as equation the lowest common denominator of the two expressions.

Example: `CommonDenominator(3 / (2 x + 1), 3 / (4 x^2 + 4 x + 1))` yields $f(x) = \frac{4x^2 + 4x + 1}{x^2 + 4x + 1}$.

CAS Syntax

`CommonDenominator(<Expression>, <Expression>)`

Returns the lowest common denominator of the two expressions.

Example: `CommonDenominator(3 / (2 x + 1), 3 / (4 x^2 + 4 x + 1))` yields $4x^2 + 4x + 1$.

Cross Command

`Cross(<Vector u> , <Vector v>)`

Calculates the cross product of u and v . Instead of vectors you can also use lists.

Example:

`Cross((1, 3, 2), (0, 3, -2))` yields $(-12, 2, 3)$, `Cross({1, 3, 2}, {0, 3, -2})` yields $\{-12, 2, 3\}$

For 2D vectors or points the result is the z-coordinate of the actual cross product.

Example: `Cross((1, 2), (4, 5))` yields -3 .

Hint: If a vector in the CAS View contains undefined variables, the command yields a formula for the cross product, e.g. `Cross((a, b, c), (d, e, f))` yields $(bf - ce, af - cd, ae - bd)$.

Notes: You can also use the `Predefined_Functions_and_Operators` operator $u \otimes v$ See also Dot Command.

Dimension Command

Dimension(<Object>)

Gives the dimension of a vector or a Matrix.

Example:

`Dimension({1, 2, 0, -4, 3})` yields 5.

Example:

`Dimension({{1, 2}, {3, 4}, {5, 6}})` yields {3, 2}.

CAS Syntax

Dimension(<Object>)

Gives the dimension of a vector or matrix.

Example:

`Dimension({1, 2, 0, -4, 3})` yields 5.

Example:

`Dimension({{a, b}, {c, d}, {e, f}})` yields {3, 2}.

Division Command

Division(<Dividend Number>, <Divisor Number>)

Gives the quotient (integer part of the result) and the remainder of the division of the two numbers.

Example:

`Division(16, 3)` yields {5, 1}.

Division(<Dividend Polynomial>, <Divisor Polynomial>)

Gives the quotient and the remainder of the division of the two polynomials.

Example:

`Division(x^2 + 3 x + 1, x - 1)` yields {x + 4, 5}.

Note: In the *Algebra View* only one variable can be used and it will always be renamed to *x*. In the *CAS View* multivariable division is also supported. Examples:
`Division(x^2+y^2, x+y)` yields {x - y, 2y^2}.
`Division(x^2+y^2, y+x)` yields {y - x, 2x^2}.

Divisors Command

`Divisors(<Number>)`

Calculates the number of all the positive divisors, including the number itself.

Example: `Divisors(15)` yields 4, the number of all positive divisors of 15, including 15.

CAS Syntax

`Divisors(<Number>)`

Calculates the number of all the positive divisors, including the number itself.

Example: `Divisors(15)` yields 4, the number of all positive divisors of 15, including 15.

Note: See also DivisorsList Command and DivisorsSum Command.

DivisorsList Command

`DivisorsList(<Number>)`

Gives the list of all the positive divisors, including the number itself.

Example: `DivisorsList(15)` yields {1, 3, 5, 15}, the list of all positive divisors of 15, including 15.

CAS Syntax

`DivisorsList(<Number>)`

Gives the list of all the positive divisors, including the number itself.

Example: `DivisorsList(15)` yields {1, 3, 5, 15}, the list of all positive divisors of 15, including 15.

Note: See also Divisors Command and DivisorsSum Command.

DivisorsSum Command

DivisorsSum(<Number>)

Calculates the sum of all the positive divisors, including the number itself.

Example: DivisorsSum(15) yields 24, the sum $1 + 3 + 5 + 15$.

CAS Syntax

DivisorsSum(<Number>)

Calculates the sum of all the positive divisors, including the number itself.

Example: DivisorsSum(15) yields 24, the sum $1 + 3 + 5 + 15$.

Note: See also Divisors Command and DivisorsList Command.

Dot Command

Dot(<Vector>, <Vector>)

Returns the dot product (scalar product) of the two vectors.

Example:

Dot({1, 3, 2}, {0, 3, -2}) yields 5, the scalar product of {1, 3, 2} and {0, 3, -2}.

Note:

See also Cross Command.

Eliminate Command

CAS Syntax

Eliminate(<List of Polynomials>, <List of Variables>)

Considers the algebraic equation system defined by the polynomials, and computes an equivalent system after eliminating all variables in the given list.

Example: Eliminate({ $x^2 + x$, $y^2 - x$ }, { x }) yields {}.

Note: See also GroebnerLexDeg command.

GroebnerDegRevLex Command

CAS Syntax

GroebnerDegRevLex(<List of Polynomials>)

Computes the Gröbner basis of the list of the polynomials with respect to graded reverse lexicographical ordering of the variables (also known as *total degree reverse lexicographic ordering*, *degrevlex* or *grevlex* ordering).

Example: GroebnerDegRevLex({ $x^3 - y - 2$, $x^2 + y + 1$ }) yields {}.

GroebnerDegRevLex(<List of Polynomials>, <List of Variables>)

Computes the Gröbner basis of the list of the polynomials with respect to graded reverse lexicographical ordering of the given variables (also known as *total degree reverse lexicographic ordering*, *degrevlex* or *grevlex* ordering).

Example: GroebnerDegRevLex({ $x^3 - y - 2$, $x^2 + y + 1$ }, { y , x }) yields {}.

Note: See also GroebnerLex and GroebnerLexDeg commands.

GroebnerLex Command

CAS Syntax

`GroebnerLex(<List of Polynomials>)`

Computes the Gröbner basis of the list of the polynomials with respect to lexicographical ordering of the variables (also known as *lex*, *plex* or *pure lexical ordering*).

Example: `GroebnerLex({x^3-y-2, x^2+y+1})` yields `{}`.

`GroebnerLex(<List of Polynomials>, <List of Variables>)`

Computes the Gröbner basis of the list of the polynomials with respect to lexicographical ordering of the given variables (also known as *lex*, *plex* or *pure lexical ordering*).

Example: `GroebnerLex({x^3-y-2, x^2+y+1}, {y, x})` yields `{}`.

Note: See also `GroebnerDegRevLex` and `GroebnerLexDeg` commands.

GroebnerLexDeg Command

CAS Syntax

`GroebnerLexDeg(<List of Polynomials>)`

Computes the Gröbner basis of the list of the polynomials with respect to graded lexicographical ordering of the variables (also known as *grlex*, *tdeg*, *lexdeg*, *total degree lexicographic ordering* or *elimination ordering*).

Example: `GroebnerLexDeg({x^3 - y - 2, x^2 + y + 1})` yields `{}`.

`GroebnerLexDeg(<List of Polynomials>, <List of Variables>)`

Computes the Gröbner basis of the list of the polynomials with respect to graded lexicographical ordering of the given variables (also known as *grlex*, *tdeg*, *lexdeg*, *total degree lexicographic ordering* or *elimination ordering*).

Example: `GroebnerLexDeg({x^3 - y - 2, x^2 + y + 1}, {y, x})` yields `{}`.

Note: See also `GroebnerDegRevLex` and `GroebnerLex` commands.

ImplicitDerivative Command

`ImplicitDerivative(<f(x, y)>)`

Gives the implicit derivative of the given expression.

Example:

`ImplicitDerivative(x + 2 y)` yields -0.5 .

CAS Syntax

`ImplicitDerivative(<f(x, y)>)`

Gives the implicit derivative of the given expression.

Example:

`ImplicitDerivative(x + 2 y)` yields $-$.

`ImplicitDerivative(<Expression>, <Dependent Variable>, <Independent Variable>)`

Gives the implicit derivative of the given expression.

Example:

`ImplicitDerivative(x^2 + y^2, y, x)` yields $-$.

Note:

See also Derivative Command.

InverseLaplace Command

CAS Syntax

`InverseLaplace(<Function>)`

Returns the inverse Laplace transform of the given function.

Example: `InverseLaplace[1/(1+t^2)]` returns $\sin(t)$.

`InverseLaplace(<Function>, <Variable>)`

Returns the inverse Laplace transform of the function, with respect to the given variable.

Examples:

`InverseLaplace[exp(- a*b), a]` returns $\frac{1}{b} \sin(bt)$

`InverseLaplace[exp(- a*b), b]` returns $\frac{1}{a} \cos(at)$

Note: See also Laplace command.

IsPrime Command

IsPrime(<Number>)

Gives *true* or *false* depending on whether the number is prime or not.

Example:

- IsPrime(10) yields *false*,
- IsPrime(11) yields *true*.

CAS Syntax

IsPrime(<Number>)

Gives *true* or *false* depending on whether the number is prime or not.

Example:

- IsPrime(10) yields *false*,
- IsPrime(11) yields *true*.

Laplace Command

CAS Syntax

Laplace(<Function>)

Returns the Laplace transform of the given function

Example: Laplace[sin(t)] returns

Laplace(<Function>, <Variable>)

Returns the Laplace transform of the function, with respect to the given variable.

Examples:

Laplace[sin(a*t),t] returns

Laplace[sin(a*t),a] returns

Note: See also InverseLaplace command.

LeftSide Command

`LeftSide(<Equation>)`

Gives the left-hand side of the simplified equation.

Example:

`LeftSide(4x = 1 - 3y)` yields $4x + 3y$.

CAS Syntax

`LeftSide(<Equation>)`

Gives the left-hand side of the equation.

Example:

`LeftSide(x + 2 = 3 x + 1)` yields $x + 2$.

`LeftSide(<List of Equations>)`

Gives the list of the left-hand sides of the equations.

Example:

`LeftSide({a^2 + b^2 = c^2, x + 2 = 3 x + 1})` yields .

`LeftSide(<List of Equations>, <Index>)`

Gives the left-hand side of the equation specified by the index.

Example:

`LeftSide({a^2 + b^2 = c^2, x + 2 = 3 x + 1}, 1)` yields .

Note:

See also RightSide Command.

MatrixRank Command

MatrixRank(<Matrix>)

Returns the rank of given matrix.

Examples:

- `MatrixRank({{2, 2}, {1, 1}})` yields 1.
- `MatrixRank({{1, 2}, {3, 4}})` yields 2.
- Let $A = \begin{Bmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \\ 2 & 2 & 2 \end{Bmatrix}$ be a 3x3-matrix. `MatrixRank(A)` yields 2.

Hint: In the CAS View this command also works with undefined variables. Example: `MatrixRank({{1, 2}, {k*1, k*2}})` yields 1.

MixedNumber Command

CAS Syntax

MixedNumber(<Number>)

Converts the given number to a mixed number.

Examples:

- `MixedNumber(3.5)` yields 3.
- `MixedNumber(12 / 3)` yields 4.
- `MixedNumber(12 / 14)` yields 0.

Note:

See also Rationalize Command.

NIntegral Command

`NIntegral(<Function>, <Start x-Value>, <End x-Value>)`

Computes (numerically) the definite integral of the given function f , from a (*Start x-Value*) to b (*End x-Value*).

Example: `NIntegral(e^{-x^2} , 0, 1)` yields 0.75.

`NIntegral(<Function>, <Start x-Value>, <Start y-Value>, <End x-Value>)`

Computes (numerically) the indefinite integral of the given function, and plots the graph of that function through (*Start x-Value*, *Start y-Value*), with end point at (*End x-Value*).

Example: `NIntegral($\sin(x)/x$, π , 1, 2π)` plots the graph of the indefinite integral of the given function in the interval $[\pi, 2\pi]$. The value of c is defined by the initial condition (*start x-Value*, *start y-Value*)=(π , 1).

Hint: In the CAS View the following syntax can also be used:
`NIntegral(<Function>, <Variable>, <Start Value>, <End Value>)` Computes (numerically) the definite integral $\int_a^b f(t) dt$ of the given function f , from a (*Start value*) to b (*End value*), with respect to the given variable. Example: `NIntegral(e^{-a^2} , a, 0, 1)` yields 0.75.

NSolutions Command

The following commands are only available in the CAS View.

`NSolutions(<Equation>)`

Attempts (numerically) to find a solution for the equation for the main variable. For non-polynomials you should always specify a starting value (see below)

Example:

`NSolutions($x^6 - 2x + 1 = 0$)` yields {0.51, 1} or {0.508660391642, 1} (the number of decimals depends on the chosen global rounding)

`NSolutions(<Equation>, <Variable>)`

Attempts (numerically) to find a solution of the equation for the given unknown variable. For non-polynomials you should always specify a starting value (see below)

Example:

`NSolutions($a^4 + 34a^3 = 34$, a)` yields { $a = -34.00086498588374$, $a = 0.9904738885574178$ }.

`NSolutions(<Equation>, <Variable = starting value>)`

Finds numerically the list of solutions to the given equation for the given unknown variable with its starting value.

Examples:

- `NSolutions($\cos(x) = x$, x = 0)` yields {0.74}
- `NSolutions($a^4 + 34a^3 = 34$, a = 3)` yields the list {0.99}.

`NSolutions(<List of Equations>, <List of Variables>)`

Attempts (numerically) to find a solution of the set of equations for the given set of unknown variables.

Example:

`NSolutions({ $\pi / x = \cos(x - 2y)$, $2y - \pi = \sin(x)$ }, {x = 3, y = 1.5})`
yields the list {3.14, 1.57}

Note: If you don't give a starting point like $a=3$ or $\{x = 3, y = 1.5\}$ the numerical algorithm may find it hard to find a solution (and giving a starting point doesn't guarantee that a solution will be found). The number of decimals depends

on the choosen in Options Menu#Rounding global rounding. NSolutions won't work for functions that are asymptotic to the x-axis. They can often be reformulated though. NSolutions will work only if the function is continuous See also Solutions Command and NSolve Command.

NSolve Command

CAS Syntax

This command is only available in the CAS View.

NSolve(<Equation>)

Attempts (numerically) to find a solution for the equation for the main variable. For non-polynomials you should always specify a starting value (see below).

Example:

`NSolve(x^6 - 2x + 1 = 0)` yields $\{x = 0.51, x = 1\}$.

NSolve(<Equation>, <Variable>)

Attempts (numerically) to find a solution of the equation for the given unknown variable. For non-polynomials you should always specify a starting value (see below).

Example:

`NSolve(a^4 + 34a^3 = 34, a)` yields $\{a = -34, a = 0.99\}$.

NSolve(<Equation>, <Variable = starting value>)

Finds numerically the list of solutions to the given equation for the given unknown variable with its starting value.

Examples:

- `NSolve(cos(x) = x, x = 0)` yields $\{x = 0.74\}$
- `NSolve(a^4 + 34a^3 = 34, a = 3)` yields $\{a = 0.99\}$.

NSolve(<List of Equations>, <List of Variables>)

Attempts (numerically) to find a solution of the set of equations for the given set of unknown variables.

Example:

`NSolve({pi / x = cos(x - 2y), 2 y - pi = sin(x)}, {x = 3, y = 1.5})` yields $\{x = 3.14, y = 1.57\}$.

Note:

- If you don't give a starting point like $a=3$ or $\{x = 3, y = 1.5\}$ the numerical algorithm may find it hard to find a solution (and giving a starting point doesn't guarantee that a solution will be found)
- The number of decimals depends on the choosen in global rounding.
- NSolve won't work for functions that are asymptotic to the x-axis or other extreme examples. They can often be reformulated though.
- NSolve will work only if the function is continuous!
- See also Solve Command and NSolutions Command.

NextPrime Command

`NextPrime(<Number>)`

Returns the smallest prime greater than the entered number.

Example:

`NextPrime(10000)` yields *10007*.

Note:

See also PreviousPrime Command.

Numeric Command

CAS Syntax

`Numeric(<Expression>)`

Tries to determine a numerical approximation of the given expression. The number of decimals depends on the global rounding you choose in the Options Menu.

Example: `Numeric(3 / 2)` yields *1.5*.

`Numeric(<Expression>, <Significant Figures>)`

Tries to determine a numerical approximation of the given expression, using the entered number of significant figures.

Example: `Numeric(sin(1), 20)` yields *0.84147098480789650665*.

Note: If you don't specify enough digits then you can get an apparently wrong answer due to floating point cancellation
^[1]. Example:

`Numeric(-500000000/785398163*sin(785398163/500000000)*1258025227.19^2+500000000/785398163*1258025227.19^2,10)`
will give 4096 but

`Numeric(-500000000/785398163*sin(785398163/500000000)*1258025227.19^2+500000000/785398163*1258025227.19^2,30)`
will give 0.318309886345536696694580314215.

Note: See also Numeric tool.

References

[1] http://docs.oracle.com/cd/E19957-01/806-3568/ngc_goldberg.html

PreviousPrime Command

`PreviousPrime(<Number>)`

Returns the greatest prime smaller than the entered number.

Example:

`PreviousPrime(10000)` yields 9973.

Note:

See also NextPrime Command.

RandomPolynomial Command

`RandomPolynomial(<Degree> , <Minimum for Coefficients> , <Maximum for Coefficients>)`

Returns a randomly generated polynomial in x of degree d , whose (integer) coefficients are in the range from *minimum* to *maximum*, both included.

Examples:

- `RandomPolynomial(0, 1, 2)` yields either 1 or 2.
- `RandomPolynomial(2, 1, 2)` yields a random polynomial with a degree of two and only 1 and 2 as coefficients, for example $2x^2 + x + 1$.

CAS Syntax

The following command is only available in the CAS View.

`RandomPolynomial(<Variable> , <Degree> , <Minimum for Coefficients> , <Maximum for Coefficients>)`

Returns a randomly generated polynomial in *Variable* of degree d , whose (integer) coefficients are in the range from *minimum* to *maximum*, both included.

Examples:

- `RandomPolynomial(a, 0, 1, 2)` yields either 1 or 2.
- `RandomPolynomial(a, 2, 1, 2)` yields a random polynomial with a degree of two and only 1 and 2 as coefficients, for example $2a^2 + a + 1$.

Note: In both cases if *minimum* or *maximum* are not integers, `round(minimum)` and `round(maximum)` are used instead.

Rationalize Command

CAS Syntax

`Rationalize(<Number>)`

Creates the fraction of the given *Number*, and rationalizes the denominator, if it contains square roots.

Examples:

- `Rationalize(3.5)` yields .
- `Rationalize(1/sqrt(2))` yields .

Note: See also MixedNumber Command.

RightSide Command

`RightSide(<Equation>)`

Gives the right-hand side of the simplified equation.

Example:

`RightSide(x + 2 = 3x + 1)` yields 0.5

CAS Syntax

`RightSide(<Equation>)`

Gives the right-hand side of the equation.

Example:

`RightSide(x + 3 = 3 x + 1)` yields $3x + 1$.

`RightSide(<List of Equations>)`

Gives the list of the right-hand sides of the equations.

Example:

`RightSide({a^2 + b^2 = c^2, x + 2 = 3x + 1})` yields { c^2 , $3x + 1$ }.

`RightSide(<List of Equations>, <Index>)`

Gives the right-hand sides of the equation specified by the index.

Example:

`RightSide({a^2 + b^2 = c^2, x + 2 = 3 x + 1}, 1)` yields .

Note:

See also LeftSide Command.

Solutions Command

CAS Syntax

Solutions(<Equation>)

Solves a given equation for the main variable and returns a list of all solutions.

Example: `Solutions(x^2 = 4x)` yields $\{0, 4\}$, the solutions of $x^2 = 4x$.

Solutions(<Equation>, <Variable>)

Solves an equation for a given unknown variable and returns a list of all solutions.

Example: `Solutions(x * a^2 = 4a, a)` yields $\{\}$, the solutions of $xa^2 = 4a$.

Solutions(<List of Equations>, <List of Variables>)

Solves a set of equations for a given set of unknown variables and returns a list of all solutions.

Examples:

- `Solutions({x = 4 x + y, y + x = 2}, {x, y})` yields $\{-1, 3\}$, the sole solution of $x = 4x + y$ and $y + x = 2$, displayed as .
- `Solutions({2a^2 + 5a + 3 = b, a + b = 3}, {a, b})` yields $\{-3, 6\}, \{0, 3\}$, displayed as .

Note: Sometimes you need to do some manipulation to allow the automatic solver to work, for example `Solutions(TrigExpand(sin(5/4 π + x) - cos(x - 3/4 π) = sqrt(6) * cos(x) - sqrt(2)))` See also Solve Command.

Solve Command

Note: Commands **Solve** and Solutions solve an equation or a system of equations over the real numbers symbolically. To solve equations numerically, use the NSolve Command. For solving equations in complex numbers see CSolve Command.

The following commands are only available in the CAS View.

Solve(<Equation in x>)

Solves a given equation for the main variable and returns a list of all solutions.

Example: `Solve(x^2 = 4x)` yields $\{x = 4, x = 0\}$, the solutions of $x^2 = 4x$.

Solve(<Equation>, <Variable>)

Solves an equation for a given unknown variable and returns a list of all solutions.

Example: `Solve(x * a^2 = 4a, a)` yields $\{\}$, the solutions of $xa^2 = 4a$.

Solve(<List of Equations>, <List of Variables>)

Solves a set of equations for a given set of unknown variables and returns a list of all solutions.

Examples:

- `Solve({x = 4 x + y, y + x = 2}, {x, y})` yields $(x = -1, y = 3)$, the sole solution of $x = 4x + y$ and $y + x = 2$
- `Solve({2a^2 + 5a + 3 = b, a + b = 3}, {a, b})` yields $\{a = 0, b = 3\}, \{a = -3, b = 6\}$.

Solve(<Equation>, <Variable>, <List of assumptions>)

Solves an equation for a given unknown variable with the list of assumptions and returns a list of all solutions.

Examples:

- `Solve(u *x < a, x, u>0)` yields $\{x < a/u\}$, the solution of $u *x < a$ assuming that $u>0$
- `Solve(u *x < a, x, {u<0, a<0})` yields $\{x > a/u\}$.

`Solve(<List of Parametric Equations>, <List of Variables>)`

Solves a set of parametric equations for a given set of unknown variables and returns a list of all solutions.

Example:

- `Solve({{(x, y) = (3, 2) + t*(5, 1), (x, y) = (4, 1) + s*(1, -1)}}, {x, y, t, s})` yields $\{x = 3, y = 2, t = 0, s = -1\}$.

Note: The right hand side of equations (in any of the above syntaxes) can be omitted. If the right hand side is missing, it is treated as 0. Sometimes you need to do some manipulation to allow the automatic solver to work, for example `Solve(TrigExpand(sin(5/4 π + x) - cos(x - 3/4 π) = sqrt(6) * cos(x) - sqrt(2)))`. For piecewise-defined functions, you will need to use `NSolve` Command `NSolve`

SolveCubic Command

CAS Syntax

`SolveCubic(<Cubic Polynomial>)`

Solves a given cubic polynomial and returns a list of all solutions.

Example:

`SolveCubic(x3 - 1)` yields $\{1, , \}$.

Note:

Often the answers are very cumbersome, e.g. `SolveCubic(x3 + x2 + x + 2)` in which case `Solve(x3 + x2 + x + 2)` or `CSolve(x3 + x2 + x + 2)` may work better for you.

Substitute Command

CAS Syntax

Substitute(<Expression>, <from>, <to>)

Substitutes *from* in *expression* with *to*.

Examples:

- Substitute(($x^2 / (4x + 6)$) $^2 + 6(x^2 / (4x + 6)) + 8$, x^2 , $a * (4x + 6)$)
yields $a^2 + 6a + 8$.
- Substitute(($3m - 3$) $^2 - (m + 3)^2$, m , a) yields $8a^2 - 24a$.

Substitute(<Expression>, <Substitution List>)

Substitutes in *expression* every variable of the list with the variable or number you choose for it.

Example: Substitute($2x + 3y - z$, { $x = a$, $y = 2$, $z = b$ }) yields $2a - b + 6$.

ToComplex Command

ToComplex(<Vector>)

Transforms a vector or point to a complex number in algebraic form.

Example:

ToComplex((3, 2)) yields $3 + 2i$.

CAS Syntax

ToComplex(<Vector>)

Transforms a vector or point to a complex number in algebraic form.

Example:

ToComplex((3, 2)) yields $3 + 2i$.

Note:

- The complex *i* is obtained by pressing ALT + i.
- See also ToExponential Command, ToPoint Command and ToPolar Command.

ToExponential Command

CAS Syntax

`ToExponential(<Complex Number>)`

Transforms a complex number into its exponential form.

Example: `ToExponential(1 + i)` yields $e^{i\pi/3}$.

Note:

- The complex i is obtained by pressing ALT + i.
- See also ToPoint Command, ToComplex Command and ToPolar Command.

ToPoint Command

`ToPoint(<Complex Number>)`

Creates a point from the complex number.

Example: `ToPoint(3 + 2i)` creates a point with coordinates $(3, 2)$.

Notes:

- The complex i is obtained by pressing ALT + i.
- See also the following commands: ToComplex, ToExponential and ToPolar.

ToPolar Command

`ToPolar(<Vector>)`

Transforms a vector into its polar coordinates.

Example: `ToPolar({1, sqrt(3)})` yields $(2; 60^\circ)$ in the Algebra View and $(2;)$ in the CAS View.

`ToPolar(<Complex Number>)`

Transforms a complex number into its polar coordinates.

Example: `ToPolar(1 + sqrt(3) * i)` yields $(2; 60^\circ)$ in the Algebra View and $(2;)$ in the CAS View.

Note:

- The complex i is obtained by pressing ALT + i.
- See also ToComplex Command, ToExponential Command and ToPoint Command.

3D Commands

<links/>

Plane Command

Plane(<Polygon>)

Creates a plane through the polygon.

Plane(<Conic>)

Creates a plane through the conic.

Plane(<Point>, <Plane>)

Creates a plane through the given point, parallel to the given plane.

Plane(<Point>, <Line>)

Creates a plane through the given point and line.

Plane(<Line> , <Line>)

Creates the plane through the lines (if the lines are in the same plane).

Plane(<Point>, <Point>, <Point>)

Creates a plane through three points.

Note: See also Plane through 3 Points and Plane tools.

Cone Command

Cone(<Circle>, <Height>)

Creates a cone with given base and height.

Cone(<Point>, <Point>, <Radius>)

Creates a cone with vertex (second point), circle center (first point) and given radius.

Cone(<Point>, <Vector>, <Angle α >)

Creates an infinite cone with given point as vertex, axis of symmetry parallel to the given vector and apex angle 2α .

Note: This command yields *undefined* if angle $\geq 180^\circ$.

Note: See also InfiniteCone command, the Cone tool and the Extrude to Pyramid or Cone tool, that operates by dragging or selecting a circle and entering altitude to create a right circular cone.

Height Command

Height(<Solid>)

Calculates the "oriented" height of the given solid.

Examples:

- Height(<Cone>) calculates the "oriented" height of the given cone.
- Height(<Cylinder>) calculates the "oriented" height of the given cylinder.
- Height(<Polyhedron>) calculates the "oriented" height of the given solid polyhedron.

Sphere Command

Sphere(<Point>, <Radius>)

Creates a sphere with center and radius.

Sphere(<Point>, <Point>)

Creates a sphere with center in the first point through the second point.

Example:

`Sphere((0, 0, 0), (1, 1, 1))` yields $x^2 + y^2 + z^2 = 3$

Note: See also Sphere with Center through Point Tool and Sphere with Center and Radius Tool.

Net Command

Net(<Polyhedron> , <Number>)

Creates the net of a convex polyhedron, on the plane containing the face used for its construction. The number is used to define the progress of the unfolding procedure, and needs to be between 0 and 1. The net is totally unfold when the given number is 1.

Note: The net of a cube is displayed as Latin cross.

Net(<Polyhedron> , <Number> , <Face> , <Edge> , <Edge> , ...)

Applicable only to cubes (for the moment), allows you to create different nets of a cube, by specifying the face and edges that need to be cut to create the net.

Note:

- To explore the different configurations of the net of a cube, see this example file ^[1] on GeoGebra.
- See also Net tool.

References

[1] <http://geogebra.org/material/show/id/136596>

Top Command

Top(<Quadric>)

Creates the top of the limited quadric.

Examples:

- Top(cylinder) yields a circle.
- Top(cone) yields the cone end (point).

Note: See also Bottom Command, Ends Command and Side Command.

Surface Command

Surface(<Expression>, <Expression>, <Expression>, <Parameter Variable 1>, <Start Value>, <End Value>, <Parameter Variable 2>, <Start Value>, <End Value>)

Yields the Cartesian parametric 3D surface for the given *x*-expression (first <*Expression*>), *y*-expression (second <*Expression*>) and *z*-expression (third <*Expression*>), using two <*Parameter Variables*> within the given intervals [<*Start Value*>, <*End Value*>].

Example: Let *r* and *R* be two positive real numbers: Surface((R + r cos(u)) cos(v) , (R + r cos(u)) sin(v) , r sin(u) , u, 0, 2 π , v, 0, 2 π) creates the torus generated by a circle of radius *r* whose center rotates about zAxis at a distance *R*.

Note:

- *End Value* must be greater than or equal to *Start Value* and both must be finite.
- *x*, *y* and *z* are not allowed as parameter variables.

Surface(<Function>, <Angle>)

Creates a surface of revolution, rotating the given *Function* from 0 to given *Angle* around the x-axis.

Surface(<Curve>, <Angle>, <Line>)

Creates a surface of revolution, rotating the *Curve* from 0 to given *Angle* around the *Line*.

Octahedron Command

Octahedron(<Point>, <Point>, <Direction>)

Creates an octahedron having the segment between the two points as an edge.

The other vertices are univocally determined by the given direction, that needs to be:

- a vector, a segment, a line, a ray **orthogonal** to the segment, or
- a polygon, a plane **parallel** to the segment.

The created octahedron will have:

- a face with the segment as an edge in a plane orthogonal to the given vector/segment/line/ray, or
- a face with the segment as an edge in a plane parallel to the polygon/plane.

Octahedron(<Point>, <Point>, <Point>)

Creates an octahedron with the three points of the first face. The points have to draw an equilateral triangle for the octahedron to be defined.

Octahedron(<Point>, <Point>)

Creates an octahedron with the two points of the first face, and the third point automatically created on a circle, so that the octahedron can rotate around its first edge.

Note: Octahedron(A, B) is a shortcut for Octahedron(A, B, C) with C = Point(Circle(Midpoint(A, B), Distance(A, B) sqrt(3) / 2, Segment(A, B))).

Note: See also Cube, Tetrahedron, Icosahedron, Dodecahedron commands.

InfiniteCone Command

InfiniteCone(<Point>, <Vector>, <Angle α >)

Creates an infinite cone with given point as vertex, axis of symmetry parallel to the given vector and apex angle 2α .

InfiniteCone(<Point>, <Point>, <Angle α >)

Creates an infinite cone with given first point as vertex, line through two points as axis of symmetry and apex angle 2α .

InfiniteCone(<Point>, <Line>, <Angle α >)

Creates an infinite cone with given point as vertex, axis of symmetry parallel to given line and apex angle 2α .

Notes: If you enter the angle without the degree symbol, you will get the apex angle in radian. See also the Cone CommandCone command and Cone ToolCone tool.

InfiniteCylinder Command

`InfiniteCylinder(<Line>, <Radius>)`

Creates an infinite cylinder with given radius and given line as an axis of symmetry.

Example:

`InfiniteCylinder(xAxis, 2)` creates an infinite cylinder with radius 2 and with the x-axis as axis of symmetry.

`InfiniteCylinder(<Point>, <Vector>, <Radius>)`

Creates an infinite cylinder with given radius and with axis of symmetry through a given point parallel to the vector.

`InfiniteCylinder(<Point>, <Point>, <Radius>)`

Creates an infinite cylinder with given radius and with line through two points as an axis of symmetry.

Note: See also the Cylinder command and Cylinder tool.

IntersectConic Command

`IntersectConic(<Plane>, <Quadric>)`

Intersects the plane with the quadric.

`IntersectConic(<Quadric>, <Quadric>)`

Returns a conic defined in case where the intersection is actually a conic.

Example:

`IntersectConic(sphere1, sphere2)` creates the intersection conic of two spheres.

Note: See also Intersect and IntersectPath commands.

Side Command

Side(<Quadric>)

Creates the side of the limited quadric.

Example:

`Side(cylinder)` creates the curved surface area of the cylinder.

Note: See also Top Command, Bottom Command and Ends Command.

PerpendicularPlane Command

PerpendicularPlane(<Point>, <Line>)

Creates a plane through the given point, perpendicular to the given line.

PerpendicularPlane(<Point>, <Vector>)

Creates a plane through the given point, perpendicular to the given vector.

Note: See also Perpendicular Plane Tool.

Polyhedron Command

Polyhedron Command

Prism Command

Prism(<Point>, <Point>, ...)

Returns a prism defined by the given points.

Example:

`Prism(A, B, C, D)` creates the prism with base ABC and top DEF. The vectors AD, BE, CF are equal.

Prism(<Polygon>, <Point>)

Creates a prism with the given polygon as base and the point as first top point.

Example:

`Prism(poly1, A)` creates a prism with base *poly1* and top point A.

Prism(<Polygon>, <Height value>)

Creates a right prism with the polygon as base and given height.

Example:

`Prism(poly1, 3)` creates a prism with base *poly1* and height 3.

Note: See also Prism and Extrude to Prism tools.

Dodecahedron Command

Dodecahedron(<Point>, <Point>, <Direction>)

Creates a dodecahedron having the segment between two points as an edge.

The other vertices are univocally determined by the given direction, that needs to be:

- a vector, a segment, a line, a ray **orthogonal** to the segment, or
- a polygon, a plane **parallel** to the segment.

The created dodecahedron will have:

- a face with the segment as an edge in a plane orthogonal to the given vector/segment/line/ray, or
- a face with the segment as an edge in a plane parallel to the polygon/plane.

Dodecahedron(<Point>, <Point>, <Point>)

Creates a dodecahedron with three (adjacent) points of the first face. The points have to start a regular pentagon for the dodecahedron to be defined.

Dodecahedron(<Point>, <Point>)

Creates a dodecahedron with two (adjacent) points of the first face, and the third point automatically created on a circle, so that the dodecahedron can rotate around its first edge.

Note: Dodecahedron(A, B) is a shortcut for Dodecahedron(A, B, C) with $C = \text{Point}(\text{Circle}(((1 - \sqrt{5})) A + (3 + \sqrt{5})) B / 4, \text{Distance}(A, B) \sqrt{10 + 2\sqrt{5}} / 4, \text{Segment}(A, B))$.

Note: See also Cube, Tetrahedron, Icosahedron, Octahedron commands.

Icosahedron Command

Icosahedron(<Point>, <Point>, <Direction>)

Creates an icosahedron having the segment between the two points as an edge.

The other vertices are univocally determined by the given direction, that needs to be:

- a vector, a segment, a line, a ray **orthogonal** to the segment, or
- a polygon, a plane **parallel** to the segment.

The created icosahedron will have:

- a face with the segment as an edge in a plane orthogonal to the given vector/segment/line/ray, or
- a face with the segment as an edge in a plane parallel to the polygon/plane.

Icosahedron(<Point>, <Point>, <Point>)

Creates an icosahedron with the three points of the first face. The points have to draw an equilateral triangle for the icosahedron to be defined.

Icosahedron(<Point>, <Point>)

Creates an icosahedron with the two points of the first face, and the third point automatically created on a circle, so that the icosahedron can rotate around its first edge.

Note: Icosahedron(A, B) is a shortcut for Icosahedron(A, B, C) with $C = \text{Point}(\text{Circle}(\text{Midpoint}(A, B), \text{Distance}(A, B) \sqrt{3} / 2, \text{Segment}(A, B)))$.

Note: See also Cube, Tetrahedron, Octahedron, Dodecahedron commands.

Tetrahedron Command

Tetrahedron(<Point>, <Point>, <Direction>)

Creates a tetrahedron having the segment between the two points as an edge.

The other vertices are univocally determined by the given direction, that needs to be:

- a vector, a segment, a line, a ray **orthogonal** to the segment, or
- a polygon, a plane **parallel** to the segment.

The created tetrahedron will have:

- a face with the segment as an edge in a plane orthogonal to the given vector/segment/line/ray, or
- a face with the segment as an edge in a plane parallel to the polygon/plane.

Tetrahedron(<Point>, <Point>, <Point>)

Creates a tetrahedron with the three points of the first face. The points have to draw an equilateral triangle for the tetrahedron to be defined.

Tetrahedron(<Point>, <Point>)

Creates a tetrahedron with the two points of the first face, and the third point automatically created on a circle, so that the tetrahedron can rotate around its first edge.

Note: Tetrahedron[A, B] is a shortcut for Tetrahedron(A, B, C) with $C = \text{Point}(\text{Circle}(\text{Midpoint}(A, B), \text{Distance}(A, B) \sqrt{3} / 2, \text{Segment}(A, B)))$.

Note: See also Cube, Octahedron, Icosahedron, Dodecahedron commands.

Bottom Command

Bottom(<Quadric>)

Creates the bottom of the limited quadric.

Example:

`Bottom(cylinder)` yields a circle.

Note: See also Top Command, Ends Command and Side Command.

Volume Command

Volume(<Solid>)

Calculates the volume of the given solid.

Example:

- Volume(<Pyramid>) calculates the volume of the given pyramid.
- Volume(<Prism>) calculates the volume of the given prism.
- Volume(<Cone>) calculates the volume of the given cone.
- Volume(<Cylinder>) calculates the volume of the given cylinder.

Note: See also Volume tool.

Cube Command

Cube(<Point>, <Point>, <Direction>)

Creates a cube having the segment between the two points as an edge.

The other vertices are uniquely determined by the given direction, that should be one of:

- a vector, a segment, a line, a ray **orthogonal** to the segment, or
- a polygon, a plane **parallel** to the segment.

The created cube will have:

- a face with the segment as an edge in a plane orthogonal to the given vector/segment/line/ray, or
- a face with the segment as an edge in a plane parallel to the polygon/plane.

Cube(<Point>, <Point>, <Point>)

Creates a cube with three (adjacent) points of the first face. The points have to start a square for the cube to be defined.

Cube(<Point>, <Point>)

Creates a cube with two (adjacent) points of the first face, and the third point automatically created on a circle, so that the cube can rotate around its first edge.

Note: Cube(A, B) is a shortcut for Cube(A, B, C) with C = Point(Circle(B, Distance(A, B), Segment(A, B))).

Note: See also Tetrahedron, Octahedron, Icosahedron, Dodecahedron commands.

Cylinder Command

`Cylinder(<Circle>, <Height>)`

Creates a cylinder with given base and given height.

`Cylinder(<Point>, <Point>, <Radius>)`

Creates a cylinder with given radius and with given points as the centers of the top and bottom.

Note: See also the InfiniteCylinder command and the Cylinder and Extrude to Prism or Cylinder tools.

Pyramid Command

`Pyramid(<Point>, <Point>, ...)`

Returns a pyramid defined by the given points.

Example:

`Pyramid(A, B, C, D)` creates the pyramid with base *ABC* and apex *D*.

`Pyramid(<Polygon>, <Point>)`

Creates a pyramid with the given polygon as base and the point as apex.

Example:

`Pyramid(poly1, A)` creates a pyramid with base *poly1* and apex *A*.

`Pyramid(<Polygon>, <Height>)`

Returns a centered pyramid defined by the polygon as base and given height.

Example:

`Pyramid(poly1, 3)` creates a centered pyramid with base *poly1* and height 3.

Note: See also Pyramid and Extrude to Pyramid tools.

Ends Command

`Ends(<Quadric>)`

Creates the top and the bottom of the limited quadric.

Examples:

- `Ends(cylinder)` yields two circles.
- `Ends(cone)` yields a circle and the cone end (point).

Note: See also Top Command, Bottom Command and Side Command.

PlaneBisector Command

`PlaneBisector(<Point> , <Point>)`

Creates the plane orthogonal bisector between the two points.

`PlaneBisector(<Segment>)`

Creates the plane orthogonal bisector of the segment.

Predefined Functions and Operators

To create numbers, coordinates, or equations using the Input Bar you may also use the following pre-defined functions and operations. Logic operators and functions are listed in article about Boolean values.

Note: The predefined functions need to be entered using parentheses. You must not put a space between the function name and the parentheses.

Operation / Function	Input
e (Euler's number)	Alt + e
i (Imaginary unit)	Alt + i
π	Alt + p or pi
$^\circ$ (Degree symbol)	Alt + o or deg
Addition	+
Subtraction	-
Multiplication	* or Space key
Scalar product	* or Space key
Vector product(see Points and Vectors)	\otimes
Division	/
Exponentiation	^ or superscript (x^2 or x^2)
Factorial	!
Parentheses	()
x-coordinate	$x()$
y-coordinate	$y()$
z-coordinate	$z()$

Argument (also works for 3D points / vectors)	arg()
Conjugate	conjugate()
Real	real()
Imaginary	imaginary()
Absolute value	abs()
Altitude angle (for 3D points / vectors)	alt()
Sign	sgn() or sign()
Greatest integer less than or equal	floor()
Least integer greater than or equal	ceil()
Round to nearest integer (or to y decimal places)	round(x) or round(x, y)
Square root	sqrt()
Cubic root	cbrt()
The nth root of x	nroot(x, n)
Random number between 0 and 1	random()
Exponential function	exp() or e^x
Logarithm (natural, to base e)	ln()
Logarithm to base 2	log ₂ () or ld()
Logarithm to base 10	log ₁₀ () or log() or lg()
Logarithm of x to base b	log(b, x)
Cosine	cos()
Sine	sin()
Tangent	tan()
Secant	sec()
Cosecant	csc() or cosec()
Cotangent	cot() or cotan()
Arc cosine (answer in radians)	acos() or arccos()
Arc cosine (answer in degrees)	acosd()
Arc sine (answer in radians)	asin() or arcsin()
Arc sine (answer in degrees)	asind()
Arc tangent (answer in radians, between -π/2 and π/2)	atan() or arctan()
Arc tangent (answer in degrees, between -90° and 90°)	atand()
Arc tangent (answer in radians, between -π and π) ^[1]	atan2(y, x)
Arc tangent (answer in degrees, between -180° and 180°) ^[1]	atan2d(y, x)
Hyperbolic cosine	cosh()
Hyperbolic sine	sinh()
Hyperbolic tangent	tanh()
Hyperbolic secant	sech()
Hyperbolic cosecant	csch()
Hyperbolic cotangent	coth() or cotanh()

Antihyperbolic cosine	acosh() or arccosh()
Antihyperbolic sine	asinh() or arcsinh()
Antihyperbolic tangent	atanh() or arctanh()
Beta function [2] $B(a, b)$	beta(a, b)
Incomplete beta function [3] $B(x; a, b)$	beta(a, b, x)
Incomplete regularized beta function [4] $I(x; a, b)$	betaRegularized(a, b, x)
Gamma function $\Gamma(x)$	gamma(x)
(Lower) incomplete gamma function [5] $\gamma(a, x)$	gamma(a, x)
(Lower) incomplete regularized gamma function $P(a, x) = \gamma(a, x) / \Gamma(a)$ [6]	gammaRegularized(a, x)
Gaussian Error Function	erf(x)
Digamma function	psi(x)
The Polygamma function [7] is the $(m+1)$ th derivative of the natural logarithm of the Gamma function, gamma(x) [8] ($m=0,1$)	polygamma(m, x)
The Sine Integral [9] function	sinIntegral(x)
The Cosine Integral [10] function	cosIntegral(x)
The Exponential Integral [11] function	expIntegral(x)
The Riemann-Zeta [12] function $\zeta(x)$	zeta(x)
Lambert's W function [13] LambertW(x, branch)	LambertW(x, 0), LambertW(x, -1)

Note: The x, y, z operators can be used to get corresponding coefficients of a line.

References

- [1] <http://en.wikipedia.org/wiki/Atan2>
- [2] <http://mathworld.wolfram.com/BetaFunction.html>
- [3] <http://mathworld.wolfram.com/IncompleteBetaFunction.html>
- [4] <http://mathworld.wolfram.com/RegularizedBetaFunction.html>
- [5] <http://mathworld.wolfram.com/IncompleteGammaFunction.html>
- [6] <http://mathworld.wolfram.com/RegularizedGammaFunction.html>
- [7] http://en.wikipedia.org/wiki/Polygamma_function
- [8] http://en.wikipedia.org/wiki/Gamma_function
- [9] <http://mathworld.wolfram.com/SineIntegral.html>
- [10] <http://mathworld.wolfram.com/CosineIntegral.html>
- [11] <http://mathworld.wolfram.com/ExponentialIntegral.html>
- [12] http://en.wikipedia.org/wiki/Riemann_zeta_function
- [13] https://en.wikipedia.org/wiki/Lambert_W_function

Imaginary Function

`imaginary(<Complex Number>)`

Returns the imaginary part of a given complex number.

Example:

`imaginary(17 + 3 i)` yields 3.

Note:

- The complex `i` is obtained by pressing ALT + i.
- See also real Function.

FractionalPart Function

`fractionalPart(<Expression>)`

Returns the fractional part of the expression.

Examples:

- `fractionalPart(6 / 5)` yields in CAS View, 0.2 in Algebra View.
- `fractionalPart(1/5 + 3/2 + 2)` yields in CAS View, 0.7 in Algebra View.

Note:

In Mathematics fractional part function is defined sometimes as $\text{assgn}(x)(\lfloor x \rfloor - \lfloor x \rfloor)$. In other cases $\text{assgn}(x)(\lfloor x \rfloor - \lfloor x \rfloor)$. **GeoGebra** uses the second definition (also used by Mathematica).

To obtain the first function you may use `f(x) = x - floor(x)` See also Predefined Functions and Operators.

Nroot Function

`nroot(<Expression>, <N>)`

Calculates the n^{th} root of a given expression.

Example: `nroot(16, 4)` yields 2 . `nroot(x^8, 2)` yields $f(x) = \sqrt[2]{x^8}$ in Algebra, but x^4 in the CAS View.

Note: See also Predefined Functions and Operators.

Real Function

`real(<Complex Number>)`

Returns the real part of a given complex number.

Example:

`real(17 + 3 i)` yields 17 , the real part of the complex number $17 + 3i$.

Note:

- The complex i is obtained by pressing ALT + i.
- See also imaginary Function.

User interface

Views

What are Views

GeoGebra provides different *Views* for mathematical objects, which are displayed in different representations (e.g. algebraic and graphical) and are linked dynamically. This means that if you modify an object in any of the *Views*, its representations in the other *Views* automatically adapt to these changes if possible.

It is possible to customize the position and dimensions of the *Views* on the screen of your device, by dragging them with the mouse (version 5) or by selecting the *View* menu, then dragging the icon on the top right corner of the *View* that you wish to move (version 6).

Main Views

Algebra View:

Algebraic representations of objects are displayed and can be entered directly using the (virtual) keyboard (e.g. coordinates of points, equations).

Graphics View:

Mathematical objects can be constructed with your mouse or by using a touch pad and changed dynamically afterwards.

3D Graphics View:

Three dimensional mathematical objects can be constructed and changed dynamically.

Spreadsheet View:

You can work with data and explore statistical concepts.

CAS View:

GeoGebra's Computer Algebra System can be used for numerical and symbolic computations.

Other Display Features

Construction Protocol: This interactive list of your construction steps allows you to redo your construction step by step.

Probability Calculator: Allows you to easily calculate and graph probability distributions.

Graphics View

Graphics View User Interface

The *Graphics View* always displays the graphical representation of objects created in GeoGebra. In addition, the *Graphics View Toolbar* is displayed at the top of the GeoGebra window, with the *Undo / Redo* buttons in the top right corner. The *Graphics View* is part of almost all *Perspectives*.

Graphics View

Customizing the Graphics View

The *Graphics View* may include various types of grids and axes. For more information see Customizing the Graphics View. You may also change the layout of GeoGebra's user interface according to your needs.

Displaying a Second Graphics View

A second *Graphics View* may be opened using the *View Menu*. If two *Graphics Views* are opened, one of them is always active (either it's being worked with, or it is the last *View* that has been worked with). All visible objects created by *Commands* appear in the active *Graphics View*.

Hint: You may specify for each object, in which *Graphic View(s)* it should be visible by using the *Advanced* tab of the Properties Dialog.

Creating Mathematical Objects

Constructions using Tools

Using the construction *Tools* available in the *Graphics View Toolbar* you can create geometric constructions in the *Graphics View*. Select any construction tool from the *Graphics View Toolbar* and read the provided tooltip, in order to find out how to use the selected *Tool*.

Note: Any object you create in the *Graphics View* also has an algebraic representation in the *Algebra View*.

Example: Select the *Circle with Center through Point Tool* and click twice in the *Graphics View*. The first click creates the center point while the second click creates a circle and a point on the circle.

Graphics View Toolbar

The *Graphics View Toolbar* provides a wide range of *Tools* that allow you to create the graphical representations of objects directly in the *Graphics View*. Every icon in the *Toolbar* represents a *Toolbox* that contains a selection of related construction *Tools*. In order to open a *Toolbox*, you need to click on the corresponding default *Tool* shown in the *Graphics View Toolbar* (GeoGebra Web and Tablet Apps) or on the small arrow in the lower right corner of the *Toolbar* icon (GeoGebra Desktop).

Note: The *Tools* of the *Graphics View Toolbar* are organized by the nature of resulting objects or the functionality of the *Tools*. You will find *Tools* that create different types of points in the *Points Toolbox* (default icon) and *Tools* that allow you to apply geometric transformations in the *Transformations Toolbox* (default icon).

Direct Input using the Input Bar

You can also create objects in the *Graphics View* by entering their algebraic representation or corresponding *Commands* into the *Input Bar*.

Hint: The *Input Bar* can be shown using the *View Menu*.

Modifying Mathematical Objects

The Move Tool

After activating the *Move Tool* you are able to move objects in the *Graphics View* by dragging them with the mouse or with a touch pad.

Note: At the same time, their algebraic representations are dynamically updated in the *Algebra View*.

Copy & Paste

Via Keyboard Shortcut Ctrl + C and Ctrl + V (Mac OS:Ctrl + C and Ctrl + V) you can *Copy and Paste* selected objects (except if they depend on the coordinate axes) into either the same or into another window.

Note: *Copy and Paste* will copy every ancestor of the selected objects but makes the non-selected objects invisible.

Example: If you copy objects depending on sliders into a new window, it will copy the slider (invisible) into the window, too.

The pasted object is fixed when you click on the *Graphics View*. If the copied object depends on at least one point then it can snap onto existing points when pasted (but only the point following the mouse pointer will do this).

Display of Mathematical Objects

Graphics View Style Bar

The *Graphics View Style Bar* contains buttons to

- show / hide the coordinate axes and a grid (different kinds of grids in the GeoGebra Web and Tablet Apps)
- go back to default view
- change the *Point Capturing* settings
- open the *Properties Dialog* (GeoGebra Web and Tablet Apps)
- display additional *Views* in the GeoGebra window (GeoGebra Web and Tablet Apps)

Style Bar for Tools and Objects

Depending on the *Tool* or object you select, the buttons in the *Style Bar* adapt to your selection. Please see Style Bar Options for Tools and Objects for more information.

Hiding Mathematical Objects in the Graphics View

You may hide objects in the *Graphics View* by either

- using the *Show / Hide Object Tool*
- opening the *Context Menu* and unchecking *Show Object*

Customizing the Graphics View

In order to adjust the visible part of the Graphics View, you can drag it by using tool Move Graphics View Tool and use the following ways of zooming:

- You may use the Zoom In Tool and Zoom Out Tool in order to zoom in the *Graphics View*. **Note:** The position of your click determines the center of zoom.
- You may use the scroll wheel of your mouse in order to zoom in the *Graphics View*.
- You may use keyboard shortcuts to zoom in **Ctrl + +** and to zoom out **Ctrl + -**.
- After right clicking (Mac OS: Ctrl-click) on an empty spot in the *Graphics View* a context menu appears which allows you to zoom.
- You may also specify range of *x*- and *y*-axis in the Properties Dialog for *Graphics View* (see below).

Showing and hiding objects

In the Algebra View, the icon to the left of every object shows its current visibility state (shown or hidden). You may directly click on the little marble icon in order to change the visibility status of an object. For more information see visibility.

Properties of Graphics View (Properties Dialog)

Coordinate axes, grid and some other properties can be customized using the Properties Dialog of the *Graphics View*. After right clicking (Mac OS: Ctrl-click) in the *Graphics View*, you can open this dialog window by selecting *Graphics...* from the appearing Context Menu.

Customizing Coordinate Axes and Grid

To show or hide the axes and the grid, right click (Mac OS: Ctrl-click) on the drawing pad and select the corresponding items *Axes* or *Grid* from the appearing context menu. For further setting you need to open the *Properties Dialog*.

- On tab *Basic*, you can, for example, change the line style and color of the coordinate axes, and set the ratio between the axes. To make sure the axes ratio cannot be changed by any command or user action, you may lock it using the lock icon.
- Clicking on tabs *xAxis* and *yAxis* allows you to customize the axes individually, set the distance of the tickmarks, labeling, axes visibility, units and more. If you want the cross of the axes to be at point (a,b) , you can set *Cross at* parameter for *xAxis* to b and for *yAxis* to a . Option *Stick to edge* means that the line remains close to the bottom or left border of the screen. To draw only the part of the axis to the right or to the top of the axes intersection, you can select *Positive direction only*.
- On tab *Grid*, you can change the color and line style of the coordinate grid, set the distance and ratio for grid lines to a certain value, and the grid visibility. Three types of grid are available: Cartesian, polar and isometric.

Note: Axes scaling is possible in every mode by pressing and holding the Shift-key (PC: also Ctrl-key) while dragging the axis. Range of the axes may be given dynamically, e.g. in Basic tab you can set X Min to $x(A)$ and Y Min to $y(A)$ to ensure the lower left corner of the view remains in point A. In such setting, the view cannot be zoomed.

Showing Navigation Bar

You can add the Navigation Bar for Construction Steps to the *Graphics View* by enabling it in *Basic* tab of the *Properties Dialog*. You can also add the *Play* button to allow animating the construction steps and a button to show the Construction Protocol.

Miscellaneous settings

These settings are located in the last part of the *Basic* tab of *Properties Dialog*.

Background color

Allows you to change background color of the *Graphics View*.

Tooltips

Allows you to set the behavior of tooltips in the *Graphics View*. See article on Tooltips for details.

Show mouse coordinates

Enables display of mouse coordinates next to the mouse pointer.

Algebra View

Algebra View User Interface

By default, the *Algebra View* is opened next to the *Graphics View*. In addition, either the *Input Bar* is displayed at the bottom of the GeoGebra window (GeoGebra Desktop), or an *Input Field* is integrated directly in the *Algebra View* (GeoGebra Web and Tablet Apps). The *Graphics View Toolbar* is displayed at the top of the GeoGebra window, with the *Undo / Redo* buttons in the top right corner.

GeoGebra Web and Tablet Apps

GeoGebra Desktop

The *Algebra View* is part of the *Algebra Perspective*, although you may change the layout of GeoGebra's user interface according to your needs.

Creating Mathematical Objects

Direct Input

In the *Algebra View* you can directly enter algebraic expressions using the integrated *Input Field* (GeoGebra Web and Tablet Apps) or the *Input Bar* at the bottom of the GeoGebra window (GeoGebra Desktop). After hitting the Enter key your algebraic input appears in the *Algebra View* while its graphical representation is automatically displayed in the *Graphics View*.

Example: The input $y = 2 \times + 3$ gives you the linear equation in the *Algebra View* and the corresponding line in the *Graphics View*.

Commands

GeoGebra also offers a wide range of *Commands* that can be used to create objects in the *Algebra View*. Just start typing the name of a *Command* into the *Input Bar* or *Input Field* and GeoGebra will offer you a list of *Commands* that match your input.

Tools

Although the *Algebra View* doesn't have its own *Toolbar*, you can create *Dependent Objects*. Select a tool from the *Graphics View Toolbar* and click on any appropriate object in the *Algebra View* in order to create a new *Dependent Object*.

Example: Create two points *A* and *B*, whose coordinates are displayed in the *Algebra View*. Select the *Line Tool* from the *Graphics View Toolbar* and click on both points in the *Algebra View* in order to create a line through points *A* and *B*.

Modifying Mathematical Objects

You can modify the algebraic representation of mathematical objects directly in the *Algebra View*.

Activate the *Move* tool and double-click a *Free Object* in the *Algebra View*. In the appearing text box you can directly modify its algebraic representation. After hitting the Enter key, both the algebraic representation in the *Algebra View* and the graphical representation of the object in the *Graphics View* will automatically adapt to your changes.

If you double-click on a *Dependent Object* in the *Algebra View*, a dialog window appears allowing you to Redefine the object.

Display of Mathematical Objects

By default, mathematical objects are organized by *Object Types* in the *Algebra View*. In GeoGebra Desktop, you may use the *Style Bar* option *Sort by* in order to re-sort the objects by *Dependency*, *Layer* or *Construction Order*.

Hint: You can collapse or expand each group of objects individually (e.g. all points, all free objects, all objects on a specific layer) by clicking on the plus or minus symbol in front of the group's name.

Algebra View Style Bar

The *Algebra View Style Bar* provides buttons to

- show / hide *Auxiliary Objects*
- sort the list of objects by different criteria
- display additional *Views* in the GeoGebra window (GeoGebra Web and Tablet Apps)

Hiding Mathematical Objects in the Algebra View

If you want to hide the algebraic representation of an object in the *Algebra View*, you may specify the object as an *Auxiliary Object*. Open the tab *Basic* of the Properties Dialog and check *Auxiliary Object*.

By default, *Auxiliary Objects* are not shown in the *Algebra View*. You can change this setting by selecting *Auxiliary Objects* from the *Context Menu* (right-click or Mac OS Ctrl-click), or by selecting on the appropriate icon in the *Algebra View Style Bar*.

Spreadsheet View

Spreadsheet View User Interface

By default, the *Spreadsheet View* is opened next to the *Graphics View*. The *Spreadsheet View Toolbar* is displayed at the top of the GeoGebra window, with the *Undo / Redo* buttons in the top right corner.

Spreadsheet View

The *Spreadsheet View* is part of the *Spreadsheet Perspective*, although you may change the layout of GeoGebra's user interface according to your needs.

Customizing the Spreadsheet View

The *Spreadsheet View* can be customized according to your preferences by

- opening the *Preferences Dialog* and selecting the option *Spreadsheet* (GeoGebra Desktop)
- opening the *Spreadsheet View Style Bar* and selecting *Preferences* (GeoGebra Web and Tablet Apps)

In the appearing dialog window you may change the layout by choosing whether to

- Show the *Input Bar*
- Show *Gridlines*
- Show the *Column or Row Header*
- Show *Vertical or Horizontal Scrollbars*

In addition, you may change the behavior of the *Spreadsheet View* by

- allowing *Use of Buttons and Checkboxes* or *Tooltips*
- requiring " $=$ " before *Commands*
- using *Auto-complete*

Creating Mathematical Objects

In GeoGebra's *Spreadsheet View* every cell has a specific name that allows you to directly address each cell.

Example: The cell in column A and row 1 is named *A1*.

Note: These cell names can be used in expressions and *Commands* in order to address the content of the corresponding cell.

Direct Input, Selection and Commands

In the *Spreadsheet Cells* you can enter not only numbers, but all types of General Objects and Geometrical Objects that are supported by GeoGebra (e.g. coordinates of points, Functions, Commands).

To select non adjacent columns or cells in the spreadsheet, use the shortcut **Ctrl + Click**.

Relative Cell Names

If you copy content from one cell to another, by default all references are changed accordingly to the target position.

Example: Let $A1=1$, $A2=2$. In $B1$ put $(A1, A1)$. By copying $B1$ to $B2$ (either via $Ctrl + C$, $Ctrl + V$ or by dragging the cell corner) you get $(A2, A2)$ in $B2$.

To prevent this behavior, you can insert $\$$ before the column and/or row of the referenced cell.

Note: On Mac OS the Copy & Paste shortcuts are $Cmd + C$ and $Cmd + V$

Input Data into the Spreadsheet View

Manual Entry, Commands, and Tracing

Besides manually adding entries into the *Spreadsheet View* cells, you may use the commands *FillColumn*, *FillRow* or *FillCells*. You can also enter data by using the feature *Tracing to Spreadsheet*.

Copy Data from the Algebra View

With a simple drag and drop operation it is also possible to copy objects from the *Algebra View* to the *Spreadsheet View*. If you drag a list, its elements will be pasted horizontally, starting from the cell in which you release the left mouse button or touchscreen. Pressing the Shift key while dragging opens a dialog window when the mouse button is released, allowing you to choose whether the pasted objects will be Free or Dependent, as well as to choose the vertical placement of the copied objects (check option *Transpose*).

Copy Data from Other Spreadsheet Software

GeoGebra allows you to import data from other spreadsheet software into the *Spreadsheet View*.

- Select and copy the data you want to import. For example, you may use the keyboard shortcut $Ctrl + C$ (Mac OS: $Cmd + C$) in order to copy the data to your computer's clipboard.
- Open a GeoGebra window and show the *Spreadsheet View*.
- Click on the spreadsheet cell that should contain the first data value (e.g. cell $A1$)
- Paste the data from your computer's clipboard into GeoGebra's *Spreadsheet View*. For example, you may select a cell and use the keyboard shortcut $Ctrl + V$ (Mac OS: $Cmd + V$) in order to paste the data into the highlighted spreadsheet cell.

Import Data Files from other Applications

You can also import data from other applications, if stored using *.txt*, *.csv* and *.dat* formats. Simply right click on a free cell of the *Spreadsheet View*, then choose the *Import Data File...* option.

Note: GeoGebra uses the dot $.$ as decimal separator, and the comma $,$ as field separator. Ensure to check if your data file matches these settings before importing.

Spreadsheet View Toolbar

The *Spreadsheet View Toolbar* provides a range of *Tools* that allow you to create objects in the *Spreadsheet View*. Every icon in the *Toolbar* represents a *Toolbox* that contains a selection of related *Tools*. In order to open a *Toolbox*, you need to click on the corresponding default *Tool* shown in the *Spreadsheet View Toolbar* (GeoGebra Web and Tablet Apps) or on the small arrow in the lower right corner of the *Toolbar* icon (GeoGebra Desktop).

Note: The *Tools* of the *Spreadsheet View Toolbar* are organized by the nature of resulting objects or their functionality. For example, you will find *Tools* that analyze data in the *Data Analysis Toolbox*.

Display of Mathematical Objects

Display of Spreadsheet Objects in other Views

If possible, GeoGebra immediately displays the graphical representation of the object you entered in a *Spreadsheet Cell* in the *Graphics View* as well. Thereby, the name of the object matches the name of the *Spreadsheet Cell* used to initially create it (e.g. $A5, C1$).

Note: By default, *Spreadsheet Objects* are classified as *Auxiliary Objects* in the *Algebra View*. You can show or hide these *Auxiliary Objects* by selecting *Auxiliary Objects* from the *Context Menu* or by clicking on the icon in the *Algebra View Style Bar*.

Using Spreadsheet Data in other Views

You may process the *Spreadsheet Data* by selecting multiple cells and right-clicking (Mac OS: Cmd-clicking) on the selection. In the appearing *Context Menu*, choose the submenu *Create* and select the appropriate option (*List, List of points, Matrix, Table, Polyline* and *Operation table*).

Operation Table

For a function with two parameters you can create an *Operation Table* with values of the first parameter written in the top row and values of second parameter written in the left column. The function itself must be entered in the top left cell.

After entering the function and the parameter values in the appropriate cells, select the rectangular area of the desired *Operation Table* with the mouse. Then, right click (Mac OS: Cmd-click) on the selection and choose option *Create > Operation Table* of the appearing *Context Menu*.

Example: Let $A1 = x$, $A2 = 1$, $A3 = 2$, $A4 = 3$, $B1 = 1$, $C1 = 2$ and $D1 = 3$. Select cells $A1:D4$ with the mouse. Then, right click (Mac OS: Cmd-click) on the selection and choose *Create > Operation Table* in the *Context Menu* to create a table containing the results of substitution of the inserted values in the given function.

Spreadsheet View Style Bar

The *Spreadsheet View Style Bar* provides buttons to

- show / hide the *Input Bar* (GeoGebra Desktop)
- change the text style to **bold** or *italic*
- set the text alignment to *left*, *center*, or *right*
- change the background color of a cell
- change the cell borders (GeoGebra Desktop)
- open the *Properties Dialog* (GeoGebra Web and Tablet Apps)
- display additional *Views* in the GeoGebra window (GeoGebra Web and Tablet Apps)

CAS View

CAS View User Interface

By default, the *CAS View* is opened next to the *Graphics View*. Depending on which one of these two *Views* is active, either the *CAS View Toolbar* or the *Graphics View Toolbar* is displayed at the top of the GeoGebra window, with the *Undo / Redo* buttons in the top right corner.

CAS View

The *CAS View* is part of the *CAS Perspective*, although you may change the layout of GeoGebra's user interface according to your needs.

Creating Mathematical Objects

Direct Input

The *CAS View* allows you to use GeoGebra's CAS (Computer Algebra System) for symbolic computations. It consists of cells with an *Input Field* at the top and output display at the bottom. You can use these *Input Fields* in the same way as the normal *Input Bar*, with the following differences:

- You can use variables that were not assigned any value.
- $=$ is used for equations and $:=$ for assignments. This means that the input $a=2$ will not assign value 2 to a . See the section about assignments for details.
- Multiplication needs to be marked explicitly. While in the *Input Bar* you can use both $a(b+c)$ and $a*(b+c)$ for multiplication, in the *CAS View* only $a*(b+c)$ is valid.

Keyboard Shortcuts for Direct Input

In the *CAS View* of GeoGebra's Desktop Version the following keyboard shortcuts help you to evaluate or check your input.

Note: Instead of using these keyboard shortcuts, you may also use the corresponding *Tools* of the *CAS View Toolbar*.

- Enter: Evaluates your input.

Notes:

- Assignments are always evaluated, e.g. $a := 5$
- You may suppress any output with a semicolon at the end of your input, e.g. $a := 5;$

.

Variable Assignments and Connection with other Views

You may use the `:=` notation for assignments, e.g. `b := 5`, `a(n) := 2n + 3`.

- *Free up a name:* Use `Delete[b]` in order to free up a variable name again.
- *Redefine a variable or function:* You may redefine a variable or function, but you must do so **in the same cell**, otherwise it will be treated as a new variable and automatically renamed.

Variables and functions are always shared between the *CAS View* and the other *Views* if possible. For example:

- If you define `b := 5` in the *CAS View*, then you can use `b` in all the other *Views* of GeoGebra.
- If you define a function `f(x) = x^2` in another *View*, you can also use this function in the *CAS View*.

Note:

The output will always be just the expression **after** the `:=`, e.g. if you type `b := 5` the output will be `5`. Please also note, that for clarification actually `b := 5` will be displayed.

Equations

- You may write equations using the simple *Equals* sign, e.g. `3x + 5 = 7`.
- *Arithmetic operations:* You can perform arithmetic operations on equations, which is useful for manual equation solving. **Example:** `(3x + 5 = 7) - 5` subtracts 5 from both sides of the equation.
- *Extracting one Side:* You may extract either the left or right side of an equation by using the commands `LeftSide[]` and `RightSide[]`. **Example:** `LeftSide[3x + 5 = 7]` returns `3x + 5` and `RightSide[3x + 5 = 7]` returns `7`

Row References

You can refer to other rows in the *CAS View* in two ways:

- **Static row references** copy the output and **won't be updated** if the *referenced* row is subsequently changed
 - #: Copies the previous output.
 - #5: Copies the output of row 5.
- **Dynamic row references** insert a reference to another row instead of the actual output and therefore **will be updated** if the *referenced* row is subsequently changed
 - \$: Inserts a reference to the previous output.
 - \$5: Inserts a reference to the output of row 5

CAS Commands

GeoGebra also offers a wide range of *CAS Commands* that can be used to create objects in the *CAS View*. Just start typing the name of a *Command* into the *Input Field* and GeoGebra will offer you a list of *Commands* that match your input.

Note: For a complete list of *Commands* see section CAS Commands.

Note: From GeoGebra 5.0 onwards, the *CAS View* supports exact versions of some *Geometry Commands*

.

CAS View Toolbar

The *CAS View Toolbar* provides a range of *CAS Tools* that allow you to evaluate input and perform calculations. Just enter your input and select the corresponding *CAS Tool* afterwards in order to apply it to your input.

Hint: In GeoGebra Classic 5 you may select part of the input text to only apply the operation to this selected part. This feature is not available in Classic 6 at the moment.

Note: For a complete list of *Tools* see *CAS Tools*.

Context Menus

Row Header Context Menu

In the GeoGebra Desktop Version you can right click (MacOS: Ctrl-click) on a row header in order to show a *Context Menu* with the following options:

- **Insert Above:** Inserts an empty row above the current one.
- **Insert Below:** Inserts an empty row below the current one.
- **Delete Row:** Deletes the contents of the current row.
- **Text:** Toggles between the current result and a text showing the current result contained in the row, which allows the user to insert comments.
- **Copy as LaTeX (GeoGebra Desktop):** Copies the contents of the current row to your computer's clipboard, allowing you to paste the contents e.g. in a Text object.

Note: To copy the contents of more than one CAS row as LaTeX, select the rows you want with Ctrl-click (MacOS: Cmd-click), then right-click (MacOS: Ctrl-click) on the row header and select *Copy as LaTeX*.

Cell Context Menu

In the GeoGebra Desktop Version you can right click (MacOS: Ctrl-click) on a CAS output cell in order to show a *Context Menu* with the following options:

- **Copy:** Copies the cell contents to the your computer's clipboard. Then, right click on a new cell in order to show the **Paste** option.
- **Copy as LaTeX:** Copies the cell contents in LaTeX format to the your computer's clipboard, so it can be pasted into a Text object or a LaTeX editor.
- **Copy as LibreOffice Formula:** Copies the cell contents in LibreOffice formula format to your computer's clipboard, so it can be pasted in a word processing document.
- **Copy as Image:** Copies the cell contents in PNG format to your computer's clipboard, so it can be pasted into an Image object or in a word processing document.

Display of Mathematical Objects

CAS View Style Bar

The *CAS View Style Bar* provides buttons to

- change the text style (**bold** and *italics*) and color
- display a virtual keyboard (GeoGebra Desktop)
- display additional *Views* in the GeoGebra window (GeoGebra Web and Tablet Apps)

Showing CAS Objects in the *Graphics View*

In the *CAS View*, the icon to the left of every row shows the current visibility state (shown or hidden) of the object defined in it (when possible). You may directly click on the little *Show / Hide Object* icon in order to change the visibility status of the object in the *Graphics View*.

Probability Calculator

The *Probability Calculator* is one of GeoGebra's main perspectives. You may use it in order to calculate and graph probability distributions, as well as to conduct statistical tests.

Distributions

Tab *Distribution* allows you to graph a variety of probability distributions. Just select the distribution you want to work with from the list available in the drop down menu (e.g. Normal, Binomial) and GeoGebra will graph it for you. Then, you may adjust the parameters of the distribution in the adjacent text boxes.

You may also use the buttons provided in order to change the appearance of the distribution:

- Toggle between the probability density function and the cumulative distribution function of the distribution
- Modify your graph in order calculate a cumulative probability (e.g. $P(x \leq X)$, $P(x \geq X)$). To calculate a probability select the interval type using the buttons provided. Then adjust the interval in the adjacent text boxes or drag the corresponding markers along the x -axis in the graph.

Probability Calculator: Tab Distribution

Probability Calculator Style Bar

The *Probability Calculator Stylebar* provides options to overlay your distribution with the *Normal Curve* and to export the graph.

Note: You may export your distribution as a picture file (.png), copy it to your computer's clipboard (GeoGebra Desktop) or copy it to the Graphics View (GeoGebra Desktop).

Drag and Drop: In the GeoGebra Desktop Version, you may use *Drag and Drop* in order to transfer the plot of your distribution either to the *Graphics View* or to another application that will accept images. Just position the mouse at the top of the *Probability Calculator* screen and the cursor will change to a hand cursor. This new cursor allows you to drag the plot into *Graphics View 1* or *2* to create a new plot or to drag an image of the plot into another application.

Statistics

Tab *Statistics* allows you to conduct a variety of statistical tests. Just select the test you want to work with from the list available in the drop down menu (e.g. *Z Test of a Mean*) and specify your *Null Hypothesis*, as well as your *Alternative Hypothesis*. Then, adjust the parameters of your test in the provided text boxes and GeoGebra will automatically provide the results of your statistical test.

Probability Calculator: Tab Statistics

3D Graphics View

3D Graphics View User Interface

By default, the *3D Graphics View* is opened next to the *Algebra View*. In addition, either the *Input Bar* is displayed at the bottom of the GeoGebra window (GeoGebra Desktop), or an *Input Field* is integrated directly in the *Algebra View* (GeoGebra Web and Tablet Apps). The *3D Graphics View Toolbar* is displayed at the top of the GeoGebra window, with the *Undo / Redo* buttons in the top right corner.

3D Graphics View

The *3D Graphics View* is part of the *3D Graphics Perspective*, although you may add a *3D Graphics View* window to any *Perspective* at any time by using the *View Menu* or the *Views button* in the *Style Bar*.

Customizing the 3D Graphics View

You can customize the *3D Graphics View* according to the mathematical topic you want to work with. The basic setup can be changed using the *3D Graphics View Style Bar* (e.g. display of coordinate axes, xOy -plane, grid). In addition, the *Preferences Dialog* provides more options to customize the *3D Graphics View*. You may also change the layout of GeoGebra's user interface according to your needs.

Creating Mathematical Objects

Constructions with the Mouse

Using the construction *Tools* available in the *3D Graphics View Toolbar* you can create geometric constructions in the *3D Graphics View* with the mouse. Select any construction tool from the *3D Graphics View Toolbar* and read the tooltip provided in the *3D Graphics View* in order to find out how to use the selected *Tool*.

Note: Any object you create in the *3D Graphics View* also has an algebraic representation in the *Algebra View*.

Example: Select the *Sphere with Center through Point Tool* and click in the *3D Graphics View* twice. The first click creates the center point while the second click creates a sphere and a point on the sphere.

Hint: In order to create a new point in the 3-dimensional space, you need to... click and hold the mouse key (tap and hold) in order to define its x- and y-coordinate, then drag the point up or down in order to change the z-coordinate and release the click (tap) once you reached the desired coordinates.

3D Graphics View Toolbar

The *3D Graphics View Toolbar* provides a wide range of *Tools* that can be operated with the mouse and allow you to create the three-dimensional graphical representations of objects directly in the *3D Graphics View*. Every icon in the *Toolbar* represents a *Toolbox* that contains a selection of related construction *Tools*. In order to open a *Toolbox*, you need to click on the corresponding default *Tool* shown in the *3D Graphics View Toolbar* (GeoGebra Web and Tablet Apps) or on the small arrow in the lower right corner of the *Toolbar* icon (GeoGebra Desktop).

Note: The *Tools* of the *3D Graphics View Toolbar* are organized by the nature of resulting objects or the functionality of the *Tools*. You will find *Tools* that create different types of planes in the *Planes Toolbox* or *Tools* that allow you to create geometric solids in the Geometric Solids Toolbox.

Direct Input using the Input Bar

GeoGebra's *3D Graphics View* supports points, vectors, lines, segments, rays, polygons, and circles in a three-dimensional coordinate system. You may either use the *Tools* provided in the *3D Graphics View Toolbar*, or directly enter the algebraic representation of these objects in the *Input Bar* or *Input Field* of the *Algebra View* (GeoGebra Web and Tablet Apps).

Example: Enter $A = (5, -2, 1)$ into the *Input Bar* or *Input Field* of the *Algebra View* in order to create a point in the three-dimensional coordinate system.

Furthermore, you may now create surfaces, planes, as well as geometric solids (pyramids, prisms, spheres, cylinders, and cones).

Example: Enter $f(x, y) = \sin(x \cdot y)$ in order to create the corresponding surface.

Commands

In addition to the wide range of *Commands* available for the other *Views* of GeoGebra, there also is a selection of *3D Commands* specifically for the *3D Graphics View*.

Example: Let $A = (2, 2, 0)$, $B = (-2, 2, 0)$, $C = (0, -2, 0)$, and $D = (0, 0, 3)$. Input the command `Pyramid[A, B, C, D]` and hit the Enter key in order to create the pyramid with base ABC and apex D .

Moving Mathematical Objects in 3D

Move Tool

Using the *Move Tool* you may drag and drop *Free Points* in the *3D Graphics View*. In order to move a point in the three-dimensional coordinate system, you can switch between two modes by clicking on the point:

- **Mode xOy -plane:** You may move the point parallel to the xOy -plane without changing the z -coordinate.
- **Mode z -axis:** You may move the point parallel to the z -axis without changing the x - and y -coordinates.

Moving Objects using Keyboard Shortcuts

In the *3D Graphics View*, you may use the Page Up key in order to move a selected object up and the Page Down key in order to move a selected object down.

Display of Mathematical Objects

Translation of the Coordinate System

You may translate the coordinate system by using the *Move Graphics View Tool* and dragging the background of the *3D Graphics View* with your pointing device. Thereby, you can switch between two modes by clicking on the background of the *3D Graphics View*:

- **Mode xOy -plane:** You may translate the scene parallel to the xOy -plane.
- **Mode z -axis:** You may translate the scene parallel to the z -axis.

Alternatively you can hold the Shift key and drag the background of the *3D Graphics View* in order to translate the coordinate system. Again, you need to click in order to switch between the two modes while holding the Shift key.

Note: You can go back to the default view by selecting the button *Back to Default View* in the *3D Graphics View Style Bar*.

Rotation of the Coordinate System

You may rotate the coordinate system by using the *Rotate 3D Graphics View Tool* and dragging the background of the *3D Graphics View* with your pointing device.

Alternatively you can right-drag the background of the *3D Graphics View* in order to rotate the coordinate system.

If you want to continue the rotation of the coordinate system when the mouse is released, you may use the option *Start Rotating the View* and *Stop Rotating the View* in the *3D Graphics View Style Bar*.

Note: You can go back to the default rotation by selecting the button *Rotate back to default view* in the *3D Graphics View Style Bar*.

Viewpoint in front of an Object

You may use the *View in front of Tool* in order to view the coordinate system from a viewpoint in front of the selected object.

Zoom

You may use the *Zoom In Tool* and *Zoom Out Tool* in order to zoom in the *3D Graphics View*.

Hint: You may also use the wheel of your mouse to zoom.

3D Graphics View Style Bar

The *3D Graphics View Style Bar* contains buttons to

- show / hide the coordinate axes, the xOy -plane, and a grid in the xOy -plane
- go back to default view
- change the *Point Capturing* settings
- start / stop rotating the view automatically
- adjust the view direction
- choose the type of projection

- open the *Properties Dialog* (GeoGebra Web and Tablet Apps)
- display additional *Views* in the GeoGebra window (GeoGebra Web and Tablet Apps)

Style Bar for Tools and Objects

Depending on the *Tool* or object you select, the buttons in the *Style Bar* adapt to your selection. Please see Style Bar Options for Tools and Objects for more information.

Construction Protocol

GeoGebra Web and Tablet Apps

You can access the interactive **Construction Protocol** by selecting the *Construction Protocol* option in the *View Menu*. The *Construction Protocol* is a table that shows all construction steps. Using the *Navigation Bar* you can also animate the construction steps. To show the *Navigation Bar* at the bottom of the GeoGebra window, select the *Navigation Bar* option in the *View Menu*.

GeoGebra Desktop

You can access the interactive **Construction Protocol** by selecting the *Construction Protocol* option in the *View Menu*. The *Construction Protocol* is a table that shows all construction steps, allowing you to redo a construction step by step. Using the *Navigation Bar* you can also animate the construction steps. To show the *Navigation Bar* at the bottom of the GeoGebra window, right click in the *Graphics View*, then select the *Navigation bar* option in the *Context Menu* displayed.

Navigating and Modifying the Construction Protocol

You may use the keyboard to navigate in the *Construction Protocol*:

- Press the ↑ up arrow on your keyboard to go to the previous construction step.
- Press the ↓ down arrow on your keyboard to go to the next construction step.
- Press the Home key to return to the beginning of the *Construction Protocol*.
- Press the End key to move to the end of the *Construction Protocol*.
- Press the Delete key to delete the selected construction step.

Note: Deleting also affects other objects that depend on the selected object/construction step.

You may also use the mouse in order to navigate in the *Construction Protocol*:

- Double click a row to select a construction step.
- Double click the header of any column to go to the beginning of the *Construction Protocol*.
- Drag and drop a row to move a construction step to another position in the *Construction Protocol*.

Note: This is not always possible due to the dependencies between different objects.

- Right click on a row to open the context menu related to the currently selected object.

Note: You can insert construction steps at any position. Select the construction step below the one you would like to insert a new construction step. Leave the *Construction Protocol* window open while you create a new object. This new construction step is immediately inserted into the selected position of the *Construction Protocol*.

Select the options listed when you select the first left icon of the Construction Protocol toolbar, to decide which informations related to the construction will be shown. The *Breakpoint* option allows you to define certain construction steps as breakpoints, i.e. group several objects together. When navigating through your construction

using the Navigation Bar, selected groups of objects are shown at the same time.

Exporting the Construction Protocol as a Webpage

GeoGebra Desktop allows you to export the *Construction Protocol* as a Webpage. Open the *Construction Protocol* using the related option in the *View Menu*, then select the third icon of the *Construction Protocol's* toolbar (*Export as Webpage*).

In the export window of the *Construction Protocol* you can enter *Title*, *Author*, and a *Date* for the construction and choose whether or not you want to include a picture of the *Graphics View* and the *Algebra View*. In addition, you can also choose to export a *Colorful Construction Protocol*. This means that objects in the *Construction Protocol* will match the color of the corresponding objects in the construction.

Note: The exported HTML file can be viewed with any Internet browser (e.g. Firefox, Internet Explorer, Safari) and edited with many text processing systems (e.g. OpenOffice Writer).

Input Bar

Input Bar and Algebra Input

In GeoGebra Desktop, the *Input Bar* is by default located in the bottom of GeoGebra window. You can show it or hide it using the *View Menu* or can change its position within the GeoGebra window.

In the GeoGebra Web and Tablet Apps an *Algebra Input* is integrated directly into the *Algebra View*. Therefore, the *Input Bar* is not displayed by default if the *Algebra View* is part of the GeoGebra window. However, the *Input Bar* can be shown using the *View Menu*, replacing the *Input Field*.

Both, the *Input Bar* and *Algebra Input* are providing the same functionalities in GeoGebra. However, the *Algebra Input* additionally features an *Equation Editor*, which makes it easier for you to enter equations and expressions.

GeoGebra Web and Tablet Apps GeoGebra Desktop

Algebraic Input and Commands

The *Input Bar* allows you to directly create and redefine mathematical objects in the *Algebra View* by entering or modifying their algebraic representations (e.g. values, coordinates, equations).

Example: The input $f(x) = x^2$ gives you the function f in the *Algebra View* and its function graph in the *Graphics View*.

Note: Always press Enter after typing algebraic input into the *Input Bar*.

Additionally, you may input Commands in order to easily create new or work with existing objects. For more information, please see Geometric Objects and General Objects.

Example: Typing $A=(1, 1)$ and hitting the Enter key creates a free point A with coordinates $(1, 1)$. Create another free point $B=(3, 4)$ in the same way. Then, type in `Line[A, B]` in order to create a dependent line through both points A and B .

You can toggle the focus between the *Input Bar* and the *Graphics View* at any time by pressing the Enter key. This allows you to enter expressions and commands into the *Input Bar* without having to click on it first.

Displaying Input History

After placing the cursor in the *Input Bar* you can use the ↑ up and ↓ down arrow keys on your keyboard in order to navigate through prior input step by step. Hit the Enter key in order to transfer the selected prior input back into the *Input Bar*.

Insert Name, Value, or Definition of an Object into the Input Bar of the GeoGebra Desktop Version

Insert the name of an object: Activate the *Move Tool* and select the object whose name you want to insert into the *Input Bar*. Then, press F5 on your keyboard.

Note: The name of the object is appended to any expression you typed into the *Input Bar* before pressing F5.

Insert the value of an object: To insert an object's value (e.g. coordinates of a point $(1, 3)$, equation $3x - 5y = 12$) into the *Input Bar*, select the *Move Tool* and click on the object whose value you want to insert into the *Input Bar*. Then, press F4 on your keyboard.

Note: The value of the object is appended to any expression you typed into the *Input Bar* before pressing F4.

Insert the definition of an object: There are two ways of inserting an object's definition (e.g. $A = (4, 2)$, $c = \text{Circle}[A, B]$) into the *Input Bar*.

- Hold the Alt key and select the object to insert the object's definition and delete whatever input might have been in the *Input Bar* before.
- Activate the *Move Tool* and select the object whose definition you want to insert into the *Input Bar*. Then, press F3 on your keyboard.

Note: The definition of the object replaces any expression you typed into the *Input Bar* before pressing F3.

Menubar

The Menubar is always situated in the top part of GeoGebra window. For applets it can be switched on and off during export. It contains following menus:

- File Menu
- Edit Menu
- View Menu
- Perspectives Menu (Web and Tablet App Version only)
- Options Menu
- Tools Menu
- Window Menu (Desktop Version only)
- Help Menu

Toolbar

By default the **Toolbar** is located at the top of the GeoGebra window or right below the *Menubar* (GeoGebra Desktop). The *Toolbar* is divided into *Toolboxes*, containing one or more related *Tools*.

Graphics View Toolbar

Tool Help

If you select a *Tool*, a tooltip appears explaining how to use this *Tool*.

Note: When you select the tooltip in the GeoGebra Web and Tablet Apps Version, a web page providing help for the selected tool opens in your browser.

If you are using GeoGebra Desktop, click on the *Help* button in the upper right corner of the GeoGebra window in order to show the *Tool Help Dialog* and get more information about how to use the selected *Tool*. Furthermore, you can access the online help by clicking on the button *Show Online Help* provided in the *Tool Help Dialog*. In addition, you can display the *Toolbar Help* to the right of the *Toolbar* by using the *Layout* option in the *View Menu* and checking *Show Toolbar Help*.

Different Toolbars for Different Views

Each *View* except the Algebra View has its own *Toolbar*, providing *Tools* specific for the *View* you are working with.

Graphics View Toolbar

3D Graphics View Toolbar

CAS View Toolbar

Spreadsheet View Toolbar

Once you start using another *View* within the GeoGebra window, the *Toolbar* changes automatically. If you open another *View* in a separate window, it will have its *Toolbar* attached.

Customizing the Toolbar

Creating a Custom Toolbar

The different *Toolbars* can be customized by selecting *Customize Toolbar...* from the *Tools Menu*.

GeoGebra Desktop

- From the drop-down list select the *Toolbar* of the *View* you want to edit.
- To **remove a Tool or entire Toolbox** from the custom *Toolbar*, select it in the list on the left hand side of the appearing dialog window and click the button *Remove >*.
- To **add a Tool** to your custom *Toolbar*, select it in the right list and click the button *< Insert*.
- To **add a Tool to a new Toolbox**, select a *Toolbox* in the left list and the *Tool* you want to insert in the right list. Then, click *< Insert*. Your *Tool* is inserted as part of a new *Toolbox* below the selected *Toolbox*.
- To **add a Tool to an existing Toolbox**, open the *Toolbox* in the left list and select the *Tool* above the desired position of the new *Tool*. Then, select the *Tool* from the right list and click *< Insert*.
- To **move a Tool from one Toolbox to another**, you need to remove the *Tool* first and then add it to the other *Toolbox*.

GeoGebra Web and Tablet Apps

- In the right upper corner of the GeoGebra window, select the *Toolbar* of the *View* you want to edit.
- To **remove a Tool or entire Toolbox** from the custom *Toolbar*, select it in the list on the left hand side of the appearing dialog window and drag and drop it in the right list.
- To **add a Tool** to your custom *Toolbar*, select it in the right list and drag and drop it in the left list.
- To **add a Tool to a new Toolbox**, select the *Tool* you want to insert in the right list and drag and drop it below a *Toolbox* in the left list. Your *Tool* is inserted as part of a new *Toolbox* below the other *Toolbox*.
- To **add a Tool to an existing Toolbox**, open the *Toolbox* in the left list. Select a *Tool* in the right list and drag and drop that *Tool* below a *Tool* of the opened *Toolbox* in the left list.
- To **move a Tool from one Toolbox to another**, open both *Toolboxes*, select a *Tool* and drag and drop it in the other *Toolbox*.

Note: You can restore the default *Toolbar* using the button [Restore Default Toolbar](#) in the right lower corner of the dialog window.

Changing the Position of the Toolbar

In GeoGebra Desktop, you can change the position of the *Toolbar* using the *Layout* option in the *View Menu*.

Changing the Toolbar in a GeoGebra Applet

The appearance of the *Toolbar* in a Dynamic Worksheets can be set using the *customToolBar* parameter.

Style Bar

What is the Style Bar

The *Style Bar* allows you to quickly and easily change a selection of basic properties of *Views* or objects. In order to open or close the *Style Bar*, either click on the little arrow next to the name of the corresponding *View* in GeoGebra Desktop, or click the *Style Bar Button* in the GeoGebra Web and Tablet Apps. Please note that there are still more properties you can change using the *Properties Dialog*.

Style Bar Options for Views

Each *View* has its own *Style Bar* options.

In GeoGebra Web and Tablet App, depending on the *View* you select, the *Style Bar* can be opened by using one of the following *Style Bar Buttons*, and displays the most useful options to the related *View*.

[Graphics View](#) [3D Graphics View](#) [Algebra View](#) [CAS View](#) [Spreadsheet View](#)

Graphics View Style Bar

Use the *Graphics View Style Bar Button* in order to open the *Graphics View Style Bar* providing the following options:

- **Show / Hide the Axes:** You may show or hide the coordinate axes.
- **Show / Hide the Grid:** You may show or hide the grid (GeoGebra Desktop) or select which type of grid should be shown (GeoGebra Web and Tablet Apps).
- **Go Back to Default View:** If the view on the *Graphics View* was changed before, this button sets the origin of the coordinate axes back to its default position.
- **Change the Point Capturing Settings:** You can choose between the settings *Automatic*, *Snap to Grid*, *Fixed to Grid*, and *Off*. For more information please see *Point Capturing*.
- **Open the Properties Dialog:** Open the *Properties Dialog* for the *Graphics View* or a selected object (GeoGebra Web and Tablet Apps).
- **Display Views:** You may display additional *Views* in the GeoGebra window (GeoGebra Web and Tablet Apps).

3D Graphics View Style Bar

Use the *3D Graphics View Style Bar Button* in order to open the *3D Graphics View Style Bar* providing the following options:

- **Show / Hide the Axes:** You may show or hide the coordinate axes, as well as the xOy -plane.
- **Show / Hide the Grid:** You may show or hide a grid in the xOy -plane.
- **Go Back to Default View:** You may move the coordinate system back to the default view.
- **Change the Point Capturing:** You can choose between the settings *Automatic*, *Snap to Grid*, *Fixed to Grid*, and *Off*. For more information please see *Point Capturing*.
- **Start / Stop rotating the View Automatically:** You may rotate the *3D Graphics View* automatically and can set the direction as well as the speed of the rotation.
- **Adjust the View Direction:** You may choose between the view towards the xOy -plane, xOz -plane, yOz -plane, or rotate back to the default view.
- **Toggle Clipping Box:** You may adapt the size of the clipping box and may choose between the options *None*, *Small*, *Medium* and *Large*. (In the GeoGebra Web and Tablet Apps Version this option can be found in the *Basic* tab of the *Properties Dialog*.)
- **Choose the Type of Projection:** You may choose between a *Parallel Projection*, *Perspective Projection*, *Projection for 3D Glasses*, and *Oblique Projection*.
- **Open the Properties Dialog:** Open the *Properties Dialog* for the *3D Graphics View* or a selected object (GeoGebra Web and Tablet Apps).
- **Display Views:** You may display additional *Views* in the GeoGebra window (GeoGebra Web and Tablet Apps).

Algebra View Style Bar

Use the *Algebra View Style Bar Button* in order to open the *Algebra View Style Bar* providing the following options:

- **Show / Hide Auxiliary Objects:** Toggling this button shows or hides *Auxiliary Objects* in the *Algebra View*.
- **Sort Objects by:** You may organize the mathematical objects according to the following criteria:
 - *Dependency:* The mathematical objects are organized as *Free Objects* and *Dependent Objects*. If you create a new object without using any other existing objects, it is classified as a *Free Object*. If your newly created object was created by using other existing objects, it is classified as a *Dependent Object*.
 - *Object Type:* By default, the mathematical objects are organized by *Object Types* (e.g. Angles, Lines, Points), which are displayed in alphabetical order.

- **Layer:** The mathematical objects are organized according to the layer they are drawn on. For more information see *Layers*.
- **Construction Order:** The mathematical objects are organized by construction order.
- **Display Views:** You may display additional *Views* in the GeoGebra window (GeoGebra Web and Tablet Apps).

CAS View Style Bar

Use the *CAS View Style Bar Button* in order to open the *CAS View Style Bar* providing the following options:

- **Text:** Select the button in order to change the *Text Color* and set the style to **Bold** or *Italic*.
- **Virtual Keyboard:** Display a virtual keyboard in a separate window (GeoGebra Desktop).
- **Display Views:** You may display additional *Views* in the GeoGebra window (GeoGebra Web and Tablet Apps).

Spreadsheet View Style Bar

Use the *Spreadsheet View Style Bar Button* in order to open the *Spreadsheet View Style Bar* providing the following options:

- **Show Input Bar:** Toggling this button shows or hides the *Input Bar* at the top of the *Spreadsheet View* (GeoGebra Desktop).
- **Bold:** Set the font style to bold.
- **Italic:** Set the font style to italic.
- **Text Alignment:** Set the text alignment to *Left*, *Center*, or *Right*.
- **Background Color:** Change the background color of a cell.
- **Cell Borders:** Change the style of the cell borders (GeoGebra Desktop)
- **Properties Dialog:** Open the *Properties Dialog* (GeoGebra Web and Tablet Apps)
- **Display Views:** You may display additional *Views* in the GeoGebra window (GeoGebra Web and Tablet Apps).

Style Bar Options for Tools and Objects

Depending on the *Tool* or existing object you select, the *Style Bar* offers a selection of buttons to change the following properties of either the selected object(s) or the object(s) you are about to create using the selected *Tool*:

- **Point Style:** You may choose between different point styles (e.g. dot, cross, arrow, diamond) and set the point size.
- **Line Style:** You may choose between different line styles (e.g. dashed, dotted) and set the line thickness.
- **Color of Object:** You may set a different color for the selected object.
- **Color and Transparency of Object Filling:** You may choose the color and transparency of the filling for the selected object.
- **Text Style:** You may set the *Text Color*, *Background Color*, Text Style (**bold**, *italic*), and Font Size for a text object.
- **Labelling Settings:** You may choose from the following Labelling settings
 - **Hidden:** No label is displayed.
 - **Name:** Only the name of the object is displayed (e.g. A).
 - **Name & Value:** Name and value of the object are shown (e.g. A = (1, 1)).
 - **Value:** Only the value of the object is displayed (e.g. (1, 1)).
- **Caption:** You may display a caption that differs from the name or value of the object (e.g., if you want to give several objects the same label) and can be specified in the *Properties Dialog*.
- **Absolute Position on Screen:** You may fix an object (e.g. a text box) in the screen so that it is not affected by moving the (3D) *Graphics View* or zooming (GeoGebra Desktop).

- **Properties Dialog:** Open the *Properties Dialog* (GeoGebra Web and Tablet Apps)
- **Display Views:** You may display additional *Views* in the GeoGebra window (GeoGebra Web and Tablet Apps).

Navigation Bar

GeoGebra offers a **Navigation Bar** that allows you to navigate through the construction steps of your GeoGebra file. The *Navigation Bar* is shown at the bottom of the *Graphics View*.

To display it: Right click (Mac OS: Ctrl-click) on an empty spot on the drawing pad, then select the option *Navigation Bar* in the appearing context menu or activate the option *Navigation Bar for Construction Steps* in the *Basic* tab of the Properties Dialog of the *Graphics View*.

The *Navigation Bar* provides a set of navigation buttons, and displays the number of construction steps (e.g. 2 / 7 means that the second step of a total of 7 construction steps is currently displayed):

- button: go back to step 1
- button: go back step by step
- button: go forward step by step
- button: go to the last step
- Play: automatically play the construction, step by step

Note: You may change the speed of this automatic play feature using the text box to the right of the Play button.

- Pause: pause the automatic play feature

Note: This button only appears after activating the Play button.

- button: This button opens the Construction Protocol

Note: This button only appears if option *Button to open construction protocol* is enabled.

File Menu

GeoGebra Web and Tablet App Version

New

Opens a new and empty user interface in the same window.

You are asked if you would like to save the existing construction before opening the new user interface.

Open

Allows you to open a GeoGebra worksheet (file name extension GGB), Style Template File (file name extension GGB), GeoGebra tool (file name extension GGT) or dynamic worksheet (HTM or HTML file produced by GeoGebra) that is saved on GeoGebra or on your computer.

It also allows you to open and insert a GeoGebra file in another one.

Note: In order to open a GeoGebra file you can also drag it to the GeoGebra window and drop it there.

Save

Allows you to save your current construction on your GeoGebra account.

Share

Uploads your worksheet directly to GeoGebra^[20].

Export

Offers several export possibilities:

- ggb
- png
- Animated GIF

GeoGebra Desktop Version

New Window

Keyboard shortcut: Ctrl + N (MacOS: Cmd + N)

Opens a new GeoGebra window that uses the default settings of the GeoGebra user interface.

Note: If you change and save some of these settings, the new GeoGebra window will open using your customized settings.

New

Opens a new and empty user interface in the same GeoGebra window. You are asked if you would like to save the existing construction before opening the new user interface.

Open...

Keyboard shortcut: Ctrl + O (MacOS: Cmd + O)

Opens a GeoGebra worksheet (file name extension GGB), Style Template File (file name extension GGB), GeoGebra tool (file name extension GGT) or dynamic worksheet (HTM or HTML file produced by GeoGebra) that is saved on your computer.

It also allows you to open and insert a GeoGebra file in another one.

Note: In order to open a GeoGebra file you can also drag it to the GeoGebra window and drop it there.

Open from GeoGebra...

Opens the GeoGebra Web page, our collection of free materials, in which you can select and open a file into your GeoGebra window by double-clicking it.

Open Recent (submenu)

Lists up to eight recently opened files.

Save

Keyboard shortcut: Ctrl + S (MacOS: Cmd + S)

Allows you to save your current construction as a GeoGebra file (file name extension GGB) on your computer.

Note: If the file was saved before, this menu item overwrites the old file by using the same file name.

Save as...

Allows you to save your current construction as a GeoGebra file (file name extension GGB). You will be asked to enter a new name for your GeoGebra file before it is saved on your computer.

Share

Lets you upload your worksheet directly to GeoGebra^[20], see also Dynamic Worksheet as Webpage (html)... .

Export (submenu)

Offers several export possibilities:

- Dynamic Worksheet as Webpage (html)...
- Graphics View as Picture (png, eps)...
- Graphics View to Clipboard
- ...and others

Print Preview

Keyboard shortcut: Ctrl + P (MacOS: Cmd + P)

Opens the Print Preview window for the Graphics View. You may specify *Title*, *Author*, *Date* and the *Scale* of your printout (in cm).

Note: Press Enter after a change is made in order to update the preview of your printout.

Close

Keyboard shortcut: Alt + F4 (MacOS: Cmd + W)

Closes the GeoGebra window. If you didn't save your construction prior to selecting *Close*, you are asked if you would like to do so.

Edit Menu

Undo

Keyboard shortcut: Ctrl + Z (MacOS: Cmd + Z)

Allows you to undo your activities step by step.

Note: You can also use the *Undo* button to the right of the Toolbar.

Redo

Keyboard shortcut: Ctrl + Y (MacOS: Cmd + Shift + Z)

Allows you to redo your activities step by step.

Note: You can also use the *Redo* button to the right of the Toolbar.

Copy

Keyboard shortcut: Ctrl + C (MacOS: Cmd + C)

Copies the currently selected objects to GeoGebra's internal clipboard

Paste

Keyboard shortcut: Ctrl + V (MacOS: Cmd + V)

Pastes the objects from GeoGebra's internal clipboard into the currently selected Graphics View. You must then choose where you would like to put them (if you move the object near an existing point it will "lock" onto it).

Object Properties...

Keyboard shortcut: Ctrl + E (MacOS: Cmd + E)

Opens the Properties Dialog which allows you to modify the properties of all objects used in the GeoGebra file.

Select All

Keyboard shortcut: Ctrl + A (MacOS: Cmd + A)

Allows you to select all objects used in your construction.

Select Current Layer

Keyboard shortcut: Ctrl + L (MacOS: Cmd + L)

Allows you to select all objects that are on the same layer as a selected object.

Note: You need to select one object that lies on the desired layer prior to using this menu item.

Select Descendants

Keyboard shortcut: Ctrl + Shift + J (MacOS: Cmd + Shift + J)

Allows you to select all objects that depend on the selected object.

Note: You need to select the parent object prior to using this menu item.

Select Ancestors

Keyboard shortcut: Ctrl + J (MacOS: Cmd + J)

Allows you to select all objects that are ancestors of the selected object, meaning all objects the selected one depends on.

Note: You need to select the dependent object prior to using this menu item.

Invert Selection

Keyboard shortcut: Ctrl + I (MacOS: Cmd + I)

Deselects selected objects and vice versa.

Show / Hide Objects

Keyboard shortcut: Ctrl + G (MacOS: Cmd + G)

Toggles the visibility of selected objects.

Show / Hide Labels

Keyboard shortcut: Ctrl + Shift + G (MacOS: Cmd + Shift + G)

Shows hidden labels for selected objects and hides the shown ones.

Graphics View to Clipboard (Desktop Version only)

Keyboard shortcut: Ctrl + Shift + C (MacOS: Cmd + Shift + C)

Copies the *Graphics View* to your computer's clipboard. Afterwards, you can easily paste this picture into other documents (e.g. word processing document).

Delete

Keyboard shortcut: Delete

Allows you to delete selected objects and their dependent objects.

Note: You need to select the objects you want to delete first (e.g. use a selection rectangle).

View Menu

Algebra

Keyboard shortcut: Ctrl + Shift + A (MacOS: Cmd + Shift + A)

This menu item allows you to show or hide the Algebra View.

Graphics

Keyboard shortcut: Ctrl + Shift + 1 (MacOS: Cmd + Shift + 1)

This menu item allows you to show or hide the Graphics View.

Graphics 2

Keyboard shortcut: Ctrl + Shift + 2 (MacOS: Cmd + Shift + 2)

This menu item allows you to show or hide a *Second Graphics View*.

3D Graphics

Keyboard shortcut: Ctrl + Shift + 3 (MacOS: Cmd + Shift + 3)

This menu item allows you to show or hide the 3D Graphics View.

Spreadsheet

Keyboard shortcut: Ctrl + Shift + S (MacOS: Cmd + Shift + S)

This menu item allows you to show or hide the Spreadsheet View.

CAS

Keyboard shortcut: Ctrl + Shift + K (MacOS: Cmd + Shift + K)

This menu item allows you to show or hide the CAS View.

Probability Calculator

Keyboard shortcut: Ctrl + Shift + P (MacOS: Cmd + Shift + P)

This menu item allows you to show or hide the Probability Calculator.

Construction Protocol

Keyboard shortcut: Ctrl + Shift + L (MacOS: Cmd + Shift + L)

This menu item opens the Construction Protocol.

Keyboard (GeoGebra Desktop only)

This menu item allows you to show or hide the Virtual Keyboard, that you can use with a mouse, and contains the standard keyboard characters, as well as the most used mathematical symbols and operators.

Input Bar

This menu item allows you to show or hide the Input Bar and the Command List at the bottom of the GeoGebra window.

Layout... (GeoGebra Desktop only)

This menu opens the dialog in which you can configure the layout of:

- the Input Bar
- the Toolbar
- the Views
- the Sidebar

Navigation Bar (GeoGebra Web and Tablet App only)

This menu item allows you to show or hide the Navigation Bar at the bottom of the GeoGebra window.

Refresh Views

Keyboard shortcut: Ctrl + F (MacOS: Cmd + F)

This menu item allows you to repaint all views on screen.

Note: You can use this menu item to delete any traces of points or lines in the *Graphics View*.

Recompute all objects

Keyboard shortcut: Ctrl + R (MacOS: Cmd + R)

or F9

This menu item recomputes all objects used in your GeoGebra file.

Note: You can use this menu item to create new random numbers if you used any in your GeoGebra file.

Perspectives

Standard Perspectives

Algebra Geometry Spreadsheet

CAS 3D Graphics Probability

GeoGebra provides a set of standard *Perspectives*. You can easily switch between *Perspectives* using the *Perspectives Sidebar* (GeoGebra Desktop) or the *Perspectives Menu* (GeoGebra Web and Tablet Apps). For more information please check out Selecting a *Perspective*.

Algebra Perspective

The *Algebra Perspective* consists of the Algebra View and the Graphics View. By default, the coordinate axes are shown in the *Graphics View*. Furthermore, the *Graphics View Toolbar* is displayed at the top of the GeoGebra window. Please note that in *GeoGebra 5.0 Desktop* the *Input Bar* is displayed as well at the bottom of the GeoGebra window while in the *GeoGebra 5.0 Web and Tablet Apps* an *Input Field* is integrated into the *Algebra View* (see Input Bar and Input Field).

Geometry Perspective

The *Geometry Perspective* displays the *Graphics View* without the coordinate axes, as well as the *Graphics View Toolbar*.

Spreadsheet Perspective

The *Spreadsheet Perspective* consists of the Spreadsheet View and the *Graphics View*. By default, the coordinate axes are shown in the *Graphics View*. Depending on which of these *Views* is activated, either the *Graphics View Toolbar* or the *Spreadsheet View Toolbar* are shown at the top of the GeoGebra window.

CAS Perspective

The *CAS Perspective* consists of the CAS View and the *Graphics View*. By default, the coordinate axes are shown in the *Graphics View*. Depending on which of these *Views* is activated, either the *Graphics View Toolbar* or the *CAS View Toolbar* are shown at the top of the GeoGebra window.

3D Graphics Perspective

The *3D Graphics Perspective* consists of the 3D Graphics View and the *Algebra View*. By default, the coordinate axes and the xOy -plane are shown in the *3D Graphics View*. Furthermore, the *3D Graphics View Toolbar* is displayed at the top of the GeoGebra window. Please note that in *GeoGebra 5.0 Desktop* the *Input Bar* is displayed as well at the bottom of the GeoGebra window while in the *GeoGebra 5.0 Web and Tablet App* an *Input Field* is integrated into the *Algebra View* (see Input Bar and Input Field).

Probability Perspective

The *Probability Perspective* shows the *Probability Calculator*, which allows you to easily calculate and graph probability distributions.

Customized Perspectives

GeoGebra allows you to customize its user interface according to the mathematical topic you want to work with. You can add additional *Views* or other user interface components (e.g. *Input Bar*, second *Graphics View*) to the standard *Perspectives* by using the *View Menu* or the corresponding *Views Button* in the *Style Bar* (GeoGebra Web and Tablet Apps). Furthermore, you can change the position of some of these elements using the *Layout* option in the *Preferences Dialog* in GeoGebra Desktop or the *Drag View Button* in the Web App. For more information please see Customizing the Layout of the User Interface.

Options Menu

Global options may be changed in the menu **Options**.

Note: To change object settings, please use the Context Menu and Properties Dialog.

Algebra Descriptions

You can set how objects will be represented in Algebra View with this item. There are three possibilities:

- *Value*: show current value of the object.
- *Definition*: show user-friendly description of the object, e.g. "Intersection of a and b ."
- *Command*: show the command that was used to create the object, e.g. "Intersect[a,b]".

Rounding

This menu item allows you to set the number of decimal places or significant figures displayed on screen.

Labeling

You can specify whether the label of a newly created object should be shown or not. You can choose between the settings *Automatic*, *All New Objects*, *No New Objects*, *New Points Only*'.

Note: The setting *Automatic* shows the labels of newly created objects if the *Algebra View* is shown.

Font Size

This menu item determines the font size for labels and text in points (pt).

Note: If you are using GeoGebra as a presentation tool, increasing the font size makes it easier for your audience to read the text, labels, and algebraic input you are using.

Language

GeoGebra is multilingual and allows you to change the current language setting. This affects all input including command names and all output.

Note: No matter which language was selected, the globe icon will lead you back to the *Language* menu. All language names are always displayed in English.

Advanced (GeoGebra Desktop Version only)

This menu item opens the *Advanced* section of the Preferences Dialog.

Note: You can also open this dialog window by right clicking (Mac OS: Ctrl-click) on the Graphics View or Spreadsheet View and selecting *Graphics ...* or *Spreadsheet Options* respectively.

Save Settings

GeoGebra remembers your favorite settings (e.g. settings in the *Options* menu, current *Toolbar* and *Graphics View* settings) if you select Save settings in the Options menu.

Restore Default Settings

You can restore the default settings of GeoGebra using this menu item.

Tools Menu

Create New Tool

Based on an existing construction you can create your own tools in GeoGebra. After preparing the construction of your tool, choose *Create new tool* in the Tools Menu. In the appearing dialog you can specify the output and input objects of your tool and choose a name for the Toolbar icon and corresponding command.

Note: Your tool can be used both with the mouse and as a command in the Input Bar. All tools are automatically saved in your GGB construction file.

Manage Tools

Using the *Manage tools* dialog you can delete a tool or modify its name and icon. You can also save selected tools to a GeoGebra Tools File (GGT). This file can be used later on (File menu, Open) to load the tools into another construction.

Note: Opening a GGT file doesn't change your current construction, but opening a GGB file does.

Customize Toolbar

Opens Customize Toolbar Dialog.

Window Menu

Note: The following only applies to the GeoGebra Desktop Version.

New Window

Keyboard shortcut: Ctrl + N (MacOS: Cmd + N)

See File Menu > New Window.

List of GeoGebra windows

If you have more than one GeoGebra window open, this menu item allows you to switch between these different windows.

Note: This might be helpful when you are using GeoGebra as a presentation tool and want to have several GeoGebra files open at the same time as well as to toggle between them.

Help Menu

Note: Following four menu items work only provided you have access to the internet. You can download the GeoGebra Tutorials^[1] to work offline.

Tutorials

This menu item opens the tutorial part of GeoGebraWiki in your browser.

Help

This menu item opens the HTML-version of the GeoGebra help (the Manual part of GeoGebraWiki) in your browser.

GeoGebra Forum

This menu item opens the GeoGebra User Forum^[2] in your default web browser. You can post and answer GeoGebra-related questions and problems in the GeoGebra User Forum.

Report Bug

This menu item opens the GeoGebra User Forum^[3] in your default web browser and gives you a short instruction on how to report a bug.

About / License

This menu item opens a dialog window that gives you information about the license of GeoGebra and gives credit to people who support the GeoGebra project by contributing in many different ways (e.g. programming, translations).

References

[1] <http://wiki.geogebra.org/en/Tutorials>

[2] <http://help.geogebra.org/>

[3] <http://forum.geogebra.org/bugs/?v=web&lang=en>

Context Menu

The Context Menu provides a quick way to change the behavior or advanced properties of an object. Right click (Mac OS: Ctrl-click) (or long-tap) on an object in order to open its *Context Menu*. For example, it allows you to change the object's algebraic notation (e.g. polar or Cartesian coordinates, implicit or explicit equation) and to directly access features like Rename, Delete, Trace On and Animation On.

Note: If you open the Context Menu for a point in the Graphics View, it gives you the option **Record to Spreadsheet** (only if the Spreadsheet View is opened). Once selected, this feature allows you to record the coordinates of the point in the Spreadsheet View if it is moved.

Note: Selecting Properties in the *Context Menu* opens the Properties Dialog, where you can change the properties of all objects used.

Customize the Settings

GeoGebra allows you to change and save settings using the Options Menu. For example, you may change the Angle Unit from Degree to Radians (GeoGebra Desktop Version only), or change the Point Style, Checkbox Size, and Right Angle Style. In addition, you may change how Coordinates are displayed on screen and which objects are labeled (Labeling).

Please see the section about the *Options menu* for more information.

You can save your customized settings by selecting item *Save Settings* from the Options menu. After doing so, GeoGebra will remember your customized settings and use them for every new GeoGebra file you create.

Note: You may restore the default settings by selecting *Restore Default Settings* from the *Options menu*.

Note: If you use GeoGebra as a presentation tool, you might want to increase the *Font Size* (*Options menu*) so your audience can easily read text and labels of objects.

Export Graphics Dialog

Note: The following only applies to **GeoGebra Classic 5**. For newer versions please see the `ExportImage()` command

This dialog is accessible via the *Export* submenu of File Menu (item *Graphics View as Picture (png, eps)....*)

Keyboard shortcut: Ctrl + Shift + U (Mac OS: Cmd + Shift + U)

This dialog allows you to save GeoGebra's Graphics View as a picture file on your computer. In the appearing dialog window, you can select the picture file *Format*, change the *Scale* (in cm) and *Resolution* (in dpi) of the picture, and set the image as *Transparent*.

Note: If you create points called *Export_1* and *Export_2* then these will define the rectangle that is exported, otherwise just the visible *Graphics View* is exported

When exporting the *Graphics View* as a picture you can choose out of the following formats:

PNG – Portable Network Graphics

This is a pixel graphics format. The higher the resolution (dpi), the better the quality (300dpi will usually suffice). PNG graphics should not be scaled subsequently to avoid a loss of quality.

PNG graphic files are well suited for the use on web pages (HTML) and in word processing documents.

Note: Whenever you insert a PNG graphic file into a word processing document (menu *Insert, Image* from file) make sure that the size is set to 100 %. Otherwise the given scale (in cm) would be changed.

EPS – Encapsulated Postscript

This is a vector graphics format. EPS pictures may be scaled without loss of quality. EPS graphic files are well suited for the use with vector graphics programs (e.g. Corel Draw) and professional text processing systems (e.g. LaTeX).

The resolution of an EPS graphic is always 72dpi. This value is only used to calculate the true size of an image in centimeters and has no effect on the image's quality.

Note: The transparency effect with filled polygons or conic sections is not possible with EPS. Objects can only be either 100% opaque or transparent.

PDF – Portable Document Format

(see EPS format above)

Note: In SVG and PDF export you have the option to export text as editable text or shapes. This stores the text either as text (this lets you edit the text in e.g. InkScape) or as Bézier curves (this guarantees that the text looks the same even if the correct font is not installed).

SVG – Scalable Vector Graphic

(see EPS format above)

EMF – Enhanced Metafile

(see EPS format above)

Export Worksheet Dialog

Note: The following only applies to the GeoGebra Desktop Version.

GeoGebra allows you to create interactive webpages, so called Dynamic Worksheets, from your files. In the File Menu, you need to select item *Export*, then click on item *Dynamic Worksheet as Webpage (html)*. This opens the export dialog window for Dynamic Worksheets.

Upload to GeoGebra

Under this tab you can enter a title for your construction, a text above and below the construction (e.g. a description of the construction and some tasks), and then Upload it to GeoGebra [20].

Note: When you upload a file to GeoGebra, you will be asked to create an account and/or login first.

Properties Dialog

The Properties Dialog allows you to modify properties of objects (e. g. size, color, filling, line style, line thickness, visibility) as well as automate some object actions using Javascript or GeoGebra Script.

How to Open the Properties Dialog

You can open the Properties Dialog in several ways:

- Select item *Object Properties* from the Edit Menu.
- Right-click (Mac OS: Ctrl-click) on an object and select *Object Properties* from the appearing Context Menu.
- Select the *Properties* button from the *Style Bar* (GeoGebra Web and Tablet Apps Version).
- Select item *Properties* in the upper right corner of the GeoGebra window (GeoGebra Desktop Version).
- Select the Move Tool and double click on an object in the Graphics View. In the appearing *Redefine Dialog* window, click on the button *Object Properties* (GeoGebra Desktop).

Organization of Objects

In the **Properties Dialog** objects are organized by types (e.g. points, lines, circles) in the list on the left hand side, which makes it easier to handle large numbers of objects. You need to select one or more objects from this list in order to change their properties.

Note: By selecting a heading in the list of objects (e.g. *Point*, *Line*) you can select all objects of this type and then quickly change the properties for all these objects.

Modifying Object Properties

You can modify the *Properties* of selected objects using the tabs of the *Properties Dialog* window.

- *Basic*
- *Text*
- *Color*
- *Position*
- *Style*
- *Algebra*
- *Advanced*
- *Scripting*

Note: Depending on the selection of objects in the list, a different set of tabs may be available.

Close the **Properties Dialog** when you are done with changing properties of objects.

Redefine Dialog

Redefining objects is a very versatile way to change a construction. Please note that this may also change the order of the construction steps in the Construction Protocol.

Note: The redefined element can only depend on elements defined earlier in the construction order, so you may need to change order of the elements in Construction Protocol.

In GeoGebra, an object may be redefined in different ways:

- Select Move Tool and double click on any object in the Algebra View.
 - For free objects an editing field is opened allowing you to directly change the algebraic representation of the object. Hit the Enter key in order to apply these changes.
 - For dependent objects the Redefine dialog is opened allowing you to redefine the object.
- Select Move Tool and double click on any object in the Graphics View. This opens the Redefine dialog and allows you to redefine the object.
- Change any object by entering its name and the new definition into the Input Bar.
- Open the Properties Dialog and change the definition of an object on tab *Basic*.

Note:

Fixed objects cannot be redefined. In order to redefine a fixed object, you need to free it first using tab *Basic* of the *Properties Dialog*.

Note:

You can also redefine existing objects in the *Input Bar*. For example type `a : Segment [A, B]` to redefine *a* to be a segment.

Examples

Example: In order to place an existing free point A onto an existing line h, you first need to double click on the point A to open the Redefine dialog window. Then change the definition to `Point [h]` in the appearing text field and press Enter. To remove point A from this line and make it free again, you need to redefine it to some free coordinates, e.g. (1, 2).

Example: Another example is the conversion of a line h through two points A and B into a segment. Open the Redefine dialog for line h and change `Line [A, B]` into `Segment [A, B]`

Tool Creation Dialog

First, create the construction your tool should be able to create later on. In the Tools menu, select *Create New Tool* in order to open the corresponding dialog box. Now you need to fill in the three tabs *Output Objects*, *Input Objects*, and *Name and Icon* in order to create your custom tool.

Example: Create a Square-tool that creates a square whenever you click on two existing points or on two empty spots in the Graphics View.

- Construct a square starting with two points A and B. Construct the other vertices and connect them with the tool `Polygon` to get the square `poly1`.
- Select *Create New Tool* in the *Tools Menu*.
- Specify the Output Objects: Click on the square or select it from the drop down menu. Also, specify the edges of the square as Output Objects.
- Specify the Input Objects: GeoGebra automatically specifies the Input Objects for you (here: points A and B). You can also modify the selection of input objects using the drop down menu or by clicking on them in your construction.
- Specify the Tool Name and Command Name for your new tool.

Note: The Tool Name will appear in GeoGebra's Toolbar, while the Command Name can be used in GeoGebra's Input Bar.

- You may also enter text to be shown in the Toolbar Help.
- You can also choose an image from your computer for the Toolbar icon. GeoGebra resizes your image automatically to fit on a Toolbar button.

Note: Outputs of the tool are not moveable, even if they are defined as `Point [<Path>]`. In case you need moveable output, you can define a list of commands and use it with Execute Command.

Keyboard Shortcuts

Key	Shortcut	Desktop (Windows)	Web (Windows)	Web (Mac)	Shortcut Mac Desktop	Action
A	Ctrl + A	X	X	X	Cmd + A	Select All
A	Ctrl + Shift + A	X	X	X		View/Hide Algebra Window
A	Alt + A	X	X	X	Alt + A	alpha α (Press Shift for upper-case: A)
B	Alt + B	X	X	X	Alt + B	beta β (Press Shift for upper-case: B)
B	Ctrl + Shift + B	X	X	X	Cmd + Shift + B	Export ggbBase64 string to clipboard
C	Ctrl + C	X	X	X	Cmd + C	Copy
C	Ctrl + Alt + C	X			Cmd + Alt + C	Copies values (spreadsheet)
C	Ctrl + Shift + C	X			Cmd + Shift + C	Copy Graphics View to clipboard
D	Ctrl + D	X	X		Cmd + D	Toggle value/definition/command
D	Ctrl + Shift + D	X	X			Toggle <i>Selection Allowed</i> for all "non-essential"/free geometric objects
D	Alt + D	X		X	Alt + D	delta δ (Press Shift for upper-case: Δ)
E	Ctrl + E	X	X	X	Cmd + E	Open Object Properties View
E	Ctrl + Shift + E	X	X	X	Cmd + Shift + E	Open/Close Object Properties View
E	Alt + E	X				Euler e
F	Ctrl + F	X	X	X	Cmd + F	Refresh Views
F	Alt + F	X				phi φ (Press Shift for upper-case: Φ)
G	Ctrl + G	X	X	X	Cmd + G	Show/Hide selected objects
G	Ctrl + Shift + G	X	X	X	Cmd + Shift + G	Show/Hide labels of selected objects
G	Alt + G	X	X	X		gamma γ (Press Shift for upper-case: Γ)
I	Alt + I	X	X	X	Alt + I	imaginary unit $i = \sqrt{-1}$
J	Ctrl + J	X	X	X	Cmd + J	Select ancestors
J	Ctrl + Shift + J	X	X	X	Cmd + Shift + J	Select descendants
K	Ctrl + Shift + K	X	X	X	Cmd + Shift + K	View/Hide CAS View
L	Alt + L	X	X	X		lambda λ (Press Shift for upper-case: Λ)
L	Ctrl + Shift + L	X	X	X	Cmd + Shift + L	View/Hide Construction Protocol
M	Ctrl + Shift + M	X	X	X	Cmd + Shift + M	Export HTML code string to clipboard
M	Alt + M	X	X	X	Alt + M	mu μ (Press Shift for upper-case: M)
N	Ctrl + N	X			Cmd + N	New Window
N	Ctrl + Shift + N	X			Cmd + Shift + N	Open next window (or next ggb file in folder)
N	Ctrl + Shift + Alt + N				Cmd + Shift + Alt + N	Open previous window
O	Ctrl + O	X	X		Cmd + O	Open New File
O	Alt + O	X	X	X		degree symbol \circ
P	Ctrl + P	X	X		Cmd + P	Print Preview (Desktop) / Print Menu (Web)
P	Ctrl + Shift + P	X		X	Cmd + Shift + P	Open Probability Calculator

P	Alt + P	X	X	X	Alt + P	pi π (Press Shift for upper-case: Π)
Q	Ctrl + Q	X	X			Select ancestors (deprecated)
Q	Ctrl + Shift + Q	X	X			Select descendants (deprecated)
Q				X	Cmd + Q	Quit GeoGebra
R	Ctrl + R	X	X	X	Cmd + R	Recompute all objects (including random numbers)
R	Alt + R	X	X	X		Square-root symbol: $\sqrt{}$
S	Ctrl + S	X	X	X	Cmd + S	Save
S	Ctrl + Shift + S	X	X	X	Cmd + Shift + S	View spreadsheet
S	Alt + S	X		X		sigma σ (Press Shift for upper-case: Σ)
T	Ctrl + Shift + T	X			Cmd + Shift + T	Export as PSTricks
T	Alt + T	X	X	X		theta θ (Press Shift for upper-case: Θ)
U	Alt + U	X	X		Alt + ,	infinity ∞
U	Ctrl + Shift + U	X			Cmd + Shift + U	Open Graphic Export Dialog
V	Ctrl + V	X	X		Cmd + V	Paste
W					Cmd + W	Quit GeoGebra
W	Ctrl + Shift + W	X			Cmd + Shift + W	Export Dynamic Worksheet
W	Alt + W	X	X	X		omega ω (Press Shift for upper-case: Ω)
Y	Ctrl + Y	X	X	X	Cmd + Y	Redo
Z	Ctrl + Z	X	X	X	Cmd + Z	Undo
Z	Ctrl + Shift + Z	X	X	X	Cmd + Shift + Z	Redo
0	Alt + 0	X	X	X		to the power of 0
1	Ctrl + 1	X	X	X	Cmd + 1	Standard font size, line thickness, and point size
1	Alt + 1	X	X	X		to the power of 1
1	Ctrl + Shift + 1	X	X	X	Cmd + Shift + 1	View/Hide Graphics View 1
2	Ctrl + 2	X	X	X	Cmd + 2	Increase font size, line thickness, and point size
2	Alt + 2	X	X	X		to the power of 2
2	Ctrl + Shift + 2	X	X	X	Cmd + Shift + 2	View/Hide Graphics View 2
3	Ctrl + 3	X	X	X	Cmd + 3	Black/white mode
3	Alt + 3	X	X	X		to the power of 3
4	Alt + 4	X	X	X		to the power of 4
5	Alt + 5	X	X			to the power of 5
6	Alt + 6	X	X			to the power of 6
7	Alt + 7	X	X			to the power of 7
8	Alt + 8	X	X			to the power of 8
9	Alt + 9	X	X			to the power of 9
-	-	X	X	X		Decrease selected slider/number Move selected point along path/curve
-	Ctrl + -	X	X	X		Zoom out

-	Alt + -	X				superscript minus
+	+	X	X	X		Increase selected slider/number Move selected point along path/curve
+	Ctrl + +	X	X	X		Zoom in
+	Alt + +	X		X	Alt + +	\oplus (xor)
=	=	X	X	X		Increase selected slider/number Move selected point along curve
=	Ctrl + =	X	X	X		Zoom in
=	Alt + =	X	X	X		\oplus (xor)
<	Alt + <	X		X	Alt + <	less-than-or-equal-to \leq
, (comma)	Alt + ,	X	X			less-than-or-equal-to \leq
>	Alt + >	X			Alt + Shift + >	greater-than-or-equal-to \geq
. (period)	Alt + .	X	X			greater-than-or-equal-to \geq
F1	F1	X			F1	Help
F2	F2	X			F2	Start editing selected object
F3	F3	X			F3	Copy definition of selected object to the Input Bar
F4	F4	X			F4	Copy value of selected object to the Input Bar
F4	Alt + F4	X	X			Quit GeoGebra
F5	F5	X			F5	copy name of selected object to the Input Bar
F9	F9	X	X	X	F9	Recompute all objects (including random numbers)
Enter	Enter	X	X	X	Enter	Toggle input between Graphics View and Input Bar
Tab	Ctrl + Tab	X				Cycle the focus round the open views
Left Click	Left Click	X	X	X	Left Click	(current mode)
Left Click	Alt+Left Click	X			Alt+Left Click	copy definition to input bar
Left Click	Alt+Left Drag				Alt+Left Drag	create list of selected objects in input bar
Right Click	Right click in Graphics View					Fast drag mode (drag on object) Selection rectangle Open menu (click on object) Open Preferences menu (click not on object)
Right Click	Shift+ Right Drag					Zooms without preserving the aspect ratio
Scroll Wheel	Scroll Wheel	X	X	X	Scroll Wheel	Zoom in / out (Application)
Scroll Wheel	Shift+Scroll Wheel	X	X	X	Shift+Scroll Wheel	Zoom in / out (Applet)
Scroll Wheel	Alt+Scroll Wheel	X	X	X	Alt+Scroll Wheel	Accelerated zoom in / out
Delete	Delete	X	X			Delete current selection
Backspace	Backspace	X	X	X	Backspace	Delete current selection

Up arrow ↑	↑	X	X	X	↑	Increase selected slider/number Move selected point up 3D Graphics Increase y-coordinate of selected point Go to older entry in Input Bar history Go up in construction protocol (only Desktop) Move active Graphics view up
Up arrow ↑	Ctrl + ↑	X	X			x10 speed multiplier Spreadsheet: go to top of current block of cells (or go up to next defined cell)
Up arrow ↑	Shift + ↑	X	X	X	Shift + ↑	x0.1 speed multiplier, or rescale y-axis if no objects selected
Up arrow ↑	Alt + ↑	X	X	X	Alt + ↑	x100 multiplier
Right arrow →	→	X	X	X	→	Increase selected slider/number Move selected point right 3D Graphics Increase x-coordinate of selected point Go up in construction protocol (only Desktop) Move active Graphics view right
Right arrow →	Ctrl + →	X	X			x10 speed multiplier Spreadsheet: go to right of current block of cells (or go right to next defined cell)
Right arrow →	Shift + →	X	X	X	Shift + →	x0.1 speed multiplier, or rescale x-axis if no objects selected
Right arrow →	Alt + →	X	X	X	Alt + →	x100 multiplier
Left arrow ←	←	X	X	X	←	Decrease selected slider/number Move selected point left 3D Graphics Decrease x-coordinate of selected point Go down in construction protocol (Desktop only) Move active Graphics view left
Left arrow ←	Ctrl + ←	X	X			x10 speed multiplier Spreadsheet: go to left of current block of cells (or go left to next defined cell)
Left arrow ←	Shift + ←	X	X	X	Shift + ←	x0.1 speed multiplier, or rescale x-axis if no objects selected
Left arrow ←	Alt + ←	X	X	X	Alt + ←	x100 multiplier
Down arrow ↓	↓	X	X	X	↓	Decrease selected slider/number Move selected point down 3D Graphics Decrease y-coordinate of selected point Go to newer entry in Input Bar history Go down in construction protocol (only Desktop) Move active Graphics view down

Down arrow ↓	Ctrl + ↓	X	X			x10 speed multiplier Spreadsheet: go to bottom of current block of cells (or go down to next defined cell)
Down arrow ↓	Shift + ↓	X	X	X	Shift + ↓	x0.1 speed multiplier, or rescale y-axis if no objects selected
Down arrow ↓	Alt + ↓	X	X	X	Alt + ↓	x100 multiplier
Home	Home	X	X			Go to first item in construction protocol (only Desktop) Spreadsheet: go to the first column left
PgUp ↑	□	X	X			Go to first item in construction protocol (only Desktop) 3D Graphics Increase z-coordinate of selected point
End	End	X	X			Go to last item in construction protocol (only Desktop) Spreadsheet: go to the next row with input below
PgDn ↓	□	X	X			Go to last item in construction protocol (only Desktop) 3D Graphics Decrease z-coordinate of selected point

In addition, use Alt + Shift (Mac OS X Ctrl + Shift) to get upper-case Greek characters.

Settings Dialog

Note: The following only applies to the GeoGebra Desktop Version.

This dialog is available by selecting the *Advanced* item in Options Menu, or by clicking the *Preferences* icon on the Toolbar. It is divided into different sections, which are shown depending on active objects and Views: *Properties*, *Graphics*, *CAS*, *Spreadsheet*, *Layout*, *Defaults*, and *Advanced*.

Properties

In this section the properties of objects can be changed. Clicking on the related icon displays the Properties Dialog of the object.

Graphics

This section is displayed only if the Graphics View is active and allows you to define its settings, e.g. background colour, axes and grid. See [Customizing the Graphics View](#) for further details.

CAS

This section is displayed only if the CAS View is active, and allows you to:

- set a timeout for the *CAS* calculations, in seconds
- set how to show rational exponents

Spreadsheet

This section is displayed only if the Spreadsheet View is active. Here you can customize the spreadsheet, setting preferences for the input bar, grid lines, column/row headers and scrollbars. You can also enable using buttons, checkboxes and tooltips.

Layout

This section allows you to customize the layout of the main components of your GeoGebra window, setting the position of the *Input Bar*, the Toolbar, and the sidebar, as well as the Style Bar and title bar visibility.

Defaults

This section allows you to customize the appearance and style of all the mathematical objects in GeoGebra. The list of all available objects is shown on the left of the dialog window, while on the right the Properties Dialog tabs are shown, where you can set visibility, colour, style and algebraic settings for all your mathematical objects.

Advanced

This section contains the following advanced global settings:

- *Angle Unit*: switch between *Degree* and *Radians*
- *Right Angle Style*: choose the symbol for a right angle
- *Coordinates*: define how coordinates are displayed algebraically
- *Continuity*: if Continuity is *On*, GeoGebra tries to set new calculated points near the original ones
- *Use Path and region Parameters*: you can turn On and Off this option
- *Virtual Keyboard*: options to set the *virtual keyboard language*, its width,height and opacity
- *Checkbox Size*: switch between *regular* and *large* checkboxes
- *Fonts Size*: set the Menu font size
- *Tooltips*: set the *tooltip language* and a timeout for tooltips
- *Language*: use digits and point names specified for your language

Virtual Keyboard

The Virtual Keyboard of the GeoGebra Desktop Version is a semi-transparent keyboard that is displayed on the screen when the corresponding menu item *Keyboard* in the View Menu is selected.

It contains the standard keyboard characters, as well as the most used mathematical symbols and operators, and can be used with a mouse or other pointing devices.

This makes the Virtual Keyboard particularly useful when using GeoGebra for presentations or with multimedia interactive whiteboards.

Tool Manager Dialog

You can save your custom tools so you can reuse them in other GeoGebra constructions. In the Tools Menu, select *Manage Tools* to open this dialog. Then, select the custom tool you want to save from the appearing list. Click on button *Save As...* in order to save your custom tool on your computer.

Note: User defined tools are saved as files with the file name extension GGT so you can distinguish custom tool files from usual GeoGebra files (GGB).

This dialog also allows you to remove or modify tools. If you decide to modify a tool, a new GeoGebra window appears. The input objects are listed as free objects in it. If you have finished your changes, you can save the tool via option *Create new tool* in Tools Menu. Keep the old name to overwrite the tool. To overwrite a tool which was already used, the types of input and output objects must stay the same.

Sidebar

In the *Sidebar* of the GeoGebra Desktop Version you can easily switch between different views, without selecting each individually.

Six different standard *perspectives* are available:

Algebra: Algebra View and Graphics View (with axes) are displayed.

Geometry: Just Graphics View (with grid) is displayed.

Spreadsheet: Spreadsheet View and Graphics View are displayed.

CAS: CAS View and Graphics View are displayed.

3D Graphics: Algebra View and 3D Graphics View (with axes) are displayed.

Probability: The Probability Calculator is displayed.

Note: In the GeoGebra Web and Tablet App the Perspectives can be switched via the Perspectives Menu in the Menubar.

Open Dialog - Insert File

Insert File

In the **GeoGebra Desktop Version** (Windows and Linux) you can insert a GeoGebra file into the current one, keeping the existing objects. In the menu bar, click *File - Open*, then select *Insert File (.ggb)* from the dropdown list displayed and click on the .ggb file you want to insert in the current construction.

Some objects of the inserted file may be renamed to ensure that all the objects in the current construction have a unique name.

Open Dialog - Style Templates

Apply Template

In the GeoGebra Desktop Version you can create a custom style worksheet (template), containing the visual settings of one or more objects of the same or different types, so that those settings can be then applied to objects in other GeoGebra files.

How to make a template file

Create a GeoGebra file (which will be saved later, using a proper name, e.g. *style.ggb*). In the advanced options set the styles for default elements.

How to apply a template file to a .ggb file

Open or create the file that you want to apply the style template to.

Save your file, then in the menu bar click *File - Open*

Then select item *Apply template (.ggb)* from the *File type* dropdown list, shown in the appearing dialog window. Select the file containing the style template (referring to the previous example, select the *style.ggb* file) to change object defaults according to the template file.

Accessibility

Note: For examples of accessible applets and more information see GeoGebra Accessibility ^[1].

When designing applets it is important to allow (as much as possible) for all students to access them.

Fontsize, Color & Contrast

Using large fonts with a good contrast is important. Select a large font-size in Options -> Fontsize before you start designing your applet. You can also use the keyboard shortcut Ctrl + 2 to make all fonts bigger and all lines thicker. Ctrl + 3 will change the default for all new objects to be Black and unfilled.

Ideally use dark colors on a white background and thick lines. Pure Red ^[2] and Green ^[3] don't have a very good contrast so you should use some darker versions, eg Blue ^[4], Dark Green ^[5], Dark Red ^[6]

If you use colors to distinguish otherwise-similar objects, then consider also using dashed lines for one of them. You can also find websites where you can check whether your color scheme is good, for example <http://www.vischeck.com/>

Sliders

Make sliders as long as possible so that they are usable by students whose fine motor-skills are less good. Consider adding "decrement" and "increment" Buttons at each end of the slider as well.

Keyboard Shortcuts

Students can use Tab to cycle round the objects in a worksheet you have designed. It is important that you **uncheck "Selection Allowed"** in Object Properties -> Advanced for the objects you don't want to be selected. There is a keyboard shortcut Ctrl + Shift + D in the desktop version that will toggle this property for objects **other than** Buttons / Sliders / Checkboxes / Points / Input Boxes.

The arrow keys Left, Right, Up, Down can be used to move Sliders and Points once they are selected.

Space can be used to activate a Button, toggle a Checkbox or start/stop a slider animating

Escape can be used to leave a worksheet and Enter to go back in (useful if you have several GeoGebra worksheets on one page)

+ and – can be used to move a point on a path (eg on a circle)

GeoGebra Applets are also fully touch-enabled so students can use applets on a special large tablet if necessary.

Tooltips

You can provide custom tooltips for certain objects by using the *Caption* property, and changing how the tooltip is shown: <https://wiki.geogebra.org/en/Tooltips>

Screen Readers

You can have some text attached to the Graphics Views so that it is read by a screen reader.

If you make a text object in GeoGebra called altText, altText2, altText3D then it will be attached to Graphics View 1, Graphics View 2, Graphics View 3D respectively.

References

- [1] <https://www.geogebra.org/m/r2EF8uRx>
- [2] http://snook.ca/technical/colour_contrast/colour.html#fg=FF0000,bg=FFFFFF
- [3] http://snook.ca/technical/colour_contrast/colour.html#fg=00FF00,bg=FFFFFF
- [4] http://snook.ca/technical/colour_contrast/colour.html#fg=0000FF,bg=FFFFFF
- [5] http://snook.ca/technical/colour_contrast/colour.html#fg=226600,bg=FFFFFF
- [6] http://snook.ca/technical/colour_contrast/colour.html#fg=990033,bg=FFFFFF

Publishing

Creating Pictures of the Graphics View

You can either save a picture of the Graphics View in a file or copy it to your clipboard.

Saving as File

Note: The full *Graphics View* will be saved as a picture, unless points *Export_1* and *Export_2* are defined (see below).

If your construction does not use all the available space in the *Graphics View*, you might want to...

- ...use tools Move Graphics View Tool, Zoom In Tool and/or Zoom Out Tool in order to place your construction in the upper left corner of the *Graphics View*. Afterwards, you may reduce the size of the GeoGebra window by dragging one of its corners.
- ... use the selection rectangle in order to specify which part of the *Graphics View* should be exported and saved as a picture. Therefore you have to hold the right mouse button and drag the cursor over that part of the *Graphics View*, that you want to export.

You can also create points called *Export_1* and *Export_2*, which will be used to define diagonally opposite corners of the export rectangle.

Note: Points *Export_1* and *Export_2* must be within the visible area of the *Graphics View*.

In the File Menu of the GeoGebra Desktop Version, select *Export* before clicking on *Graphics View as Picture*. In the appearing dialog window you may specify the *Format*, *Scale* (in cm), and the *Resolution* (in dpi) of the output picture file. If you are using the GeoGebra Web and Tablet Apps Version, then you can export the *Graphics View* or a selected part of it by selecting *Export* in the File Menu. There you can choose between the formats PNG and animated GIF.

Note: The true size of the exported image is shown at the bottom of the export window just above the buttons, both in centimeters and pixel.

Please find more information about the different picture files available in section Export Graphics Dialog.

Copying the Graphics View to Clipboard

In the GeoGebra Desktop Version you can also copy the *Graphics View* to the *Clipboard* of your computer in one of the following ways:

- In the menu *Edit*, select the option *Graphics View to Clipboard*.
- In the menu *File*, first select *Export* and then select the option *Graphics View to Clipboard*.
- In the export dialog of the *Graphics View* (*File – Export – Graphics View as Picture (png, eps)...*), click the *Clipboard* button.

This feature copies a screenshot of the *Graphics View* to your system's clipboard as a PNG picture, that can be then pasted into other documents (e.g. a word processing document).

Notes: The *File – Export – Graphics View as Picture (png, eps)...* option allows you to define the dimension (scale or pixels) of the exported image. The copy to Clipboard feature is not available for Mac Os.

Embedding in Webpages

To create a webpage including interactive GeoGebra construction, you have to upload the construction to GeoGebra and use the Embed button in the teacher page.

You can then paste the resulting code into your online content management system or you can save it as .html file using a text editor like Notepad and add texts above and below the construction.

Advanced settings

The Advanced Settings button allows you to change the functionality of the dynamic construction (e.g. show a reset icon and browser features) as well as to modify the user interface shown in the interactive applet (e.g. show the Toolbar, modify height and width, enabling saving and printing, and others).

Note: If the size of your applet is too big to fit on a computer screen with standard resolution (1024 x 768), you may want to resize it before the actual export as a Dynamic Worksheet.

Functionality:

- *Enable right click, zooming and keyboard editing features:* By selecting this feature you will be able to right click objects or the drawing pad in order to access the features of the context menu (e.g. show / hide object or label, trace on / off, Properties dialog). It is also possible to use the common keyboard shortcuts.
- *Enable dragging of labels:* By selecting this feature you are able to drag labels of points or objects.
- *Show icon to reset construction:* A reset icon is displayed in the upper right corner of the interactive applet allowing your students to reset the interactive figure to its initial state.

User interface:

- *Show menubar:* The menubar is displayed within the interactive applet.
- *Show toolbar:* The toolbar is displayed within the interactive applet allowing to use the geometry tools.
- *Show inputbar:* The input field is displayed at the bottom of the interactive applet allowing to use algebraic input and commands for explorations.

Note: If you reduce the size of the applet important parts of the dynamic worksheets might be invisible for users.

Hint: If you include the menubar, toolbar, or input field you might want to adjust the height of the interactive applet.

Embedding to CMS, VLE (Moodle) and Wiki

To create a webpage including interactive GeoGebra construction, you have to upload the construction to GeoGebra and use the Embed button in the teacher page.

You can then paste the resulting code into your online content management system or you can save it as .html file using a text editor like Notepad and add texts above and below the construction.

Advanced settings

The Advanced Settings button allows you to change the functionality of the dynamic construction (e.g. show a reset icon and browser features) as well as to modify the user interface shown in the interactive applet (e.g. show the Toolbar, modify height and width, enabling saving and printing, and others).

Note: If the size of your applet is too big to fit on a computer screen with standard resolution (1024 x 768), you may want to resize it before the actual export as a Dynamic Worksheet.

Functionality:

- *Enable right click, zooming and keyboard editing features:* By selecting this feature you will be able to right click objects or the drawing pad in order to access the features of the context menu (e.g. show / hide object or label, trace on / off, Properties dialog). It is also possible to use the common keyboard shortcuts.
- *Enable dragging of labels:* By selecting this feature you are able to drag labels of points or objects.
- *Show icon to reset construction:* A reset icon is displayed in the upper right corner of the interactive applet allowing your students to reset the interactive figure to its initial state.

User interface:

- *Show menubar:* The menubar is displayed within the interactive applet.
- *Show toolbar:* The toolbar is displayed within the interactive applet allowing to use the geometry tools.
- *Show inputbar:* The input field is displayed at the bottom of the interactive applet allowing to use algebraic input and commands for explorations.

Note: If you reduce the size of the applet important parts of the dynamic worksheets might be invisible for users.

Hint: If you include the menubar, toolbar, or input field you might want to adjust the height of the interactive applet.

Export to LaTeX (PGF, PStricks) and Asymptote

Note: The following only applies to the GeoGebra Desktop Version.

Export - Graphics View as Animated GIF...

When the dynamic construction depends on a slider, this menu item allows you to save the Graphics View as an Animated GIF. Just drag a selection rectangle around the portion of the construction you need to export (or resize the GeoGebra window), then select File -> Export -> Graphics View as Animated GIF. A dialog will let you choose the animating slider name, as well as the timing between frames and the option to display the animation as loop.

Export - Graphics View as PStricks...

Keyboard shortcut: Ctrl + Shift + T (MacOS: Cmd + Shift + T)

This menu item allows you to save the *Graphics View* as a PStricks^[1] picture file, which is a LaTeX picture format.

Export - Graphics View as PGF/TiKZ...

This menu item allows you to save the *Graphics View* as a PGF^[2] picture file, which is a LaTeX picture format.

Export - Graphics View as Asymptote...

This menu item allows you to save the *Graphics View* as a Asymptote^[3] picture file.

Limits of these export functions

Apart from Animated GIF export, these exports work only for the 2D View and the following object types can't be exported

- implicit curves
- loci

References

[1] <http://tug.org/PSTricks/main.cgi/>

[2] <http://sourceforge.net/projects/pgf/>

[3] <http://asymptote.sourceforge.net/>

Printing Options

The following only applies to the GeoGebra Desktop Version!

Printing the Graphics View

GeoGebra allows you to print the Graphics View of your constructions. You can find the corresponding item *Print Preview* in the File Menu. In the appearing *Print Preview Dialog* window, you can specify the Title, Author, and a Date for the construction. In addition, you can set the Scale of your printout (in cm) and change the Orientation of the paper used (portrait or landscape).

Note: In order to update the *Print Preview* after you made changes to the text or layout of the printout, you need to press Enter.

Printing the Construction Protocol

If you want to print the Construction Protocol, you first need to open the Construction Protocol dialog window by using the View Menu. Then, select Print... from the File menu of this new window. The Print dialog window allows you to enter the Author or change the paper Margins and Orientation before printing the Construction Protocol.

Note: You may switch the different columns Name, Definition, Command, Algebra, and Breakpoint of the Construction Protocol on and off by using the View Menu of the Construction Protocol dialog window.

Article Sources and Contributors

Introduction *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Manual> *Contributors:* Christina.biermair, Florian.Sonner, GeoGebra Docu Team, Jhohen, Markus, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Sarah., Wkuellinger, Zbynek

Compatibility *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Compatibility> *Contributors:* Markus, Markus.winkler, Murkle, Zbynek

Installation Guide *Source:* https://wiki.geogebra.org/s/en/index.php?title=Reference:GeoGebra_Installation *Contributors:* Alicia.hofstaetter, Davidfruelund, Flefevre, Irkagira, Julia.Wolfinger, Kovács.Zoltán, Markus, Mathmum, Murkle, Noel.Lambert, Spanish1, Wkuellinger, Zbynek, Zoltan

Free, Dependent and Auxiliary Objects *Source:* https://wiki.geogebra.org/s/en/index.php?title=Free%2C_Dependent_and_Auxiliary_Objects *Contributors:* Andrea.duringer, Jhohen, Markus.winkler, Mathmum, Murkle, Zbynek

Geometric Objects *Source:* https://wiki.geogebra.org/s/en/index.php?title=Geometric_Objects *Contributors:* Jhohen, Markus.winkler, Mathmum, Zbynek

Points and Vectors *Source:* https://wiki.geogebra.org/s/en/index.php?title=Points_and_Vectors *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

Lines and Axes *Source:* https://wiki.geogebra.org/s/en/index.php?title=Lines_and_Axes *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

Conic sections *Source:* https://wiki.geogebra.org/s/en/index.php?title=Conic_sections *Contributors:* Markus.winkler, Mathmum, Zbynek

Functions *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Functions> *Contributors:* Markus.winkler, Mathmum, Zbynek

Curves *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Curves> *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Murkle, Zbynek

Inequalities *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Inequalities> *Contributors:* Birgit.Lachner, Markus.winkler, Mathmum, Noel.Lambert, Zbynek

Intervals *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Intervals> *Contributors:* Markus.winkler, Mathmum, Zbynek

General Objects *Source:* https://wiki.geogebra.org/s/en/index.php?title=General_Objects *Contributors:* Markus.winkler, Mathmum, Zbynek

Numbers and Angles *Source:* https://wiki.geogebra.org/s/en/index.php?title=Numbers_and_Angles *Contributors:* Markus.winkler, Mathmum, Zbynek

Texts *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Texts> *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Murkle, Sabrina.azesberger, Zbynek

Boolean values *Source:* https://wiki.geogebra.org/s/en/index.php?title=Boolean_values *Contributors:* Markus.winkler, Mathmum, Murkle, Zbynek

Complex Numbers *Source:* https://wiki.geogebra.org/s/en/index.php?title=Complex_Numbers *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Zbynek

Lists *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Lists> *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

Matrices *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Matrices> *Contributors:* Markus.winkler, Mathmum, Murkle, Sabrina.azesberger, Zbynek

Action Objects *Source:* https://wiki.geogebra.org/s/en/index.php?title=Action_Objects *Contributors:* Markus.winkler, Mathmum, Noel.Lambert, Zbynek

Selecting objects *Source:* https://wiki.geogebra.org/s/en/index.php?title=Selecting_objects *Contributors:* Jhohen, Markus.winkler, Mathmum, Sabrina.azesberger, Zbynek

Change Values *Source:* https://wiki.geogebra.org/s/en/index.php?title=Change_Values *Contributors:* Markus.winkler, Mathmum, Zbynek

Naming Objects *Source:* https://wiki.geogebra.org/s/en/index.php?title=Naming_Objects *Contributors:* Andrea.duringer, Markus, Markus.winkler, Mathmum, Murkle, Sabrina.azesberger, Zbynek

Animation *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Animation> *Contributors:* Florian.Sonner, Markus.winkler, Mathmum, Zbynek

Tracing *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Tracing> *Contributors:* Markus.winkler, Mathmum, Sabrina.azesberger, Zbynek

Object Properties *Source:* https://wiki.geogebra.org/s/en/index.php?title=Object_Properties *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Murkle, Zbynek

Labels and Captions *Source:* https://wiki.geogebra.org/s/en/index.php?title=Labels_and_Captions *Contributors:* Birgit.Lachner, Markus.winkler, Mathmagic, Mathmum, Murkle, Noel.Lambert, Spanish1, Zbynek

Point Capturing *Source:* https://wiki.geogebra.org/s/en/index.php?title=Point_Capturing *Contributors:* Jhohen, Markus.winkler

Advanced Features *Source:* https://wiki.geogebra.org/s/en/index.php?title=Advanced_Features *Contributors:* Markus.winkler, Mathmum, Zbynek

Object Position *Source:* https://wiki.geogebra.org/s/en/index.php?title=Object_Position *Contributors:* Andrea.duringer, Markus.winkler, Zbynek

Conditional Visibility *Source:* https://wiki.geogebra.org/s/en/index.php?title=Conditional_Visibility *Contributors:* Markus.winkler, Mathmum, Zbynek

Dynamic Colors *Source:* https://wiki.geogebra.org/s/en/index.php?title=Dynamic_Colors *Contributors:* Markus.winkler, Mathmum, Zbynek

LaTeX *Source:* <https://wiki.geogebra.org/s/en/index.php?title=LaTeX> *Contributors:* Birgit.Lachner, Markus.winkler, Mathmum, Murkle, Zbynek

Layers *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Layers> *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Zbynek

Scripting *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Scripting> *Contributors:* Birgit.Lachner, Corinna, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

Tooltips *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Tooltips> *Contributors:* Markus.winkler, Sabrina.azesberger, Zbynek

Tools *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Tools> *Contributors:* Jhohen, Markus.winkler, Mathmum, Zbynek

Movement Tools *Source:* https://wiki.geogebra.org/s/en/index.php?title=Movement_Tools *Contributors:* GeoGebra Docu Team, Markus.winkler, Mathmum, Zbynek

Move Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Move_Tool *Contributors:* Florian.Sonner, Jhohen, K.Voss, Markus.winkler, Mathmum, Sabrina.azesberger, Zbynek

Record to Spreadsheet Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Record_to_Spreadsheet_Tool *Contributors:* K.Voss, Markus.winkler, Mathmum, Sarah., Zbynek

Move around Point Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Move_around_Point_Tool *Contributors:* K.Voss, Markus.winkler, Mathmum, Sabrina.azesberger, Zbynek

Point Tools *Source:* https://wiki.geogebra.org/s/en/index.php?title=Point_Tools *Contributors:* Gabriela.Ferenczy, GeoGebra Docu Team, Jhohen, Mathmum, Zbynek

Point Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Point_Tool *Contributors:* K.Voss, Markus.winkler, Mathmum, Zbynek

Attach / Detach Point Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Attach_/_Detach_Point_Tool *Contributors:* Markus.winkler, Zbynek

Complex Number Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Complex_Number_Tool *Contributors:* Markus.winkler, Murkle, Zbynek

Point on Object Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Point_on_Object_Tool *Contributors:* Andrea.duringer, Markus.winkler, Murkle, Zbynek

Intersect Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Intersect_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Midpoint or Center Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Midpoint_or_Center_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Sabrina.azesberger, Zbynek

Line Tools *Source:* https://wiki.geogebra.org/s/en/index.php?title=Line_Tools *Contributors:* Markus.winkler, Mathmum, Sabrina.azesberger, Zbynek

Line Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Line_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Sabrina.azesberger, Spanish1, Zbynek

Segment Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Segment_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Segment with Given Length Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Segment_with_Given_Length_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Ray Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Ray_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Vector from Point Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Vector_from_Point_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Vector Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Vector_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Special Line Tools *Source:* https://wiki.geogebra.org/s/en/index.php?title=Special_Line_Tools *Contributors:* Mathmum, Zbynek

Best Fit Line Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Best_Fit_Line_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Parallel Line Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Parallel_Line_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Angle Bisector Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Angle_Bisector_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Perpendicular Line Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Perpendicular_Line_Tool *Contributors:* Johanna, K Voss, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Tangents Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Tangents_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Murkle, Zbynek

Polar or Diameter Line Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Polar_or_Diameter_Line_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Murkle, Zbynek

Perpendicular Bisector Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Perpendicular_Bisector_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Locus Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Locus_Tool *Contributors:* Andrea.duringer, K Voss, Markus.winkler, Mathmum, Sabrina.azesberger, Zbynek

Polygon Tools *Source:* https://wiki.geogebra.org/s/en/index.php?title=Polygon_Tools *Contributors:* Markus.winkler, Mathmum, Zbynek

Rigid Polygon Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Rigid_Polygon_Tool *Contributors:* Markus.winkler, Mathmum, Zbynek

Vector Polygon Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Vector_Polygon_Tool *Contributors:* Markus.winkler, Mathmum, Murkle, Zbynek

Polyline Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Polyline_Tool *Contributors:* Markus.winkler, Mathmum, Zbynek

Regular Polygon Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Regular_Polygon_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Polygon Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Polygon_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Circle & Arc Tools *Source:* https://wiki.geogebra.org/s/en/index.php?title=Circle_and_Arc_Tools *Contributors:* Mathmum, Zbynek

Circle with Center and Radius Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Circle_with_Center_and_Radius_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Circle through 3 Points Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Circle_through_3_Points_Tool *Contributors:* Andrea.duringer, K Voss, Markus.winkler, Mathmum, Zbynek

Circle with Center through Point Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Circle_with_Center_through_Point_Tool *Contributors:* Administrator, K Voss, Markus.winkler, Mathmum, Zbynek

Circumcircular Arc Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Circumcircular_Arc_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Circumcircular Sector Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Circumcircular_Sector_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Compass Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Compass_Tool *Contributors:* Administrator, K Voss, Markus.winkler, Mathmum, Zbynek

Circular Sector Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Circular_Sector_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Semicircle through 2 Points Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Semicircle_through_2_Points_Tool *Contributors:* Markus.winkler, Mathmum, Zbynek

Circular Arc Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Circular_Arc_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Murkle, Zbynek

Conic Section Tools *Source:* https://wiki.geogebra.org/s/en/index.php?title=Conic_Section_Tools *Contributors:* Christina.biermair, Markus.winkler, Mathmum, Zbynek

Ellipse Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Ellipse_Tool *Contributors:* Administrator, Florian.Sonner, K Voss, Markus.winkler, Mathmum, Zbynek

Hyperbola Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Hyperbola_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Conic through 5 Points Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Conic_through_5_Points_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Parabola Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Parabola_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Measurement Tools *Source:* https://wiki.geogebra.org/s/en/index.php?title=Measurement_Tools *Contributors:* Christina.biermair, Markus.winkler, Mathmum, Sabrina.azesberger, Zbynek

Distance or Length Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Distance_or_Length_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Angle Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Angle_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Slope Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Slope_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Area Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Area_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Angle with Given Size Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Angle_with_Given_Size_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Transformation Tools *Source:* https://wiki.geogebra.org/s/en/index.php?title=Transformation_Tools *Contributors:* Christina.biermair, Mathmum, Zbynek

Translate by Vector Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Translate_by_Vector_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Murkle, Zbynek

Reflect about Line Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Reflect_about_Line_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Reflect about Point Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Reflect_about_Point_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Rotate around Point Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Rotate_around_Point_Tool *Contributors:* GeoGebra Docu Team, Markus.winkler, Mathmum, Zbynek

Reflect about Circle Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Reflect_about_Circle_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Murkle, Zbynek

Dilate from Point Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Dilate_from_Point_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Special Object Tools *Source:* https://wiki.geogebra.org/s/en/index.php?title=Special_Object_Tools *Contributors:* Andrea.duringer, Christina.biermair, Jhohen, Mathmum, Zbynek

Freehand Shape Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Freehand_Shape_Tool *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Murkle, Noel Lambert

Image Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Image_Tool *Contributors:* Jhohen, Markus.winkler, Mathmum, Murkle, Zbynek

Pen Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Pen_Tool *Contributors:* Markus.winkler, Mathmum, Noel Lambert, Zbynek

Slider Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Slider_Tool *Contributors:* Andrea.duringer, Florian.Sonner, K Voss, Markus.winkler, Mathmum, Murkle, Zbynek

Relation Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Relation_Tool *Contributors:* Markus.winkler, Mathmum, Zbynek

Function Inspector Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Function_Inspector_Tool *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Zbynek

Text Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Text_Tool *Contributors:* Markus.winkler, Mathmum, Zbynek

Action Object Tools *Source:* https://wiki.geogebra.org/s/en/index.php?title=Action_Object_Tools *Contributors:* Andrea.duringer, Christina.biermair, Markus.winkler, Mathmum, Zbynek

Check Box Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Check_Box_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Input Box Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Input_Box_Tool *Contributors:* Markus.winkler, Mathmum, Zbynek

Button Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Button_Tool *Contributors:* Markus.winkler, Zbynek

General Tools *Source:* https://wiki.geogebra.org/s/en/index.php?title=General_Tools *Contributors:* Christina.biermair, Markus.winkler, Mathmum, Zbynek

Custom Tools *Source:* https://wiki.geogebra.org/s/en/index.php?title=Custom_Tools *Contributors:* Markus.winkler, Mathmum, Murkle, Zbynek

Show / Hide Label Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title>Show/_Hide_Label_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Zoom Out Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Zoom_Out_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Zoom In Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Zoom_In_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Delete Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Delete_Tool *Contributors:* Andrea.duringer, K Voss, Markus.winkler, Mathmum, Zbynek

Move Graphics View Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Move_Graphics_View_Tool *Contributors:* Jhohen, Markus.winkler, Mathmum, Sabrina.azesberger, Zbynek

Show / Hide Object Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title>Show/_Hide_Object_Tool *Contributors:* K Voss, Markus.winkler, Mathmum, Zbynek

Copy Visual Style Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Copy_Visual_Style_Tool *Contributors:* Markus.winkler, Mathmum, Zbynek

CAS Tools *Source:* https://wiki.geogebra.org/s/en/index.php?title=CAS_Tools *Contributors:* Grafmanuel, Jhohen, Spanish1, Zbynek

Derivative Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Derivative_Tool *Contributors:* Andrea.duringer, Markus.winkler, Zbynek

Evaluate Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Evaluate_Tool *Contributors:* Andrea.duringer, Markus.winkler, Zbynek

Expand Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Expand_Tool *Contributors:* Andrea.duringer, Markus.winkler, Noel Lambert, Zbynek

Factor Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Factor_Tool *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Zbynek

Integral Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Integral_Tool *Contributors:* Andrea.duringer, Markus.winkler, Zbynek

Keep Input Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Keep_Input_Tool *Contributors:* Andrea.duringer, Markus.winkler, Zbynek

Numeric Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Numeric_Tool *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Zbynek

Solve Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Solve_Tool *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Solve Numerically Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Solve_Numerically_Tool *Contributors:* Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

Substitute Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Substitute_Tool *Contributors:* Andrea.duringer, Markus.winkler, Zbynek

Spreadsheet Tools *Source:* https://wiki.geogebra.org/s/en/index.php?title=Spreadsheet_Tools *Contributors:* Andrea.duringer, Grafmanuel, Jhohen, Markus.winkler, Zbynek

One Variable Analysis Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=One_Variable_Analysis_Tool *Contributors:* Gsturr, Markus.winkler, Zbynek

Multiple Variable Analysis Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Multiple_Variable_Analysis_Tool *Contributors:* Gsturr, Mathmum, Sabrina.azesberger, Spanish1, Zbynek

Two Variable Regression Analysis Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Two_Variable_Regression_Analysis_Tool *Contributors:* Gsturr, Markus.winkler, Zbynek

Count Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Count_Tool *Contributors:* Markus.winkler, Zbynek

Maximum Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Maximum_Tool *Contributors:* Markus.winkler, Sabrina.azesberger, Zbynek

Minimum Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Minimum_Tool *Contributors:* Markus.winkler, Zbynek

Mean Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Mean_Tool *Contributors:* Markus.winkler, Sabrina.azesberger, Zbynek

List Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=List_Tool *Contributors:* Markus.winkler

List of Points Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=List_of_Points_Tool *Contributors:* Gsturr, Markus.winkler, Zbynek

Matrix Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Matrix_Tool *Contributors:* Andrea.duringer, Markus.winkler, Zbynek

Sum Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Sum_Tool *Contributors:* Andrea.duringer, Markus.winkler, Spanish1, Zbynek

Table Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Table_Tool *Contributors:* Andrea.duringer, Markus.winkler, Zbynek

3D Graphics Tools *Source:* https://wiki.geogebra.org/s/en/index.php?title=3D_Graphics_Tools *Contributors:* Grafmanuel, Jhohen

Rotate 3D Graphics View Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Rotate_3D_Graphics_View_Tool *Contributors:* Johanna, Markus.winkler, Mathmum, Noel Lambert, Sarah., Zbynek

Rotate around Line Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Rotate_around_Line_Tool *Contributors:* Markus.winkler, Noel Lambert, Sarah., Zbynek

Plane Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Plane_Tool *Contributors:* Johanna, Markus.winkler, Noel Lambert, Zbynek

Plane through 3 Points Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Plane_through_3_Points_Tool *Contributors:* Gabriela Ferenczy, Johanna, Markus.winkler, Noel Lambert, Zbynek

View in front of Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=View_in_front_of_Tool *Contributors:* Johanna, Markus.winkler, Mathmum, Noel Lambert, Sarah., Zbynek

Cone Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Cone_Tool *Contributors:* Johanna, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Circle with Axis through Point Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Circle_with_Axis_through_Point_Tool *Contributors:* Johanna, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Circle with Center, Radius and Direction Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Circle_with_Center%2C_Radius_and_Direction_Tool *Contributors:* Johanna, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Sphere with Center through Point Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Sphere_with_Center_through_Point_Tool *Contributors:* Johanna, Markus.winkler, Noel Lambert, Zbynek

Sphere with Center and Radius Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Sphere_with_Center_and_Radius_Tool *Contributors:* Johanna, Markus.winkler, Noel Lambert, Zbynek

Volume Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Volume_Tool *Contributors:* Johanna, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Cube Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Cube_Tool *Contributors:* Johanna, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Extrude to Prism or Cylinder Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Extrude_to_Prism_or_Cylinder_Tool *Contributors:* Johanna, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Extrude to Pyramid or Cone Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Extrude_to_Pyramid_or_Cone_Tool *Contributors:* Johanna, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Cylinder Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Cylinder_Tool *Contributors:* Johanna, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Intersect Two Surfaces Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Intersect_Two_Surfaces_Tool *Contributors:* Johanna, Markus.winkler, Noel Lambert, Zbynek

Reflect about Plane Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Reflect_about_Plane_Tool *Contributors:* Johanna, Markus.winkler, Noel Lambert, Zbynek

Net Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Net_Tool *Contributors:* Johanna, Markus.winkler, Mathmum, Noel Lambert, Sabrina.azesberger, Zbynek

Perpendicular Plane Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Perpendicular_Plane_Tool *Contributors:* Johanna, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Parallel Plane Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Parallel_Plane_Tool *Contributors:* Johanna, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Prism Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Prism_Tool *Contributors:* Johanna, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Pyramid Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Pyramid_Tool *Contributors:* Johanna, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Regular Tetrahedron Tool *Source:* https://wiki.geogebra.org/s/en/index.php?title=Regular_Tetrahedron_Tool *Contributors:* Johanna, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Commands *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Commands> *Contributors:* Florian.Sonner, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Geometry Commands *Source:* https://wiki.geogebra.org/s/en/index.php?title=Geometry_Commands *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Zbynek

AffineRatio Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=AffineRatio_Command *Contributors:* JohannaZ, K.Voss, Markus.winkler, Mathmum, Spanish1, Zbynek

Angle Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Angle_Command *Contributors:* Jackhu, JohannaZ, K.Voss, Markus.winkler, Mathmum, Murkle, Noel Lambert, Solyom-Gecse.Csilla, Spanish1, Zbynek

AngleBisector Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=AngleBisector_Command *Contributors:* Johanna, JohannaZ, K.Voss, Markus.winkler, Mathmum, Spanish1, Zbynek

Arc Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Arc_Command *Contributors:* K.Voss, Magdalena.SophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Area Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Area_Command *Contributors:* Administrator, Johanna, JohannaZ, K.Voss, Markus.winkler, Mathieu, Mathmum, Murkle, Zbynek

AreEqual Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=AreEqual_Command *Contributors:* Markus.winkler, Mathmum, Murkle, Sarah., Spanish1, Zbynek, Zoltan, Zoltan.Kovacs

AreCollinear Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=AreCollinear_Command *Contributors:* Markus.winkler, Mathmum, Murkle, Sarah., Spanish1, Zbynek, Zoltan.Kovacs

AreConcyclic Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=AreConcyclic_Command *Contributors:* Markus.winkler, Mathmum, Murkle, Sarah., Spanish1, Zbynek, Zoltan.Kovacs

AreCongruent Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=AreCongruent_Command *Contributors:* Markus.winkler, Mathmum, Murkle, Zbynek, Zoltan

AreConcurrent Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=AreConcurrent_Command *Contributors:* Markus.winkler, Mathmum, Murkle, Noel Lambert, Sarah., Spanish1, Zbynek, Zoltan, Zoltan.Kovacs

ArePerpendicular Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ArePerpendicular_Command *Contributors:* Markus.winkler, Mathmum, Murkle, Sarah., Spanish1, Zbynek, Zoltan, Zoltan.Kovacs

AreParallel Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=AreParallel_Command *Contributors:* Markus.winkler, Mathmum, Murkle, Sarah., Spanish1, Zbynek, Zoltan.Kovacs

Barycenter Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Barycenter_Command *Contributors:* Andrea.duringer, Magdalena.SophieF, Markus.winkler, Mathmum, Spanish1, Zbynek

Centroid Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Centroid_Command *Contributors:* K.Voss, Magdalena.SophieF, Markus.winkler, Mathmum, Spanish1, Zbynek

CircularArc Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=CircularArc_Command *Contributors:* Andrea.duringer, JohannaZ, K.Voss, Markus.winkler, Mathmum, Spanish1, Zbynek

CircularSector Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=CircularSector_Command *Contributors:* Andrea.duringer, JohannaZ, K.Voss, Markus.winkler, Mathmum, Spanish1, Zbynek

CircumcircularArc Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=CircumcircularArc_Command *Contributors:* Andrea.duringer, JohannaZ, K.Voss, Markus.winkler, Mathmum, Spanish1, Zbynek

CircumcircularSector Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=CircumcircularSector_Command *Contributors:* Andrea.duringer, JohannaZ, K Voss, Markus.winkler, Mathmum, Zbynek

Circumference Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Circumference_Command *Contributors:* Andrea.duringer, K Voss, Markus.winkler, Mathmum, Murkle, Zbynek

ClosestPoint Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ClosestPoint_Command *Contributors:* Andrea.duringer, Johanna, JohannaZ, MagdalenaSophieF, Markus.winkler, Murkle, Zbynek

CrossRatio Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=CrossRatio_Command *Contributors:* JohannaZ, K Voss, Markus.winkler, Mathmum, Spanish1, Zbynek

Cubic Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Cubic_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Murkle, Zbynek, Zoltan Kovacs

Direction Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Direction_Command *Contributors:* K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Zbynek

Distance Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Distance_Command *Contributors:* Andrea.duringer, Johanna, JohannaZ, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Sarah., Zbynek

Envelope Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Envelope_Command *Contributors:* Markus.winkler, Murkle, Sarah., Zbynek, Zoltan, Zoltan Kovacs

Intersect Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Intersect_Command *Contributors:* Andrea.duringer, Johanna, JohannaZ, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, Sabsi, Sarah., SarahSt, Zbynek

IntersectPath Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=IntersectPath_Command *Contributors:* Markus.winkler, Mathmum, Sabsi, Sarah., Zbynek

Length Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Length_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, Johanna, JohannaZ, K Voss, Markus.winkler, Mathieu, Mathmum, Murkle, Sabrina.azesberger, UnTom, Zbynek

Line Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Line_Command *Contributors:* K Voss, Kimeswenger, Markus.winkler, Mathmum, Murkle, Sabrina.azesberger, Spanish1, Zbynek

PerpendicularBisector Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=PerpendicularBisector_Command *Contributors:* Andrea.duringer, Gabriela Ferenczy, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Sabsi, Sarah., Zbynek

Locus Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Locus_Command *Contributors:* Andrea.duringer, JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Sabrina.azesberger, Zbynek

LocusEquation Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=LocusEquation_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Murkle, Spanish1, Zbynek, Zoltan, Zoltan Kovacs

Midpoint Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Midpoint_Command *Contributors:* Johanna, JohannaZ, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, Spanish1, Zbynek

PerpendicularLine Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=PerpendicularLine_Command *Contributors:* Andrea.duringer, Johanna, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Sarah., Zbynek

Perimeter Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Perimeter_Command *Contributors:* JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Zbynek

Point Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Point_Command *Contributors:* JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

PointIn Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=PointIn_Command *Contributors:* Andrea.duringer, Markus.winkler, Zbynek

Polyline Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Polyline_Command *Contributors:* JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Polygon Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Polygon_Command *Contributors:* Andrea.duringer, JohannaZ, K Voss, Markus.winkler, Mathmum, Noel Lambert, Sarah., Zbynek

Prove Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Prove_Command *Contributors:* Markus, Markus.winkler, Mathmum, Murkle, Sarah., Spanish1, Zbynek, Zoltan, Zoltan Kovacs

ProveDetails Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ProveDetails_Command *Contributors:* Markus.winkler, Mathmum, Murkle, Sarah., Spanish1, Zbynek, Zoltan, Zoltan Kovacs

Radius Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Radius_Command *Contributors:* K Voss, Kimeswenger, MagdalenaSophieF, Markus.winkler, Mathmum, Zbynek

Ray Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Ray_Command *Contributors:* Andrea.duringer, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Spanish1, Zbynek

RigidPolygon Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=RigidPolygon_Command *Contributors:* Andrea.duringer, MagdalenaSophieF, Markus.winkler, Noel Lambert, Sarah., Zbynek

Sector Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Sector_Command *Contributors:* K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Spanish1, Zbynek

Segment Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Segment_Command *Contributors:* Andrea.duringer, Johanna, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, Spanish1, Zbynek

Slope Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Slope_Command *Contributors:* Andrea.duringer, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Zbynek

Tangent Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Tangent_Command *Contributors:* Andrea.duringer, Johanna, JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Noel Lambert, Sarah., Zbynek, Zoltan

Vertex Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Vertex_Command *Contributors:* Birgit Lachner, Johanna, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, Spanish1, Zbynek

TriangleCurve Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TriangleCurve_Command *Contributors:* JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek

TriangleCenter Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TriangleCenter_Command *Contributors:* JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Trilinear Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Trilinear_Command *Contributors:* Gabriela Ferenczy, MagdalenaSophieF, Markus.winkler, Murkle, Noel Lambert, Zbynek

Algebra Commands *Source:* https://wiki.geogebra.org/s/en/index.php?title=Algebra_Commands *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Zbynek

Div Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Div_Command *Contributors:* Alexander Hartl, Christina.biermair, K Voss, Markus.winkler, Mathmum, UnTom, Zbynek

Expand Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Expand_Command *Contributors:* Alexander.Hartl, Andrea.duringer, JohannaZ, K.Voss, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

Factor Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Factor_Command *Contributors:* Alexander.Hartl, Christina.biermair, JohannaZ, K.Voss, Markus.winkler, Mathmum, Murkle, Zbynek

FromBase Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=FromBase_Command *Contributors:* Andrea.duringer, JohannaZ, Markus.winkler, Mathmum, Zbynek

IFactor Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=IFactor_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Noel.Lambert, Zbynek

GCD Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=GCD_Command *Contributors:* Alexander.Hartl, Christina.biermair, K.Voss, Kimeswenger, Markus.winkler, Mathmum, Noel.Lambert, Zbynek

LCM Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=LCM_Command *Contributors:* Alexander.Hartl, Andrea.duringer, JohannaZ, K.Voss, Markus.winkler, Mathmum, Noel.Lambert, Zbynek

Max Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Max_Command *Contributors:* Alexander.Hartl, Andrea.duringer, Christina.biermair, K.Voss, Magdalena.SophieF, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Sarah., Spanish1, Zbynek

Min Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Min_Command *Contributors:* Alexander.Hartl, Christina.biermair, JohannaZ, K.Voss, Magdalena.SophieF, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Sarah., Zbynek

Mod Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Mod_Command *Contributors:* Alexander.Hartl, Andrea.duringer, Christina.biermair, JohannaZ, K.Voss, Markus.winkler, Mathmum, Murkle, Sabrina.azesberger, Zbynek

PrimeFactors Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=PrimeFactors_Command *Contributors:* Alexander.Hartl, Andrea.duringer, Christina.biermair, JohannaZ, Markus.winkler, Mathmum, Zbynek

Simplify Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Simplify_Command *Contributors:* Alexander.Hartl, Andrea.duringer, Christina.biermair, Gabriela.Ferenczy, JohannaZ, K.Voss, Markus.winkler, Mathmum, Murkle, Spanish1, Zbynek

ToBase Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ToBase_Command *Contributors:* Andrea.duringer, JohannaZ, Magdalena.SophieF, Markus.winkler, Mathmum, Zbynek

Text Commands *Source:* https://wiki.geogebra.org/s/en/index.php?title=Text_Commands *Contributors:* GeoGebra Docu Team, Mathmum, Zbynek

ContinuedFraction Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ContinuedFraction_Command *Contributors:* Andrea.duringer, JohannaZ, Markus.winkler, Mathmum, Spanish1, Zbynek

FractionText Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=FractionText_Command *Contributors:* Andrea.duringer, Fvitabar, K.Voss, Magdalena.SophieF, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

FormulaText Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=FormulaText_Command *Contributors:* JohannaZ, K.Voss, Markus.winkler, Mathmum, Zbynek

LetterToUnicode Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=LetterToUnicode_Command *Contributors:* JohannaZ, K.Voss, Magdalena.SophieF, Markus.winkler, Mathmum, Zbynek

Ordinal Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Ordinal_Command *Contributors:* Markus.winkler, Mathmum, Murkle, Zbynek

RotateText Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=RotateText_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Zbynek

ScientificText Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ScientificText_Command *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Murkle, Zbynek

SurdText Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SurdText_Command *Contributors:* Andrea.duringer, JohannaZ, Magdalena.SophieF, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

TableText Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TableText_Command *Contributors:* JohannaZ, K.Voss, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Spanish1, Zbynek

Text Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Text_Command *Contributors:* Fvitabar, JohannaZ, K.Voss, Markus.winkler, Mathmum, Zbynek

TextToUnicode Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TextToUnicode_Command *Contributors:* JohannaZ, K.Voss, Markus.winkler, Mathmum, Zbynek

UnicodeToLetter Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=UnicodeToLetter_Command *Contributors:* JohannaZ, K.Voss, Markus.winkler, Mathmum, Zbynek

UnicodeToText Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=UnicodeToText_Command *Contributors:* JohannaZ, K.Voss, Markus.winkler, Mathmum, Noel.Lambert, Zbynek

VerticalText Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=VerticalText_Command *Contributors:* Andrea.duringer, Gsturr, Markus.winkler, Mathmum, Murkle, Spanish1, Zbynek

Logic Commands *Source:* https://wiki.geogebra.org/s/en/index.php?title=Logic_Commands *Contributors:* Markus.winkler, Mathmum, Zbynek

CountIf Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=CountIf_Command *Contributors:* JohannaZ, K.Voss, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

IsDefined Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=IsDefined_Command *Contributors:* K.Voss, Magdalena.SophieF, Markus.winkler, Mathmum, Zbynek

If Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=If_Command *Contributors:* Christina.biermair, JohannaZ, K.Voss, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Spanish1, Zbynek

IsInRegion Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=IsInRegion_Command *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Zbynek

IsInteger Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=IsInteger_Command *Contributors:* K.Voss, Magdalena.SophieF, Markus.winkler, Mathmum, Zbynek

KeepIf Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=KeepIf_Command *Contributors:* Andrea.duringer, JohannaZ, K.Voss, Magdalena.SophieF, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

Relation Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Relation_Command *Contributors:* K.Voss, Magdalena.SophieF, Markus.winkler, Mathmum, Zbynek, Zoltan

Financial Commands *Source:* https://wiki.geogebra.org/s/en/index.php?title=Financial_Commands *Contributors:* Markus.winkler, Zbynek

FutureValue Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=FutureValue_Command *Contributors:* Markus.winkler, Mathmum, Noel.Lambert, Sarah., Zbynek

Payment Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Payment_Command *Contributors:* Markus.winkler, Mathmum, Noel.Lambert, Sarah., Zbynek

Periods Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Periods_Command *Contributors:* Markus.winkler, Mathmum, Noel.Lambert, Sarah., Zbynek

PresentValue Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=PresentValue_Command *Contributors:* Markus.winkler, Mathmum, Noel.Lambert, Sabsi, Sarah., Zbynek

Rate Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Rate_Command *Contributors:* Markus.winkler, Mathmum, Noel Lambert, Sabsi, Sarah., Zbynek

Functions & Calculus Commands *Source:* https://wiki.geogebra.org/s/en/index.php?title=Functions_and_Calculus_Commands *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Asymptote Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Asymptote_Command *Contributors:* JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Zbynek

Coefficients Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Coefficients_Command *Contributors:* Alexander Hartl, Birgit Lachner, JohannaZ, Markus.winkler, Mathmum, Murkle, Noel Lambert, Spanish1, Zbynek

CompleteSquare Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=CompleteSquare_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Zbynek

ComplexRoot Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ComplexRoot_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, JohannaZ, Markus.winkler, Mathmum, Zbynek

Curvature Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Curvature_Command *Contributors:* Johanna, JohannaZ, K Voss, Markus.winkler, Mathmum, Zbynek

CurvatureVector Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=CurvatureVector_Command *Contributors:* Johanna, JohannaZ, K Voss, Markus.winkler, Mathmum, Zbynek

Curve Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Curve_Command *Contributors:* Johanna, JohannaZ, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

Degree Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Degree_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, JohannaZ, Markus.winkler, Mathmum, Spanish1, Zbynek

Denominator Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Denominator_Command *Contributors:* Alexander Hartl, Christina.biermair, JohannaZ, Markus.winkler, Mathmum, Murkle, Zbynek

Derivative Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Derivative_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, JohannaZ, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, UnTom, Zbynek

Extremum Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Extremum_Command *Contributors:* JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Spanish1, Zbynek

Factors Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Factors_Command *Contributors:* Alexander Hartl, MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Function Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Function_Command *Contributors:* Andrea.duringer, Florian Sonner, JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

ImplicitCurve Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ImplicitCurve_Command *Contributors:* JohannaZ, Markus.winkler, Murkle, Noel Lambert, Zbynek

Integral Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Integral_Command *Contributors:* Alexander Hartl, Christina.biermair, K Voss, Kimeswenger, Markus.winkler, Mathmum, Murkle, Spanish1, Zbynek

IntegralBetween Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=IntegralBetween_Command *Contributors:* Alexander Hartl, Christina.biermair, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Zbynek

Iteration Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Iteration_Command *Contributors:* JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Noel Lambert, Spanish1, Zbynek

IterationList Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=IterationList_Command *Contributors:* JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

LeftSum Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=LeftSum_Command *Contributors:* Andrea.duringer, JohannaZ, Kimeswenger, Markus.winkler, Mathmum, Spanish1, Zbynek, Zoltan

Limit Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Limit_Command *Contributors:* Alexander Hartl, Christina.biermair, JohannaZ, Kimeswenger, Markus.winkler, Mathmum, Murkle, Noel Lambert, Sabrina.azesberger, Zbynek

LimitAbove Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=LimitAbove_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, JohannaZ, Markus.winkler, Mathmum, Murkle, Zbynek

LimitBelow Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=LimitBelow_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, JohannaZ, Markus.winkler, Mathmum, Murkle, Spanish1, Zbynek

LowerSum Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=LowerSum_Command *Contributors:* JohannaZ, K Voss, Markus.winkler, Mathmum, Sabrina.azesberger, Spanish1, Zbynek, Zoltan

NSolveODE Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=NSolveODE_Command *Contributors:* Balazs.bencze, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Numerator Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Numerator_Command *Contributors:* Alexander Hartl, Christina.biermair, JohannaZ, Markus.winkler, Mathmum, Murkle, Zbynek

OsculatingCircle Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=OsculatingCircle_Command *Contributors:* Johanna, JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Zbynek

ParametricDerivative Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ParametricDerivative_Command *Contributors:* Andrea.duringer, JohannaZ, Markus.winkler, Mathmum, Murkle, Zbynek

PartialFractions Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=PartialFractions_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, JohannaZ, Markus.winkler, Mathmum, Zbynek

PathParameter Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=PathParameter_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Murkle, Zbynek

Polynomial Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Polynomial_Command *Contributors:* Johanna, JohannaZ, K Voss, Markus.winkler, Mathmum, Noel Lambert, Zbynek

RectangleSum Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=RectangleSum_Command *Contributors:* Markus.winkler, Mathmum, Spanish1, Zbynek, Zoltan

Root Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Root_Command *Contributors:* Alexander Hartl, Christina.biermair, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Zbynek

RootList Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=RootList_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Roots Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Roots_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Mathmum, Spanish1, Zbynek

SlopeField Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SlopeField_Command *Contributors:* Andrea.duringer, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Zbynek

SolveODE Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SolveODE_Command *Contributors:* Alexander.Hartl, Christina.biermair, JohannaZ, MagdalenaSophieF, Markus, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

Spline Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Spline_Command *Contributors:* Johanna, Markus.winkler, Mathmum, Zbynek

TaylorPolynomial Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TaylorPolynomial_Command *Contributors:* Alexander.Hartl, Christina.biermair, JohannaZ, K.Voss, Markus.winkler, Mathmum, Zbynek

TrapezoidalSum Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TrapezoidalSum_Command *Contributors:* JohannaZ, K.Voss, Markus.winkler, Mathmum, Spanish1, Zbynek, Zoltan

TrigExpand Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TrigExpand_Command *Contributors:* Andrea.duringer, Christina.biermair, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

TrigCombine Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TrigCombine_Command *Contributors:* Andrea.duringer, Christina.biermair, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Spanish1, Zbynek

TrigSimplify Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TrigSimplify_Command *Contributors:* Andrea.duringer, Christina.biermair, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Noel.Lambert, Spanish1, Zbynek

InflectionPoint Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=InflectionPoint_Command *Contributors:* JohannaZ, K.Voss, Markus.winkler, Mathmum, Murkle, Zbynek

UpperSum Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=UpperSum_Command *Contributors:* JohannaZ, K.Voss, Markus.winkler, Mathmum, Spanish1, Zbynek, Zoltan

Conic Commands *Source:* https://wiki.geogebra.org/s/en/index.php?title=Conic_Commands *Contributors:* Markus.winkler, Mathmum, Spanish1, Zbynek

Axes Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Axes_Command *Contributors:* Johanna, K.Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Sarah., Zbynek

Center Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Center_Command *Contributors:* Johanna, K.Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Sarah., Zbynek

Circle Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Circle_Command *Contributors:* Administrator, Johanna, JohannaZ, K.Voss, Markus.winkler, Mathmum, Murkle, Spanish1, Zbynek

Conic Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Conic_Command *Contributors:* Christina.biermair, JohannaZ, K.Voss, Markus.winkler, Mathmum, Murkle, Zbynek

ConjugateDiameter Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ConjugateDiameter_Command *Contributors:* K.Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Spanish1, Zbynek

Directrix Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Directrix_Command *Contributors:* JohannaZ, K.Voss, Markus.winkler, Mathmum, Murkle, Zbynek

Eccentricity Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Eccentricity_Command *Contributors:* Markus.winkler, Mathmum, Zbynek

Ellipse Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Ellipse_Command *Contributors:* Administrator, JohannaZ, K.Voss, Markus.winkler, Mathmum, Noel.Lambert, Zbynek

LinearEccentricity Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=LinearEccentricity_Command *Contributors:* K.Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Sabrina.azesberger, Zbynek

MajorAxis Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=MajorAxis_Command *Contributors:* JohannaZ, K.Voss, Markus.winkler, Mathmum, Sabrina.azesberger, Zbynek

SemiMajorAxisLength Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SemiMajorAxisLength_Command *Contributors:* JohannaZ, K.Voss, Markus.winkler, Mathmum, Zbynek

Focus Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Focus_Command *Contributors:* K.Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Zbynek

Hyperbola Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Hyperbola_Command *Contributors:* JohannaZ, K.Voss, Markus.winkler, Mathmum, Zbynek

Incircle Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Incircle_Command *Contributors:* Markus.winkler, Mathmum, Murkle, Zbynek

Parabola Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Parabola_Command *Contributors:* Andrea.duringer, JohannaZ, K.Voss, Markus.winkler, Mathmum, Zbynek

Parameter Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Parameter_Command *Contributors:* JohannaZ, K.Voss, Markus.winkler, Mathmum, Zbynek

Polar Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Polar_Command *Contributors:* Andrea.duringer, JohannaZ, K.Voss, Markus.winkler, Mathmum, Noel.Lambert, Zbynek

MinorAxis Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=MinorAxis_Command *Contributors:* JohannaZ, K.Voss, Markus.winkler, Mathmum, Sabrina.azesberger, Zbynek

SemiMinorAxisLength Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SemiMinorAxisLength_Command *Contributors:* JohannaZ, K.Voss, Markus.winkler, Mathmum, Zbynek

Semicircle Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Semicircle_Command *Contributors:* Andrea.duringer, JohannaZ, K.Voss, Markus.winkler, Mathmum, Zbynek

List Commands *Source:* https://wiki.geogebra.org/s/en/index.php?title=List_Commands *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Zbynek

Append Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Append_Command *Contributors:* K.Voss, Markus.winkler, Mathmum, Noel.Lambert, Spanish1, Zbynek

Element Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Element_Command *Contributors:* Alexander.Hartl, Andrea.duringer, Christina.biermair, JohannaZ, K.Voss, Markus.winkler, Mathmum, Murkle, Noel.Lambert, UnTom, Zbynek

Flatten Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Flatten_Command *Contributors:* Andrea.duringer, MagdalenaSophieF, Markus.winkler, Mathmum, Spanish1, Zbynek

First Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=First_Command *Contributors:* Alexander.Hartl, Andrea.duringer, Christina.biermair, JohannaZ, K.Voss, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

IndexOf Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=IndexOf_Command *Contributors:* Gabriela.Ferenczy, JohannaZ, Markus.winkler, Mathmum, Noel.Lambert, Zbynek

Insert Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Insert_Command *Contributors:* JohannaZ, K.Voss, Markus.winkler, Mathmum, Noel.Lambert, Spanish1, Zbynek

Intersection Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Intersection_Command *Contributors:* Gabriela.Ferenczy, K.Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

Join Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Join_Command *Contributors:* JohannaZ, K.Voss, Markus.winkler, Mathmum, Noel.Lambert, Spanish1, Zbynek

Last Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Last_Command *Contributors:* Alexander.Hartl, Andrea.duringer, Christina.biermair, JohannaZ, K.Voss, Markus.winkler, Mathmum, Noel.Lambert, Zbynek

- OrdinalRank Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=OrdinalRank_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek
- PointList Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=PointList_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Noel Lambert, Zbynek
- Product Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Product_Command *Contributors:* Alexander Hartl, Andrea.duringer, JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Noel Lambert, Spanish1, UnTom, Zbynek
- RandomElement Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=RandomElement_Command *Contributors:* Christina.biermair, Markus.winkler, Mathmum, Murkle, Noel Lambert, UnTom, Zbynek
- Remove Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Remove_Command *Contributors:* Markus.winkler, Mathieu, Mathmum, Murkle, Noel Lambert, Zbynek
- RemoveUndefined Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=RemoveUndefined_Command *Contributors:* JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Noel Lambert, Spanish1, Zbynek
- Reverse Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Reverse_Command *Contributors:* Fvitabar, JohannaZ, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Spanish1, Zbynek
- SelectedElement Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=SelectedElement_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek
- SelectedIndex Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=SelectedIndex_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek
- Sequence Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Sequence_Command *Contributors:* Alexander Hartl, JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, UnTom, Zbynek
- Sort Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Sort_Command *Contributors:* Andrea.duringer, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek
- Take Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Take_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, JohannaZ, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek
- TiedRank Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=TiedRank_Command *Contributors:* Andrea.duringer, JohannaZ, Markus.winkler, Mathmum, Murkle, Noel Lambert, Spanish1, Zbynek
- Union Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Union_Command *Contributors:* JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Noel Lambert, Spanish1, Zbynek
- Unique Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Unique_Command *Contributors:* Alexander Hartl, Andrea.duringer, Gsturr, JohannaZ, Markus.winkler, Mathmum, Noel Lambert, UnTom, Zbynek
- Zip Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Zip_Command *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Noel Lambert, Sabrina.azesberger, Spanish1, Zbynek
- Vector & Matrix Commands** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Vector_and_Matrix_Commands *Contributors:* Markus.winkler, Mathmum, Zbynek
- ApplyMatrix Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=ApplyMatrix_Command *Contributors:* Fvitabar, MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek
- Determinant Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Determinant_Command *Contributors:* Alexander Hartl, Christina.biermair, K Voss, Markus.winkler, Mathmum, Zbynek
- Identity Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Identity_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, JohannaZ, Markus.winkler, Mathmum, Murkle, Zbynek
- Invert Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Invert_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Spanish1, UnTom, Zbynek
- PerpendicularVector Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=PerpendicularVector_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Sarah., Zbynek
- ReducedRowEchelonForm Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=ReducedRowEchelonForm_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, JohannaZ, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek
- Transpose Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Transpose_Command *Contributors:* Alexander Hartl, Christina.biermair, K Voss, Markus.winkler, Mathmum, Murkle, Zbynek
- UnitPerpendicularVector Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=UnitPerpendicularVector_Command *Contributors:* Alexander Hartl, Andrea.duringer, JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Sarah., Zbynek
- UnitVector Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=UnitVector_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, UnTom, Zbynek
- Vector Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Vector_Command *Contributors:* Andrea.duringer, JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Zbynek
- Transformation Commands** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Transformation_Commands *Contributors:* Andrea.duringer, Mathmum, Zbynek
- AttachCopyToView Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=AttachCopyToView_Command *Contributors:* JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Spanish1, Zbynek
- Dilate Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Dilate_Command *Contributors:* Andrea.duringer, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Zbynek
- Reflect Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Reflect_Command *Contributors:* Fvitabar, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Sarah., Zbynek
- Rotate Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Rotate_Command *Contributors:* Andrea.duringer, Fvitabar, Johanna, JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek
- Shear Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Shear_Command *Contributors:* JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Zbynek
- Stretch Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Stretch_Command *Contributors:* Fvitabar, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Zbynek
- Translate Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Translate_Command *Contributors:* Andrea.duringer, JohannaZ, K Voss, Markus.winkler, Mathmum, Zbynek
- Chart Commands** *Source:* https://wiki.geogebra.org/s/en/index.php?title=Chart_Commands *Contributors:* Christina.biermair, Markus.winkler, Mathmum, Zbynek
- BarChart Command** *Source:* https://wiki.geogebra.org/s/en/index.php?title=BarChart_Command *Contributors:* Gsturr, JohannaZ, K Voss, Kimeswenger, Markus.winkler, Mathmum, Noel Lambert, Sarah., Zbynek

BoxPlot Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=BoxPlot_Command *Contributors:* GeoGebra Docu Team, K Voss, Kimeswenger, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

ContingencyTable Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ContingencyTable_Command *Contributors:* Gsturr, Markus.winkler, Mathmum, Murkle

DotPlot Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=DotPlot_Command *Contributors:* Johanna, JohannaZ, Markus.winkler, Mathmum, Noel Lambert, Sabsi, Sarah., Zbynek

FrequencyPolygon Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=FrequencyPolygon_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Histogram Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Histogram_Command *Contributors:* Gsturr, JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

HistogramRight Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=HistogramRight_Command *Contributors:* Christina.biermair, Kimeswenger, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

NormalQuantilePlot Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=NormalQuantilePlot_Command *Contributors:* Gsturr, MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek

ResidualPlot Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ResidualPlot_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek

StemPlot Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=StemPlot_Command *Contributors:* Gsturr, MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek

StepGraph Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=StepGraph_Command *Contributors:* Gsturr, JohannaZ, Markus.winkler, Mathmum, Murkle, Noel Lambert

StickGraph Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=StickGraph_Command *Contributors:* Gsturr, JohannaZ, Markus.winkler, Mathmum, Murkle

Statistics Commands *Source:* https://wiki.geogebra.org/s/en/index.php?title=Statistics_Commands *Contributors:* Andrea.duringer, Jhohen, Markus.winkler, Mathmum, Zbynek

ANOVA Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ANOVA_Command *Contributors:* Gsturr, MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Classes Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Classes_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

CorrelationCoefficient Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=CorrelationCoefficient_Command *Contributors:* JohannaZ, K Voss, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Covariance Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Covariance_Command *Contributors:* Alexander Hartl, Christina.biermair, JohannaZ, K Voss, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Fit Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Fit_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

FitExp Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=FitExp_Command *Contributors:* Alexander Hartl, Christina.biermair, JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

FitGrowth Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=FitGrowth_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

FitImplicit Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=FitImplicit_Command *Contributors:* Sarah., Zbynek

FitLineX Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=FitLineX_Command *Contributors:* Andrea.duringer, JohannaZ, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

FitLine Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=FitLine_Command *Contributors:* Andrea.duringer, JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

FitLog Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=FitLog_Command *Contributors:* Alexander Hartl, Christina.biermair, JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

FitLogistic Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=FitLogistic_Command *Contributors:* JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Noel Lambert, Spanish1, Zbynek

FitPoly Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=FitPoly_Command *Contributors:* Alexander Hartl, Christina.biermair, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

FitPow Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=FitPow_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

FitSin Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=FitSin_Command *Contributors:* Alexander Hartl, Christina.biermair, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

Frequency Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Frequency_Command *Contributors:* Gsturr, JohannaZ, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

FrequencyTable Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=FrequencyTable_Command *Contributors:* Andrea.duringer, JohannaZ, Markus.winkler, Mathmum, Murkle, Noel Lambert, Sabsi, Zbynek

GeometricMean Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=GeometricMean_Command *Contributors:* Markus.winkler, Mathmum, Zbynek

HarmonicMean Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=HarmonicMean_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Mean Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Mean_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, Gsturr, JohannaZ, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

MeanX Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=MeanX_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Noel Lambert, Zbynek

MeanY Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=MeanY_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Median Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Median_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, JohannaZ, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, Spanish1, Zbynek

Mode Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Mode_Command *Contributors:* K Voss, Markus.winkler, Mathmum, Noel Lambert, Spanish1, Zbynek

Normalize Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Normalize_Command *Contributors:* Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

Percentile Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Percentile_Command *Contributors:* Andrea.duringer, Gsturr, JohannaZ, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Q1 Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Q1_Command *Contributors:* JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

Q3 Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Q3_Command *Contributors:* JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

RSquare Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=RSquare_Command *Contributors:* Andrea.duringer, Gsturr, JohannaZ, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

RootMeanSquare Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=RootMeanSquare_Command *Contributors:* JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek

SD Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SD_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, Spanish1, Zbynek

SDX Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SDX_Command *Contributors:* Gabriela Ferenczy, MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek

SDY Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SDY_Command *Contributors:* Gabriela Ferenczy, MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Sxx Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Sxx_Command *Contributors:* K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Sxy Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Sxy_Command *Contributors:* K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

Syy Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Syy_Command *Contributors:* K Voss, MagdalenaSophieF, Markus.winkler, Murkle, Noel Lambert, Zbynek

Sample Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Sample_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek

SampleSD Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SampleSD_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

SampleSDX Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SampleSDX_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek

SampleSDY Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SampleSDY_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek

SampleVariance Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SampleVariance_Command *Contributors:* Alexander Hartl, Christina.biermair, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, UnTom, Zbynek

Shuffle Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Shuffle_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, Spanish1, Zbynek

SigmaXX Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SigmaXX_Command *Contributors:* K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

SigmaXY Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SigmaXY_Command *Contributors:* K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek

SigmaYY Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SigmaYY_Command *Contributors:* K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Spearman Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Spearman_Command *Contributors:* Gsturr, MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Sum Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Sum_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, JohannaZ, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

SumSquaredErrors Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SumSquaredErrors_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Mathmum, Noel Lambert, Zbynek

TMean2Estimate Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TMean2Estimate_Command *Contributors:* Christina.biermair, Gsturr, JohannaZ, Markus.winkler, Noel Lambert, Zbynek

TMeanEstimate Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TMeanEstimate_Command *Contributors:* Gsturr, JohannaZ, Markus.winkler, Noel Lambert, Zbynek

TTest Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TTest_Command *Contributors:* Andrea.duringer, Gsturr, JohannaZ, Markus.winkler, Mathmum, Noel Lambert, Zbynek

TTest2 Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TTest2_Command *Contributors:* Andrea.duringer, Gsturr, JohannaZ, Markus.winkler, Mathmum, Noel Lambert, Zbynek

TTestPaired Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TTestPaired_Command *Contributors:* Andrea.duringer, Gsturr, JohannaZ, Markus.winkler, Mathmum, Noel Lambert, Zbynek

Variance Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Variance_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Noel Lambert, Spanish1, Zbynek

Probability Commands *Source:* https://wiki.geogebra.org/s/en/index.php?title=Probability_Commands *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Zbynek

Bernoulli Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Bernoulli_Command *Contributors:* Markus.winkler, Zbynek

BinomialCoefficient Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=BinomialCoefficient_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

BinomialDist Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=BinomialDist_Command *Contributors:* Alexander Hartl, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Zbynek

Cauchy Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Cauchy_Command *Contributors:* Alexander Hartl, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Zbynek

ChiSquared Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ChiSquared_Command *Contributors:* Alexander Hartl, Christina.biermair, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Zbynek

ChiSquaredTest Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ChiSquaredTest_Command *Contributors:* Gsturr, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Spanish1, Zbynek

Erlang Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Erlang_Command *Contributors:* Corinna, JohannaZ, Markus.winkler, Murkle, Zbynek

Exponential Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Exponential_Command *Contributors:* Alexander Hartl, Cmiic, JohannaZ, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

FDistribution Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=FDistribution_Command *Contributors:* Alexander Hartl, Christina.biermair, JohannaZ, MagdalenaSophieF, Markus.winkler, Murkle, Zbynek

Gamma Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Gamma_Command *Contributors:* Alexander Hartl, Christina.biermair, JohannaZ, Markus.winkler, Murkle, Zbynek

HyperGeometric Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=HyperGeometric_Command *Contributors:* Alexander Hartl, Christina.biermair, JohannaZ, Kimeswenger, Markus.winkler, Mathmum, Zbynek

InverseBinomial Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=InverseBinomial_Command *Contributors:* JohannaZ, Markus.winkler, Spanish1, Zbynek

InverseCauchy Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=InverseCauchy_Command *Contributors:* JohannaZ, Markus.winkler, Zbynek

InverseChiSquared Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=InverseChiSquared_Command *Contributors:* JohannaZ, Markus.winkler, Zbynek

InverseExponential Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=InverseExponential_Command *Contributors:* JohannaZ, Markus.winkler, Zbynek

InverseFDistribution Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=InverseFDistribution_Command *Contributors:* JohannaZ, Markus.winkler, Zbynek

InverseGamma Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=InverseGamma_Command *Contributors:* JohannaZ, Markus.winkler, Zbynek

InverseHyperGeometric Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=InverseHyperGeometric_Command *Contributors:* JohannaZ, Markus.winkler, Zbynek

InverseLogNormal Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=InverseLogNormal_Command *Contributors:* Andrea.duringer, JohannaZ, Markus.winkler, Mathmum, Murkle, Spanish1, Zbynek

InverseLogistic Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=InverseLogistic_Command *Contributors:* Andrea.duringer, JohannaZ, Markus.winkler, Mathmum, Murkle, Zbynek

InverseNormal Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=InverseNormal_Command *Contributors:* JohannaZ, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Zbynek

InversePascal Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=InversePascal_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Murkle, Zbynek

InversePoisson Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=InversePoisson_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Zbynek

InverseTDistribution Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=InverseTDistribution_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Zbynek

InverseWeibull Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=InverseWeibull_Command *Contributors:* Andrea.duringer, MagdalenaSophieF, Markus.winkler, Murkle, Zbynek

InverseZipf Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=InverseZipf_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Zbynek

LogNormal Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=LogNormal_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Murkle, Spanish1, Zbynek

Logistic Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Logistic_Command *Contributors:* JohannaZ, Markus.winkler, Murkle, Sabrina.azesberger, Zbynek

Normal Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Normal_Command *Contributors:* Alexander Hartl, Andrea.duringer, Christina.biermair, JohannaZ, K Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Zbynek

Pascal Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Pascal_Command *Contributors:* Alexander Hartl, Christina.biermair, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Zbynek

Poisson Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Poisson_Command *Contributors:* Alexander Hartl, Christina.biermair, JohannaZ, Markus.winkler, Mathmum, Spanish1, Zbynek

RandomBetween Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=RandomBetween_Command *Contributors:* Andrea.duringer, JohannaZ, K Voss, Kimeswenger, Markus.winkler, Mathmum, Murkle, Spanish1, Zbynek

RandomBinomial Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=RandomBinomial_Command *Contributors:* Alexander Hartl, Christina.biermair, JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, Spanish1, UnTom, Zbynek

RandomNormal Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=RandomNormal_Command *Contributors:* Christina.biermair, JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, UnTom, Zbynek

RandomPoisson Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=RandomPoisson_Command *Contributors:* Christina.biermair, JohannaZ, K Voss, Markus.winkler, Mathmum, Murkle, UnTom, Zbynek

RandomUniform Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=RandomUniform_Command *Contributors:* JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Zbynek

TDistribution Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TDistribution_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

Triangular Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Triangular_Command *Contributors:* Andrea.duringer, Corinna, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Zbynek

Uniform Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Uniform_Command *Contributors:* Markus.winkler, Mathmum, Murkle, Zbynek

Weibull Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Weibull_Command *Contributors:* Alexander Hartl, JohannaZ, Kimeswenger, Markus.winkler, Mathmum, Murkle, Zbynek

Zipf Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Zipf_Command *Contributors:* Alexander Hartl, JohannaZ, Markus.winkler, Mathmum, Sabrina.azesberger, Zbynek

ZMean2Estimate Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ZMean2Estimate_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Murkle, Sabrina.azesberger, Spanish1, Zbynek

ZMean2Test Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ZMean2Test_Command *Contributors:* Andrea.duringer, JohannaZ, Markus.winkler, Murkle, Sabrina.azesberger, Zbynek

ZProportion2Estimate Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ZProportion2Estimate_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Murkle, Zbynek

ZProportion2Test Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ZProportion2Test_Command *Contributors:* Andrea.duringer, Gsturr, Markus.winkler, Mathmum, Murkle, Zbynek

ZProportionEstimate Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ZProportionEstimate_Command *Contributors:* Gsturr, Markus.winkler, Spanish1, Zbynek

ZProportionTest Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ZProportionTest_Command *Contributors:* Andrea.duringer, Gsturr, Markus.winkler, Mathmum, Murkle, Zbynek

ZMeanEstimate Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ZMeanEstimate_Command *Contributors:* JohannaZ, Markus.winkler, Murkle, Sabrina.azesberger, Zbynek

ZMeanTest Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ZMeanTest_Command *Contributors:* Andrea.duringer, JohannaZ, Markus.winkler, Mathmum, Murkle, Zbynek

Spreadsheet Commands *Source:* https://wiki.geogebra.org/s/en/index.php?title=Spreadsheet_Commands *Contributors:* Markus.winkler, Mathmum, Zbynek

Cell Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Cell_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Murkle, Zbynek

CellRange Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=CellRange_Command *Contributors:* JohannaZ, K Voss, Markus.winkler, Mathmum, Spanish1, Zbynek

Column Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Column_Command *Contributors:* K Voss, Markus.winkler, Mathmum, Spanish1, Zbynek

ColumnName Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ColumnName_Command *Contributors:* K Voss, Markus.winkler, Mathmum, Spanish1, Zbynek

FillCells Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=FillCells_Command *Contributors:* Andrea.duringer, JohannaZ, Markus.winkler, Mathmum, Murkle, Noel Lambert, Spanish1, Zbynek

FillColumn Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=FillColumn_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

FillRow Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=FillRow_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

Row Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Row_Command *Contributors:* K Voss, Markus.winkler, Mathmum, Spanish1, Zbynek

Scripting Commands *Source:* https://wiki.geogebra.org/s/en/index.php?title=Scripting_Commands *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Murkle, Noel Lambert, Spanish1, Zbynek

Button Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Button_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Mathmum, Zbynek

CenterView Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=CenterView_Command *Contributors:* Andrea.duringer, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Spanish1, Zbynek

Checkbox Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Checkbox_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

CopyFreeObject Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=CopyFreeObject_Command *Contributors:* Markus.winkler, Zbynek

Delete Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Delete_Command *Contributors:* Alexander Hartl, Christina.biermair, JohannaZ, K Voss, Markus.winkler, Mathmum, Spanish1, Zbynek

Execute Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Execute_Command *Contributors:* Gsturr, Jhohen, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, Spanish1, Zbynek

GetTime Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=GetTime_Command *Contributors:* Andrea.duringer, JohannaZ, Markus.winkler, Mathmum, Murkle, Zbynek

HideLayer Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=HideLayer_Command *Contributors:* Markus.winkler, Zbynek

Pan Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Pan_Command *Contributors:* Markus.winkler, Murkle, Zbynek

ParseToFunction Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ParseToFunction_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Noel Lambert, Zbynek

ParseToNumber Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ParseToNumber_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Zbynek

Repeat Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Repeat_Command *Contributors:* Markus.winkler, Mathmum, Noel Lambert, Zbynek

PlaySound Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=PlaySound_Command *Contributors:* Gsturr, Markus.winkler, Mathmum, Murkle, Zbynek

Rename Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Rename_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Zbynek

RunClickScript Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=RunClickScript_Command *Contributors:* Markus.winkler, Mathmum, Murkle, Noel Lambert, SarahSt, Zbynek

RunUpdateScript Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=RunUpdateScript_Command *Contributors:* Markus.winkler, Murkle, Noel Lambert, Zbynek

Turtle Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Turtle_Command *Contributors:* Markus.winkler, Mathmum, Murkle, Sabsi, Sarah., Zbynek

TurtleLeft Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TurtleLeft_Command *Contributors:* Jhohen, Markus.winkler, Mathmum, Noel Lambert, Sabsi, Sarah., Zbynek

TurtleUp Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TurtleUp_Command *Contributors:* Markus.winkler, Noel Lambert, Zbynek

TurtleRight Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TurtleRight_Command *Contributors:* Markus.winkler, Mathmum, Noel Lambert, Sabsi, Sarah., Zbynek

TurtleDown Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TurtleDown_Command *Contributors:* Markus.winkler, Noel Lambert, Zbynek

TurtleForward Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TurtleForward_Command *Contributors:* Markus.winkler, Mathmum, Noel Lambert, Sabsi, Sarah., Zbynek

TurtleBack Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TurtleBack_Command *Contributors:* Markus.winkler, Mathmum, Noel Lambert, Sabsi, Sarah., Zbynek

SelectObjects Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SelectObjects_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Zbynek

SetActiveView Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SetActiveView_Command *Contributors:* Markus.winkler, Murkle, Noel Lambert, Zbynek

SetAxesRatio Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SetAxesRatio_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Zbynek

SetBackgroundColor Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SetBackgroundColor_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel Lambert, Zbynek

SetCaption Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SetCaption_Command *Contributors:* Markus.winkler, Mathmum, Zbynek

SetColor Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SetColor_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Zbynek

SetConditionToShowObject Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SetConditionToShowObject_Command *Contributors:* Markus.winkler, Zbynek

SetCoords Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SetCoords_Command *Contributors:* JohannaZ, Markus.winkler, Murkle, Zbynek

SetDynamicColor Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SetDynamicColor_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Murkle, Spanish1, Zbynek

SetFilling Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SetFilling_Command *Contributors:* Markus.winkler, Mathmum, Zbynek

SetFixed Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SetFixed_Command *Contributors:* Markus.winkler, Murkle, Zbynek

SetLabelMode Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SetLabelMode_Command *Contributors:* Markus.winkler, Mathmum, Noel Lambert, Zbynek

SetLayer Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SetLayer_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Zbynek

SetLineStyle Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SetLineStyle_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Zbynek

SetLineThickness Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SetLineThickness_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Mathieu.Blossier, Mathmum, Murkle, Noel.Lambert, Zbynek

SetPointSize Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SetPointSize_Command *Contributors:* Markus.winkler, Mathieu.Blossier, Mathmum, Murkle, Zbynek

SetPointStyle Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SetPointStyle_Command *Contributors:* Markus.winkler, Murkle, Noel.Lambert, Zbynek

SetTooltipMode Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SetTooltipMode_Command *Contributors:* Markus.winkler, Zbynek

SetValue Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SetValue_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Spanish1, Zbynek

ShowVisibleInView Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ShowVisibleInView_Command *Contributors:* Markus.winkler, Murkle, Zbynek

ShowViewDirection Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ShowViewDirection_Command *Contributors:* Johanna, Markus.winkler, Mathmum, Murkle, Sarah., Zbynek

ShowSpinSpeed Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ShowSpinSpeed_Command *Contributors:* Markus.winkler, Mathmum, Noel.Lambert, Zbynek

ShowPerspective Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ShowPerspective_Command *Contributors:* Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

ShowSeed Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ShowSeed_Command *Contributors:* Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

ShowTrace Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ShowTrace_Command *Contributors:* Andrea.duringer, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

ShowAxes Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ShowAxes_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

ShowGrid Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ShowGrid_Command *Contributors:* JohannaZ, Markus, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

ShowLabel Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ShowLabel_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Murkle, Zbynek

ShowLayer Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ShowLayer_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Zbynek

Slider Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Slider_Command *Contributors:* Markus.winkler, Mathmum, Murkle, Zbynek

StartAnimation Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=StartAnimation_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Zbynek

StartLogging Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=StartLogging_Command *Contributors:* -

StopLogging Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=StopLogging_Command *Contributors:* -

StartRecord Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=StartRecord_Command *Contributors:* Markus.winkler, Mathieu.Blossier, Murkle, Zbynek

InputBox Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=InputBox_Command *Contributors:* Andrea.duringer, Birgit.Lachner, Markus.winkler, Spanish1, Zbynek

UpdateConstruction Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=UpdateConstruction_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Sarah., Zbynek

ZoomIn Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ZoomIn_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Murkle, Zbynek

ZoomOut Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ZoomOut_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Zbynek

Discrete Math Commands *Source:* https://wiki.geogebra.org/s/en/index.php?title=Discrete_Math_Commands *Contributors:* Markus.winkler, Mathmum, Zbynek

ConvexHull Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ConvexHull_Command *Contributors:* Markus.winkler, Mathmum, Noel.Lambert, Zbynek

DelaunayTriangulation Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=DelaunayTriangulation_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Mathmum, Noel.Lambert, Zbynek

Hull Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Hull_Command *Contributors:* -

MinimumSpanningTree Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=MinimumSpanningTree_Command *Contributors:* Markus.winkler, Mathmum, Noel.Lambert, Sabrina.azesberger, Zbynek

ShortestDistance Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ShortestDistance_Command *Contributors:* Markus.winkler, Mathmum, Noel.Lambert, Zbynek

TravelingSalesman Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=TravelingSalesman_Command *Contributors:* Markus.winkler, Mathmum, Noel.Lambert, Zbynek

Voronoi Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Voronoi_Command *Contributors:* Markus.winkler, Mathmum, Noel.Lambert, Zbynek

GeoGebra Commands *Source:* https://wiki.geogebra.org/s/en/index.php?title=GeoGebra_Commands *Contributors:* Markus.winkler, Zbynek

AxisStepX Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=AxisStepX_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Mathmum, Zbynek

AxisStepY Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=AxisStepY_Command *Contributors:* MagdalenaSophieF, Markus.winkler, Mathmum, Zbynek

ClosestPointRegion Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ClosestPointRegion_Command *Contributors:* Markus.winkler, Murkle, Zbynek

ConstructionStep Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ConstructionStep_Command *Contributors:* JohannaZ, K.Voss, Markus.winkler, Mathmum, Zbynek

Corner Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Corner_Command *Contributors:* Johanna, JohannaZ, K.Voss, Markus.winkler, Mathieu.Blossier, Mathmum, Murkle, Noel.Lambert, Zbynek

DynamicCoordinates Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=DynamicCoordinates_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Spanish1, Zbynek

Name Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Name_Command *Contributors:* K.Voss, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Sabrina.azesberger, Spanish1, Zbynek

Object Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Object_Command *Contributors:* K.Voss, Markus.winkler, Mathmum, Murkle, Spanish1, Zbynek

SlowPlot Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SlowPlot_Command *Contributors:* Markus, Markus.winkler, Mathmum, Zbynek

ToolImage Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ToolImage_Command *Contributors:* Gabriela.Ferenczy, Markus.winkler, Mathmum, Noel.Lambert, Zbynek

Optimization Commands *Source:* https://wiki.geogebra.org/s/en/index.php?title=Optimization_Commands *Contributors:* Markus.winkler, Mathmum, Zbynek

Maximize Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Maximize_Command *Contributors:* MagdalenaSophieF, Mathmum, Murkle, Noel Lambert, Sabrina.azesberger, Zbynek

Minimize Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Minimize_Command *Contributors:* MagdalenaSophieF, Mathmum, Murkle, Noel Lambert, Sabrina.azesberger, Zbynek

CAS Specific Commands *Source:* https://wiki.geogebra.org/s/en/index.php?title=CAS_Specific_Commands *Contributors:* Alexander.Hartl, Andrea.duringer, Christina.biermair, Florian.Sonner, Mathmum, Spanish1, Zbynek, Zoltan

CFactor Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=CFactor_Command *Contributors:* Alexander.Hartl, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Spanish1, UnTom, Zbynek

CIFactor Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=CIFactor_Command *Contributors:* JohannaZ, Markus.winkler, Mathmum, Noel.Lambert, Zbynek

CSolutions Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=CSolutions_Command *Contributors:* Alexander.Hartl, Andrea.duringer, Christina.biermair, JohannaZ, Markus.winkler, Mathmum, Noel.Lambert, Spanish1, Zbynek

CSolve Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=CSolve_Command *Contributors:* Alexander.Hartl, Andrea.duringer, Christina.biermair, JohannaZ, Markus.winkler, Mathmum, Noel.Lambert, UnTom, Zbynek

CommonDenominator Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=CommonDenominator_Command *Contributors:* Alexander.Hartl, JohannaZ, Markus.winkler, Mathmum, Murkle, Zbynek

Cross Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Cross_Command *Contributors:* Alexander.Hartl, Christina.biermair, JohannaZ, Markus.winkler, Mathmum, Spanish1, Zbynek

Dimension Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Dimension_Command *Contributors:* Alexander.Hartl, Andrea.duringer, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Zbynek

Division Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Division_Command *Contributors:* Alexander.Hartl, Andrea.duringer, Markus.winkler, Mathmum, Murkle, Zbynek, Zoltan

Divisors Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Divisors_Command *Contributors:* Alexander.Hartl, Andrea.duringer, JohannaZ, Markus.winkler, Mathmum, Murkle, Zbynek

DivisorsList Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=DivisorsList_Command *Contributors:* Alexander.Hartl, Andrea.duringer, JohannaZ, Markus.winkler, Mathmum, Murkle, Zbynek

DivisorsSum Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=DivisorsSum_Command *Contributors:* Alexander.Hartl, Andrea.duringer, JohannaZ, Markus.winkler, Mathmum, Murkle, Zbynek

Dot Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Dot_Command *Contributors:* Alexander.Hartl, JohannaZ, Markus.winkler, Mathmum, Murkle, Zbynek

Eliminate Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Eliminate_Command *Contributors:* Johanna, Markus.winkler, Mathmum, Zbynek, Zoltan.Kovacs

GroebnerDegRevLex Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=GroebnerDegRevLex_Command *Contributors:* Johanna, Markus.winkler, Mathmum, Noel.Lambert, Zbynek, Zoltan.Kovacs

GroebnerLex Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=GroebnerLex_Command *Contributors:* Johanna, Markus.winkler, Mathmum, Zbynek, Zoltan.Kovacs

GroebnerLexDeg Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=GroebnerLexDeg_Command *Contributors:* Johanna, Markus.winkler, Mathmum, Noel.Lambert, Zbynek, Zoltan.Kovacs

ImplicitDerivative Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ImplicitDerivative_Command *Contributors:* Alexander.Hartl, Andrea.duringer, JohannaZ, Markus.winkler, Mathmum, Murkle, Zbynek

InverseLaplace Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=InverseLaplace_Command *Contributors:* Johanna, Markus.winkler, Mathmum, Noel.Lambert, Zbynek

IsPrime Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=IsPrime_Command *Contributors:* Alexander.Hartl, Andrea.duringer, Christina.biermair, Markus.winkler, Mathmum, Murkle, Zbynek

Laplace Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Laplace_Command *Contributors:* Johanna, Markus.winkler, Mathmum, Noel.Lambert, Sabsi, Zbynek

LeftSide Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=LeftSide_Command *Contributors:* Alexander.Hartl, Andrea.duringer, JohannaZ, Kimeswenger, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Spanish1, Zbynek

MatrixRank Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=MatrixRank_Command *Contributors:* Alexander.Hartl, Andrea.duringer, Christina.biermair, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Sabrina.azesberger, Spanish1, Zbynek

MixedNumber Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=MixedNumber_Command *Contributors:* Alexander.Hartl, Corinna, JohannaZ, Markus.winkler, Mathmum, Sabrina.azesberger, Zbynek

NIntegral Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=NIntegral_Command *Contributors:* Alexander.Hartl, GeoGebra Docu Team, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Spanish1, Zbynek

NSolutions Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=NSolutions_Command *Contributors:* Alexander.Hartl, Christina.biermair, Gabriela.Ferenczy, JohannaZ, Markus.winkler, Mathmum, Murkle, Spanish1, Zbynek

NSolve Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=NSolve_Command *Contributors:* Alexander.Hartl, Christina.biermair, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Sabrina.azesberger, UnTom, Zbynek

NextPrime Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=NextPrime_Command *Contributors:* Alexander.Hartl, Andrea.duringer, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Zbynek

Numeric Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Numeric_Command *Contributors:* Alexander.Hartl, Christina.biermair, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Spanish1, Zbynek

PreviousPrime Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=PreviousPrime_Command *Contributors:* Alexander.Hartl, Andrea.duringer, JohannaZ, Markus.winkler, Mathmum, Murkle, Zbynek

RandomPolynomial Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=RandomPolynomial_Command *Contributors:* Alexander.Hartl, Andrea.duringer, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Simon, Spanish1, UnTom, Zbynek

Rationalize Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Rationalize_Command *Contributors:* Corinna, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Spanish1, UnTom, Zbynek

RightSide Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=RightSide_Command *Contributors:* Alexander.Hartl, Andrea.duringer, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Spanish1, Zbynek

Solutions Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Solutions_Command *Contributors:* Alexander.Hartl, Christina.biermair, Gabriela.Ferenczy, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel.Lambert, UnTom, Zbynek

Solve Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Solve_Command *Contributors:* Alexander.Hartl, Balazs.bencze, Christina.biermair, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Solyom-Gecse.Csilla, UnTom, Zbynek

SolveCubic Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=SolveCubic_Command *Contributors:* Johanna, Markus.winkler, Murkle, Zbynek

Substitute Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Substitute_Command *Contributors:* Alexander.Hartl, Christina.biermair, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

ToComplex Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ToComplex_Command *Contributors:* Alexander.Hartl, Andrea.duringer, Christina.biermair, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Spanish1, Zbynek

ToExponential Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ToExponential_Command *Contributors:* Alexander.Hartl, Andrea.duringer, Christina.biermair, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Spanish1, Zbynek

ToPoint Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ToPoint_Command *Contributors:* Alexander.Hartl, Andrea.duringer, Christina.biermair, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Spanish1, Zbynek

ToPolar Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=ToPolar_Command *Contributors:* Alexander.Hartl, Andrea.duringer, Christina.biermair, JohannaZ, MagdalenaSophieF, Markus.winkler, Mathmum, UnTom, Zbynek

3D Commands *Source:* https://wiki.geogebra.org/s/en/index.php?title=3D_Commands *Contributors:* Zbynek

Plane Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Plane_Command *Contributors:* Johanna, Markus.winkler, Mathmum, Noel.Lambert, Sarah., Zbynek

Cone Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Cone_Command *Contributors:* Johanna, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Spanish1, Zbynek

Height Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Height_Command *Contributors:* Johanna, Markus.winkler, Mathmum, Noel.Lambert, Zbynek

Sphere Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Sphere_Command *Contributors:* Johanna, Markus.winkler, Mathmum, Spanish1, Zbynek

Net Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Net_Command *Contributors:* Markus.winkler, Mathmum, Noel.Lambert, Zbynek

Top Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Top_Command *Contributors:* Johanna, Markus.winkler, Mathmum, Noel.Lambert, Sabsi, Sarah., Spanish1, Zbynek

Surface Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Surface_Command *Contributors:* Markus.winkler, Mathmum, Noel.Lambert, Sabsi, Spanish1, Zbynek

Octahedron Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Octahedron_Command *Contributors:* Markus.winkler, Mathieu.Blossier, Mathmum, Noel.Lambert, Sabsi, Spanish1, Zbynek

InfiniteCone Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=InfiniteCone_Command *Contributors:* Markus.winkler, Mathmum, Murkle, Sarah., Spanish1, Zbynek

InfiniteCylinder Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=InfiniteCylinder_Command *Contributors:* Markus.winkler, Mathmum, Murkle, Sarah., Spanish1, Zbynek

IntersectConic Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=IntersectConic_Command *Contributors:* Markus.winkler, Mathmum, Sarah., Zbynek

Side Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Side_Command *Contributors:* Johanna, Markus.winkler, Mathmum, Noel.Lambert, Sabsi, Zbynek

PerpendicularPlane Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=PerpendicularPlane_Command *Contributors:* Johanna, Markus.winkler, Mathmum, Murkle, Spanish1, Zbynek

Polyhedron Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Polyhedron_Command *Contributors:* -

Prism Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Prism_Command *Contributors:* Markus.winkler, Mathmum, Sabsi, Sarah., Zbynek

Dodecahedron Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Dodecahedron_Command *Contributors:* Markus.winkler, Mathieu, Mathieu.Blossier, Mathmum, Noel.Lambert, Sabsi, Spanish1, Zbynek

Icosahedron Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Icosahedron_Command *Contributors:* Markus.winkler, Mathieu.Blossier, Mathmum, Noel.Lambert, Sabsi, Spanish1, Zbynek

Tetrahedron Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Tetrahedron_Command *Contributors:* Markus.winkler, Mathieu.Blossier, Mathmum, Noel.Lambert, Sabsi, Spanish1, Zbynek

Bottom Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Bottom_Command *Contributors:* Johanna, Markus.winkler, Mathmum, Sabsi, Spanish1, Zbynek

Volume Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Volume_Command *Contributors:* Johanna, Markus.winkler, Mathmum, Sarah., Zbynek

Cube Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Cube_Command *Contributors:* Markus.winkler, Mathieu, Mathieu.Blossier, Mathmum, Murkle, Noel.Lambert, Sabsi, Spanish1, Zbynek

Cylinder Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Cylinder_Command *Contributors:* Johanna, Markus.winkler, Mathmum, Murkle, Sarah., Zbynek

Pyramid Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Pyramid_Command *Contributors:* Markus.winkler, Mathmum, Sabsi, Sarah., Zbynek

Ends Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=Ends_Command *Contributors:* Johanna, Markus.winkler, Mathmum, Noel.Lambert, Sabsi, Sarah., Zbynek

PlaneBisector Command *Source:* https://wiki.geogebra.org/s/en/index.php?title=PlaneBisector_Command *Contributors:* Johanna, Markus.winkler, Mathmum, Sarah., Spanish1, Zbynek

Predefined Functions and Operators *Source:* https://wiki.geogebra.org/s/en/index.php?title=Predefined_Functions_and_Operators *Contributors:* Alexander.Hartl, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

Imaginary Function *Source:* https://wiki.geogebra.org/s/en/index.php?title=Imaginary_Function *Contributors:* Alexander.Hartl, Christina.biermair, Markus.winkler, Mathmum, Spanish1, Zbynek

FractionalPart Function *Source:* https://wiki.geogebra.org/s/en/index.php?title=FractionalPart_Function *Contributors:* Markus.winkler, Mathmum, Murkle, Spanish1, Zbynek

Nroot Function *Source:* https://wiki.geogebra.org/s/en/index.php?title=Nroot_Function *Contributors:* Markus.winkler, Murkle, Noel.Lambert, Spanish1, Zbynek

Real Function *Source:* https://wiki.geogebra.org/s/en/index.php?title=Real_Function *Contributors:* Alexander.Hartl, Andrea.duringer, Markus.winkler, Mathmum, Spanish1, Zbynek

Views *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Views> *Contributors:* Jhohen, Markus.winkler, Mathmum, Murkle, Sabrina.azesberger, Zbynek

Graphics View *Source:* https://wiki.geogebra.org/s/en/index.php?title=Graphics_View *Contributors:* Andrea.duringer, Christina.biermair, Corinna, Florian.Sonner, Jhohen, Markus, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

Customizing the Graphics View *Source:* https://wiki.geogebra.org/s/en/index.php?title=Customizing_the_Graphics_View *Contributors:* Markus.winkler, Mathmum, Zbynek

Algebra View *Source:* https://wiki.geogebra.org/s/en/index.php?title=Algebra_View *Contributors:* Jhohen, Markus.winkler, Mathmum, Murkle, Spanish1, Zbynek

Spreadsheet View *Source:* https://wiki.geogebra.org/s/en/index.php?title=Spreadsheet_View *Contributors:* Jhohen, Markus.winkler, Mathmum, Murkle, Zbynek

CAS View *Source:* https://wiki.geogebra.org/s/en/index.php?title=CAS_View *Contributors:* Alexander.Hartl, Andrea.duringer, Christina.biermair, Florian.Sonner, Jhohen, Johannes, Markus.winkler, Mathmum, Murkle, Noel.Lambert, T.wassermair, Zbynek

Probability Calculator *Source:* https://wiki.geogebra.org/s/en/index.php?title=Probability_Calculator *Contributors:* Jhohen, Markus.winkler, Mathmum

3D Graphics View *Source:* https://wiki.geogebra.org/s/en/index.php?title=3D_Graphics_View *Contributors:* Jhohen, Markus.winkler, Mathmum, Noel.Lambert

Construction Protocol *Source:* https://wiki.geogebra.org/s/en/index.php?title=Construction_Protocol *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Zbynek

Input Bar *Source:* https://wiki.geogebra.org/s/en/index.php?title=Input_Bar *Contributors:* Jhohen, Markus.winkler, Mathmum, Murkle, Zbynek

Menubar *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Menubar> *Contributors:* Markus.winkler, Mathmum, Zbynek

Toolbar *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Toolbar> *Contributors:* Jhohen, Markus.winkler, Mathmum, Zbynek

Style Bar *Source:* https://wiki.geogebra.org/s/en/index.php?title=Style_Bar *Contributors:* Jhohen, Markus.winkler, Mathmum

Navigation Bar *Source:* https://wiki.geogebra.org/s/en/index.php?title=Navigation_Bar *Contributors:* Kimeswenger, Markus.winkler, Mathmum, Noel.Lambert, Sabrina.azesberger, Spanish1, Zbynek

File Menu *Source:* https://wiki.geogebra.org/s/en/index.php?title=File_Menu *Contributors:* Christina.biermair, Florian.Sonner, Markus.winkler, Mathmum, Murkle, Sarah., Zbynek

Edit Menu *Source:* https://wiki.geogebra.org/s/en/index.php?title=Edit_Menu *Contributors:* Florian.Sonner, Markus.winkler, Mathmum, Murkle, Zbynek

View Menu *Source:* https://wiki.geogebra.org/s/en/index.php?title=View_Menu *Contributors:* Christina.biermair, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

Perspectives *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Perspectives> *Contributors:* Corinna, Jhohen, Markus.winkler, Mathmum, Zbynek

Options Menu *Source:* https://wiki.geogebra.org/s/en/index.php?title=Options_Menu *Contributors:* Jhohen, Markus.winkler, Mathmum, Sabrina.azesberger, Zbynek

Tools Menu *Source:* https://wiki.geogebra.org/s/en/index.php?title=Tools_Menu *Contributors:* Markus.winkler, Mathmum, Sabrina.azesberger, Zbynek

Window Menu *Source:* https://wiki.geogebra.org/s/en/index.php?title=Window_Menu *Contributors:* Markus.winkler, Mathmum, Sabrina.azesberger, Zbynek

Help Menu *Source:* https://wiki.geogebra.org/s/en/index.php?title=Help_Menu *Contributors:* Markus.winkler, Murkle, Zbynek

Context Menu *Source:* https://wiki.geogebra.org/s/en/index.php?title=Context_Menu *Contributors:* Jhohen, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

Customize the Settings *Source:* https://wiki.geogebra.org/s/en/index.php?title=Customize_the_Settings *Contributors:* Markus.winkler, Mathmum, Zbynek

Export Graphics Dialog *Source:* https://wiki.geogebra.org/s/en/index.php?title=Export_Graphics_Dialog *Contributors:* Markus.winkler, Mathmum, Murkle, Zbynek

Export Worksheet Dialog *Source:* https://wiki.geogebra.org/s/en/index.php?title=Export_Worksheet_Dialog *Contributors:* Andrea.duringer, Christina.biermair, Kimeswenger, Markus.winkler, Mathmum, Zbynek

Properties Dialog *Source:* https://wiki.geogebra.org/s/en/index.php?title=Properties_Dialog *Contributors:* Andrea.duringer, Jhohen, Markus.winkler, Mathmum, Murkle, Zbynek

Redefine Dialog *Source:* https://wiki.geogebra.org/s/en/index.php?title=Redefine_Dialog *Contributors:* Andrea.duringer, Markus.winkler, Mathmum, Murkle, Zbynek

Tool Creation Dialog *Source:* https://wiki.geogebra.org/s/en/index.php?title=Tool_Creation_Dialog *Contributors:* Markus.winkler, Mathmum, Sabrina.azesberger, Zbynek

Keyboard Shortcuts *Source:* https://wiki.geogebra.org/s/en/index.php?title=Keyboard_Shortcuts *Contributors:* Andrea.duringer, Johannes, Markus.winkler, Mathmum, Murkle, Noel.Lambert, Zbynek

Settings Dialog *Source:* https://wiki.geogebra.org/s/en/index.php?title=Settings_Dialog *Contributors:* Corinna, Markus.winkler, Mathmum, Sabrina.azesberger, Zbynek

Virtual Keyboard *Source:* https://wiki.geogebra.org/s/en/index.php?title=Virtual_Keyboard *Contributors:* Markus.winkler, Mathmum, Sabrina.azesberger, Zbynek

Tool Manager Dialog *Source:* https://wiki.geogebra.org/s/en/index.php?title=Tool_Manager_Dialog *Contributors:* Andrea.duringer, Markus.winkler, Sabrina.azesberger, Zbynek

Sidebar *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Sidebar> *Contributors:* Markus.winkler, Mathmum, Noel.Lambert, Sabrina.azesberger, Spanish1, Zbynek

Open Dialog - Insert File *Source:* https://wiki.geogebra.org/s/en/index.php?title=Open_Dialog_-_Insert_File *Contributors:* Markus.winkler, Mathmum, Sarah., Zbynek

Open Dialog - Style Templates *Source:* https://wiki.geogebra.org/s/en/index.php?title=Open_Dialog_-_Style_Templates *Contributors:* Markus.winkler, Mathmum, Murkle, Sarah., Zbynek

Accessibility *Source:* <https://wiki.geogebra.org/s/en/index.php?title=Accessibility> *Contributors:* Corinna, Jhohen, Markus.winkler, Mathmum, Murkle, Zbynek

Creating Pictures of the Graphics View *Source:* https://wiki.geogebra.org/s/en/index.php?title=Creating_Pictures_of_the_Graphics_View *Contributors:* Markus.winkler, Mathmum, Zbynek

Embedding in Webpages *Source:* https://wiki.geogebra.org/s/en/index.php?title=Embedding_in_Webpages *Contributors:* Christina.biermair, Markus.winkler, Mathmum, Murkle, Zbynek

Embedding to CMS, VLE (Moodle) and Wiki *Source:* https://wiki.geogebra.org/s/en/index.php?title=Embedding_to_CMS%2C_VLE_%28Moodle%29_and_Wiki *Contributors:* Christina.biermair, Markus.winkler, Mathmum, Murkle, Zbynek

Export to LaTeX (PGF, PSTricks) and Asymptote *Source:* https://wiki.geogebra.org/s/en/index.php?title=Export_to_LaTeX_%28PGF%2C_PSTricks%29_and_Asymptote *Contributors:* Christina.biermair, Markus.winkler, Mathmum, Murkle, Zbynek

Printing Options *Source:* https://wiki.geogebra.org/s/en/index.php?title=Printing_Options *Contributors:* Markus.winkler, Mathmum, Sabrina.azesberger, Zbynek