GeoGebra Help

Official Manual 3.0

Markus Hohenwarter and Judith Hohenwarter
www.geogebra.org

GeoGebra Help 3.0

Last modified: October 15, 2007

Authors
Markus Hohenwarter, markus@geogebra.org
Judith Hohenwarter, judith@geogebra.org

GeoGebra Online
Website: www.geogebra.org
Help Search: http://www.geogebra.org/help/search.html

Contents

1. What is GeoGebra?

GeoGebra is dynamic mathematics software that joins geometry, algebra and calculus. It is developed for mathematics learning and teaching in schools by Markus Hohenwarter at Florida Atlantic University.

On the one hand, GeoGebra is a dynamic geometry system. You can do constructions with points, vectors, segments, lines, conic sections, as well as functions, and change them dynamically afterwards.

On the other hand, equations and coordinates can be entered directly. Thus, GeoGebra has the ability to deal with variables for numbers, vectors, and points, finds derivatives and integrals of functions, and offers commands like *Root* or *Extr*emum.

These two views are characteristic of GeoGebra: an expression in the algebra window corresponds to an object in the geometry window and vice versa.

2. Examples

Let's have a look at some examples in order to get an impression of GeoGebra's possibilities.

2.1. Triangle with Angles

Select mode *New point* in the toolbar. Click three times on the drawing pad to create the three vertices *A*, *B,* and *C* of the triangle.
Afterwards, select mode *Polygon* and successively click on points *A*, *B*, and *C*. To close triangle *poly1* click on the starting point *A* again. In the algebra window the area of the triangle is shown.

In order to get all the angles of a triangle, choose mode *Angle* in the toolbar and click on the triangle.

Now, choose *Move* mode and drag the vertices to modify the triangle dynamically. If you don't need the algebra window and coordinate axes, hide them by using the *View* menu.

2.2. Linear Equation y = m x + b

We will now concentrate on the meaning of *m* and *b* in the linear equation *y* = *mx* + *b* by trying different values for *m* and *b*. To do this we might enter the following lines in the input field at the screen's bottom and press the *Enter* key at the end of each line.

```
m = 1
b = 2
y = m x + b
```

Now we can change *m* and *b* using the input field or directly in the algebra window by right clicking (MacOS: *Apple* + click) one of the numbers and selecting *Redefine*. Try the following values for *m* and *b*.

```
m = 2
m = -3
b = 0
b = -1
```

Also, you can change *m* and *b* very easily using

- the arrow keys (see Animation)
- sliders: right click (MacOS: *Apple* + click) on *m* or *b* and select *Show / hide object* (also see mode Slider)

In a similar way we might investigate the equations of conic sections such as

- ellipses: `x^2/a^2 + y^2/b^2 = 1`
- hyperbolas: `b^2 x^2 – a^2 y^2 = a^2 b^2` or
- circles: `(x – m)^2 + (y – n)^2 = r^2`

2.3. Centroid of Three Points A, B, and C

We are now going to construct the centroid of three points by entering the following lines in the input field and pressing the *Enter* key at the end of each line. Of course, you can also use the mouse to do this construction using the corresponding modes (see Modes) in the toolbar.

```
A = (-2, 1)
B = (5, 0)
C = (0, 5)
M_a = Midpoint[B, C]
M_b = Midpoint[A, C]
s_a = Line[A, M_a]
s_b = Line[B, M_b]
S = Intersect[s_a, s_b]
```

Alternatively you can compute the centroid directly as `S1 = (A + B + C) / 3` and compare both results using the command `Relation[S, S1]`.

Subsequently we can experiment whether `S = S1` is true for other positions of *A*, *B*, and *C* as well. We do this by selecting mode *Move* with the mouse and dragging the points.

2.4. Split Line Segment AB at Ratio 7:3

As GeoGebra allows us to calculate with vectors, this is an easy task. Type the following lines into the input field and press the *Enter* key after each line.

```
A = (-2, 1)
B = (3, 3)
s = Segment[A, B]
T = A + 7/10 (B - A)
```

Another way of doing this could be

```
A = (-2, 1)
B = (3, 3)
s = Segment[A, B]
v = Vector[A, B]
T = A + 7/10 v
```

In a next step we could introduce a number *t*, e.g. by using a *Slider* and redefine point *T* as `T = A + t v` (see Redefine). By changing *t* you can see point *T* moving along a straight line which could now be entered in parametric form (see Line): `g: X = T + s v`

2.5. Set of Linear Equations in Two Variables

Two linear equations in *x* and *y* can be interpreted as two straight lines. The algebraic solution is the intersection point of these two lines. Just type the following lines into the input field and press the *Enter* key after each line.

```
g: 3x + 4y = 12
h: y = 2x - 8
S = Intersect[g, h]
```

To change the equations you can right click (MacOS: *Apple* + click) one of them and select *Redefine*. Using the mouse you can drag the lines in *Move* or rotate them about a point using *Rotate around point*.

2.6. Tangent to a Function of x

GeoGebra offers a command for the tangent to a function *f(x)* at *x* = *a*. Type the following lines into the input field and press the *Enter* key after each line.

```
a = 3
f(x) = 2 sin(x)
t = Tangent[a, f]
```

By animating number *a* (see Animation) the tangent slides along the graph of function *f*.

Here is another way of getting the tangent to a function *f* in a certain point *T*.

```
a = 3
f(x) = 2 sin(x)
T = (a, f(a))
t: X = T + s (1, f'(a))
```

This gives us point *T* on the graph of *f* whereby tangent *t* is given in parametric form.

By the way, you can create the tangent of a function geometrically too:

- Select mode *New point* and click on the graph of function *f* to get a new point *A* that lies on function *f*.
- Select mode *Tangents* and click successively on function *f* and on point *A*.

Now, select *Move* and drag point *A* along the function with your mouse. In this way you can observe the tangent changing dynamically too.

2.7. Investigation of Polynomial Functions

With GeoGebra you can investigate roots, local extrema, and inflection points of polynomial functions. Type the following lines into the input field and press the *Enter* key after each line.

```
f(x) = x^3 - 3 x^2 + 1
R = Root[f]
E = Extremum[f]
I = InflectionPoint[f]
```

In *Move* you can drag function *f* with the mouse now. In this context, the first two derivatives of *f* could be interesting too. You get them by typing the following command into the input field and hitting the *Enter* key after each line.

```
Derivative[f]
Derivative[f, 2]
```

2.8. Integrals

To introduce integrals, GeoGebra offers the possibility to visualize lower and upper sums of a function as rectangles. Type the following lines into the input field and press the *Enter* key after each line.

```
f(x) = x^2/4 + 2
a = 0
b = 2
n = 5
L = LowerSum[f, a, b, n]
U = UpperSum[f, a, b, n]
```

By modifying *a*, *b*, or *n* (see Animation; see mode *Slider*) you can see the impact of these parameters on the upper and lower sum. In order to change the increment of number *n* to 1 you can right click (MacOS: *Apple* + click) on number *n* and select *Properties*.

The definite integral can be shown using the command `Integral[f, a, b]`, while the antiderivative *F* is created using `F = Integral[f]`.

3. Geometric Input

In this chapter we will explain how to use the mouse to create and modify objects in GeoGebra.

3.1. General Notes

The geometry window (to the right) shows the graphical representation of points, vectors, segments, polygons, functions, straight lines, and conic sections. Whenever the mouse is moved over one of these objects a description appears and the object is highlighted.
Note: Sometimes, the geometry window will be called *drawing pad*.

There are several modes to tell GeoGebra how it should react to mouse input in the geometry window (see Modes). For example, clicking on the drawing pad can create a new point (see mode New point), intersect objects (see mode Intersect two objects), or create a circle (see Circle modes).

Note: Double clicking an object in the algebra window opens its editing field.

3.1.1. Context Menu

Right clicking on an object yields a context menu where you can, for example, choose its algebraic notation (polar or Cartesian coordinates, implicit or explicit equation,…). Here you also find commands like *Rename*, *Redefine* or *Delete* .

Choosing *Properties* in the context menu shows a dialog window, where you can change for example color, size, line thickness, line style, and filling of objects.

3.1.2. Show and Hide

Geometric objects can be drawn (show) or not (hide). Use mode *Show / hide object* or the Context menu to change this state. The icon to the left of every object in the algebra window tells us about its current visibility state ("shown" or "hidden").

Note: You can also use the mode *Check box to show and hide objects* in order to show / hide one or several objects.

3.1.3. Trace

Geometric objects can leave a trace on the screen when they are moved. Use the Context menu to switch this trace on or off.

Note: The menu item *Refresh Views* in the *View* menu clears all traces.

3.1.4. Zoom

After right clicking (MacOS: *Apple* + click) on the drawing pad a context menu appears which allows you to zoom in (see also mode Zoom in) or zoom out (see also mode Zoom out).

Note: To specify a zoom window right click (MacOS: *Apple* + click) on the drawing pad and drag the mouse.

3.1.5. Axes Ratio

Right click (MacOS: *Apple* + click) on the drawing pad and select *Properties* to get a context menu where you can

- change the ratio between *x*-axis and *y*-axis
- hide / show the coordinate axes individually
- modify the layout of the axes (e.g. ticks, color, line style)

3.1.6. Construction Protocol

The interactive construction protocol (menu *View*, *Construction Protocol*) is a table that shows all construction steps. It allows you to redo a prepared construction step by step using the navigation bar at the bottom of the window. It is even possible to insert construction steps and change their sequence. Please, find more details in the construction protocol's help menu.

Note: Using the column *Breakpoint* in the *View* menu you are able to define certain construction steps as breakpoints allowing you to group objects. When navigating through your construction by means of the navigation bar groups of objects are shown at the same time.

3.1.7. Navigation Bar

GeoGebra offers a navigation bar in order to navigate through the construction steps of a prepared construction. Select *Navigation bar for construction steps* in the *View* menu in order to display the navigation bar at the bottom of the geometry window.

3.1.8. Redefine

An object may be *redefined* using its Context menu. This is very useful for subsequent changes of your construction. You can also open the *Redefine* dialog by selecting *Move* and double clicking on a dependant object in the algebra window.

Examples:
In order to place a free point *A* on a line *h*, choose *Redefine* for point *A* and enter `Point[h]` in the input field of the appearing dialog. To remove the point from this line and make it free again, redefine it to some free coordinates.

Another example is the conversion of a line *h* through two points *A* and *B* into a segment. Choose *Redefine* and enter `Segment[A, B]` in the input field of the appearing dialog. This also works the other way round.

Redefining objects is a very versatile tool to change a construction. Please note that this may also change the order of the construction steps in the Construction protocol.

3.1.9. Properties Dialog

The properties dialog allows you to modify properties of objects (e.g., color, line style). You can open this dialog either by right clicking (MacOS: *Apple* + click) an object and selecting *Properties*, or by selecting *Properties* in the *Edit* menu.

In the properties dialog objects are organized by types (e.g., points, lines, circles) which makes it easy to handle large numbers of objects. You can modify the properties of selected objects within the tabs on the right side. Close the properties dialog when you are done with changing properties of objects.

3.2. Modes

The following modes can be activated in the toolbar or the *Geometry menu*. Click on the small arrow in the lower right corner of an icon to get a menu with other modes.
Note: In all construction modes you can easily create new points by clicking on the drawing pad.

Marking an Object

To mark an object means to click on it with the mouse.

Fast Renaming of Objects

To rename a selected or newly created object just start typing to open the *Rename* dialog for this object.

3.2.1. General Modes

Move

In this mode you can drag and drop free objects with the mouse. If you select an object by clicking on it in *Move* mode, you may

- delete it by pressing the *Del* key
- move it by using the arrow keys (see Animation)

Note: Pressing the *Esc* key activates the *Move* mode too.

By holding the *Ctrl* key you can select several objects at the same time.

Another way of selecting multiple objects is by pressing and holding the left mouse key in order to specify a selection rectangle. You may then move the selected objects by dragging one of them with the mouse.

The selection rectangle can also be used to specify a part of the graphics window for printing, exporting pictures, and for dynamic worksheets (see Printing and Export).

Rotate around point

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

a ≟ b **Relation**

Mark two objects to get information about their relation (also see command Relation).

Move drawing pad

Drag and drop the drawing pad to move the origin of the coordinate system.
Note: You can also move the drawing pad by pressing the *Shift* key (PC: also *Ctrl* key) and dragging it with the mouse.

In this mode you can also scale each of the axes by dragging it with the mouse.
Note: Scaling the axes is also possible in every other mode by pressing and holding the *Shift* key (PC: also *Ctrl* key) while dragging the axis.

Zoom in

Click on any place of the drawing pad to zoom in (also see Zoom)

Zoom out

Click on any place of the drawing pad to zoom out (also see Zoom)

Show / hide object

Click on an object to show or hide it.
Note: All objects that should be hidden are highlighted. Your changes will be applied as soon as you switch to any other mode in the toolbar.

Show / hide label

Click on an object to show or hide its label.

Copy visual style

This mode lets you copy visual properties (e.g. color, size, line style) from one object to several others. To do so, first choose the object whose properties you want to copy. Afterwards click on all other objects that should adopt these properties.

Delete object

Click on any object you want to delete.

3.2.2. Point

New point

Clicking on the drawing pad creates a new point.
Note: The coordinates of the point are fixed when the mouse button is released.

By clicking on a segment, straight line, polygon, conic section, function, or curve you create a point on this object (also see command Point). Clicking on the intersection of two objects creates this intersection point (also see command Intersect).

Intersect two objects

Intersection points of two objects can be produced in two ways. If you…

- mark two objects *all intersection points* are created (if possible).
- click on an intersection of the two objects only this *single intersection point* is created.

For segments, rays, or arcs you may specify whether you want to *allow outlying intersection points* (see 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.

Midpoint or center

Click on ...

- two points to get their midpoint.
- one segment to get its midpoint.
- a conic section to get its center.

3.2.3. Vector

Vector between two points

Mark starting point and end point of the vector.

Vector from point

Mark a point *A* and a vector *v* to create point *B* = *A* + *v* and the vector from *A* to *B*.

3.2.4. Segment

Segment between two points

Marking two points *A* and *B* fixes a segment between *A* and *B*. In the algebra window the segment's length is displayed.

Segment with given length from point

Click on a point *A* that should be the starting point of the segment. Specify the desired length *a* of the vector in the appearing window.

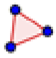
Note: This mode creates a segment with length *a* and endpoint *B* which may be rotated in mode *Move* around the starting point *A*.

3.2.5. Ray

Ray through two points

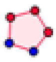
Marking two points *A* and *B* creates a ray starting at *A* through *B*. In the algebra window you see the equation of the corresponding line.

3.2.6. Polygon



Polygon

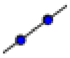
Mark at least three points which will be the vertices of the polygon. Then, click the first point again in order to close the polygon. In the algebra window you see the polygon's area.



Regular polygon

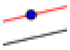
Marking two points *A* and *B* and typing a number *n* into the text field of the appearing dialog gives you a regular polygon with *n* vertices (including points *A* and *B*).

3.2.7. Line



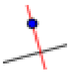
Line through two points

Marking two points *A* and *B* fixes a straight line through *A* and *B*. The line's direction vector is (*B* - *A*).



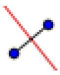
Parallel line

Marking a line *g* and a point *A* defines a straight line through *A* parallel to *g*. The line's direction is the direction of line *g*.



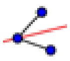
Perpendicular line

Marking a line *g* and a point *A* yields a straight line through *A* perpendicular to line *g*. The line's direction is equivalent to the perpendicular vector (also see command PerpendicularVector) of g.



Line bisector

The line bisector of a line segment is stated by a segment *s* or two points *A* and *B*. The line's direction is equivalent to the perpendicular vector (also see command PerpendicularVector) of segment *s* or *AB*.



Angular bisector

Angular bisectors can be defined in two ways.

- Marking three points *A*, *B*, *C* produces the angular bisector of the enclosed angle, where *B* is the apex.
- Marking two lines produces their two angular bisectors.

Note: The direction vectors of all angular bisectors have length 1.

Tangents

Tangents to a conic can be produced in two ways.

- Marking a point *A* and a conic *c* produces all tangents through *A* to *c*.
- Marking a line *g* and a conic *c* produces all tangents to *c* that are parallel to *g*.

Marking a point *A* and a function *f* produces the tangent line to *f* in $x = x(A)$.

Polar or diameter line

This mode creates the polar or diameter line of a conic section. You can either

- Mark a point and a conic section to get the polar line.
- Mark a line or a vector and a conic section to get the diameter line.

3.2.8. Conic Section

Circle with center through point

Marking a point *M* and a point *P* defines a circle with center *M* through *P*. This circle's radius is the distance *MP*.

Circle with center and radius

After marking a center point *M* you are asked to enter the radius in the text field of the appearing window.

Circle through three points

Marking three points *A*, *B*, and *C* defines a circle through these points. If the three points lie on one straight line, the circle degenerates to this line.

Conic through five points

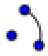
Marking five points produces a conic section through them.
Note: If no four of these five points lie on a line, the conic section is defined.

3.2.9. Arc and Sector

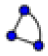
Note: The algebraic value of an arc is its length. The value of a sector is its area.

Semicircle

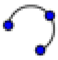
Marking two points *A* and *B* produces a semicircle above the segment *AB*.

 **Circular arc with center through two points**

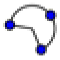
Marking three points *M*, *A*, and *B* produces a circular arc with center *M*, starting point *A* and endpoint *B*.
Note: Point *B* does not have to lie on the arc.

 **Circular sector with center through two points**

Marking three points *M*, *A*, and *B* produces a circular sector with center *M*, starting point *A* and endpoint *B*.
Note: Point *B* does not have to lie on the sector.

 **Circumcircular arc through three points**

Marking three points produces a circular arc through these points.

 **Circumcircular sector through three points**

Marking three points produces a circular sector through these points.

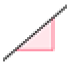
3.2.10. Number and Angle

Distance or length

This mode yields the distance of two points, of two lines, or a point and a line. It can also give you the length of a segment or the circumference of a circle.

Area

This mode gives you the area of a polygon, circle, or ellipse as a dynamic text in the geometry window.

 **Slope**

This mode gives you the slope of a line as a dynamic text in the geometry window.

Slider

Note: In GeoGebra a slider is nothing but the graphical representation of a free number or angle.

Click on any free place on the drawing pad to create a slider for a number or an angle. The appearing window allows you to specify the name, interval *[min, max]* of the number or angle, as well as the alignment and width of the slider (in pixel).

Note: You can easily create a slider for any existing free number or angle by showing this object (see Context menu; see mode Show / hide object).

The position of a slider may be absolute on the screen or relative to the coordinate system (see Properties of the corresponding number or angle).

Angle

This mode creates …

- angle between three points
- angle between two segments
- angle between two lines
- angle between two vectors
- all interior angles of a polygon

If you want to limit the maximum size of an angle to 180°, uncheck *allow reflex angles* in the *Properties dialog*.

Angle with given size

Mark two points *A* and *B* and type the angle's size into the text field of the appearing window. This mode produces a point *C* and an angle α, where α is the angle ABC.

3.2.11. Boolean

Check box to show and hide objects

Clicking on the drawing pad creates a check box (Boolean variable) in order to show and hide one or several objects. In the appearing window you can specify which objects should be affected by the check box.

3.2.12. Locus

Locus

Mark a point *B* that depends on another point *A* and whose locus should be drawn. Then, click on point *A*.
Note: Point *A* has to be a point on an object (e.g. line, segment, circle).

Example:

- Type `f(x) = x^2 – 2 x – 1` into the input field.
- Place a new point *A* on the *x*-axis (see mode New point; see command Point).
- Create point `B = (x(A), f'(x(A)))` that depends on point *A*.
- Select mode *Locus* and successively click on point *B* and point *A*.

- Drag point *A* along the *x*-axis to see point *B* moving along its locus line.

3.2.13. Geometric Transformations

The following geometric transformations work for points, lines, conic sections, polygons, and images.

Mirror object at point

At first, mark the object to be mirrored. Afterwards, click on the point that should be the mirror.

Mirror object at line

At first, mark the object to be mirrored. Afterwards, click on the line that should be the mirror.

Rotate object around point by angle

At first, mark the object to be rotated. Then, click on the point that should be the rotation center. Afterwards a window appears where you may specify the rotation angle.

Translate object by vector

At first, mark the object to be translated. Afterwards, click on the translation vector.

Dilate object from point

At first, mark the object to be dilated. Then, click on the point that should be the dilation center. Afterwards, a window appears where you may specify the dilation factor.

3.2.14. Text

ABC **Text**

With this mode you can create static and dynamic texts or LaTeX formulas within the geometry window.

- Clicking on the drawing pad creates a new text field at this location.
- Clicking on a point creates a new text field whose position is relative to this point.

Afterwards, a dialog appears where you may enter the text.

Note: It is also possible to use values of objects in order to create dynamic texts.

Input	Description
`"This is a text"`	simple text (static)
`"Point A = " + A`	dynamic text using the value of point *A*
`"a = " + a + "cm"`	dynamic text using the value of segment *a*

The position of a text may be absolute on the screen or relative to the coordinate system (see Properties of the text).

LaTeX Formulas

In GeoGebra you can also write formulas. To do so, check the box *LaTeX formula* in the dialog of the Text mode and enter your formula in LaTeX syntax.
Here, some important LaTeX commands are explained. Please have a look at any LaTeX documentation for further information.

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

3.2.15. Images

Insert image

This mode allows you to add an image to your construction.

- Clicking on the drawing pad specifies the lower left corner of the image.
- Clicking on a point specifies this point as the lower left corner of the image.

Afterwards, a file-open dialog appears where you may choose the image file to insert.

3.2.16. Properties of Images

Position

The position of an image may be absolute on screen or relative to the coordinate system (see Properties of the image). The latter is achieved by specifying up to three corner points. This gives you the flexibility to scale, rotate, and even distort images.

- 1. corner (position of the lower left corner of the image)
- 2. corner (position of the lower right corner of the image)
  Note: This corner may only be set if the 1. corner was set before. It controls the width of the image.
- 4. corner (position of the upper left corner of the image)
  Note: This corner may only be set if the 1. corner was set before. It controls the height of the image.

Note: Also see command Corner

Examples:
Let's create three points *A*, *B*, and *C* 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 explore their influence very easily.
- Set point *A* as the first and point *C* as the fourth corner and explore how dragging the points now influences the image.
- Finally, you could set all three corner points and see how dragging the points distorts your image.

You already saw how to influence the position and size of your image. If you want 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:

- 1. corner: *A*
- 2. corner: *A* + (3, 0)
- 4. corner: *A* + (0, 4)

Note: If you now drag point *A* in *Move* mode, your image keeps the desired size.

Background Image

You may set an image to be in the *background* (see Properties of the image). 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, choose *Properties* from the *Edit* menu.

Transparency

An image can be made transparent in order to see objects or axes lying behind it. You can set the transparency of an image by specifying a *filling* value between 0 % and 100 % (see Properties of the image).

4. Algebraic Input

In this chapter we will explain how to use the keyboard to create and modify objects in GeoGebra.

4.1. General Notes

Values, coordinates and equations of *free* and *dependant* objects are shown in the algebra window (on the left hand side). Free objects do not depend on any other objects and can be changed directly.

You can create and modify objects using the input field at the bottom of the GeoGebra screen (see Direct Input; see Commands).
Note: Always press the *Enter* key after typing the definition of an object into the input line.

4.1.1. Change Values

Free objects can be changed directly whereas dependant objects can't. To manipulate a free object's value overwrite it by entering the new value in the input field (see Direct Input).
Example: If you want to change the value of an existing number a = 3, type `a = 5` into the input field and press the *Enter* key.

Note: Alternatively, this can be done in the algebra window by choosing *Redefine* in the Context Menu or by double clicking an object in Move mode in the algebra window.

4.1.2. Animation

To change a number or angle continuously select Move mode. Then, click on the number or angle and press the + or – key.

Keeping one of these keys pressed allows you to produce animations.
Example: If a point's coordinates depend on a number k like in $P = (2\ k,\ k)$, the point will move along a straight line when k is changed continuously.

With the arrow keys you can move any free object in *Move* mode (see Animation; see mode Move).
Note: You can adjust the increment using the Properties dialog of this object.

Shortcuts:

- *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 its line using the + or – key (see Animation).

4.2. Direct Input

GeoGebra can treat numbers, angels, points, vectors, segments, lines, conic sections, functions, and parametric curves. We will now explain how these objects can be entered through coordinates or equations into the input field.

Note: You can also use indices within the names of objects, for example A_1 or S_{AB} is entered as `A_1` or `s_{AB}`.

4.2.1. Numbers and Angles

Numbers and angles use the sign "." as decimal point.

Example: You get a number *r* by typing in `r = 5.32`.
Note: You can also use the constant π and the Euler constant e for expressions and calculations by selecting them in the drop down menu next to the input field.

Angles are entered in *degree* (°) or *radian* (rad). The constant π is useful for radian values and can also be entered as `pi`.

Example: An angle α can be entered in degree (`α = 60`) or in radian (`α = pi/3`).

Note: GeoGebra does all internal calculations in radian. The symbol ° is nothing but a constant for π/180 to convert degree into radian.

Sliders and Arrow Keys

Free numbers and angles can be displayed as sliders in the geometry window (see mode Slider). By means of the arrow keys you may change numbers and angles in the algebra window too (see Animation).

Limit Value to Interval

Free numbers and angles may be limited to an interval *[min, max]* (see Properties dialog). This interval is also used for Sliders.

For every dependant angle you can specify whether it may become reflex or not (see Properties dialog).

4.2.2. Points and Vectors

Points and vectors may be entered in *Cartesian* or *polar coordinates* (see Numbers and Angles).
Note: Upper case labels denote points whereas lower case labels refer to vectors.

Examples:
- To enter a point *P* or a vector *v* in Cartesian coordinates use `P = (1, 0)` or `v = (0, 5)`.
- In order to use polar coordinates type in `P = (1; 0°)` or `v = (5; 90°)`.

4.2.3. Line

A line is entered as a linear equation in *x* and *y* or in parametric form. In both cases previously defined variables (e.g. numbers, points, vectors) can be used.
Note: You can enter a line's name at the beginning of the input followed by a colon.

Examples:
- Type in `g : 3x + 4y = 2` to enter line *g* as a linear equation.
- Define a parameter t (`t = 3`) before entering line *g* in parametric form using `g: X = (-5, 5) + t (4, -3)`.
- At first, define the parameters `m = 2` and `b = -1`. Then, you can enter the equation `g: y = m x + b` to get a line g in *y*-intercept-form.

xAxis and yAxis

The two coordinate axes are available in commands using the names *xAxis* and *yAxis*.
Example: The command `Perpendicular[A, xAxis]` constructs the perpendicular line to the *x*-axis through a given point *A*.

4.2.4. Conic Section

A conic section is entered as a quadratic equation in *x* and *y*. Prior defined variables (e.g. numbers, points, vectors) can be used. The conic section's name can be entered at the beginning of the input followed by a colon.

Examples:
- 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 *k1*: `k1: x^2 + y^2 = 25`
- Circle *k2*: `k2: (x–5)^2 + (y+2)^2 = 25`

Note: If you define two parameters `a = 4` and `b = 3` in advance, you can enter an ellipse as `ell: b^2 x^2 + a^2 y^2 = a^2 b^2`.

4.2.5. Function of x

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

- Function *f*: `f(x) = 3 x^3 – x^2`
- Function *g*: `g(x) = tan(f(x))`
- Nameless function: `sin(3 x) + tan(x)`

All internal functions (e.g. *sin*, *cos*, *tan*) are described in the section about arithmetic operations (see Arithmetic Operations).

In GeoGebra you can also use commands to get the Integral and Derivative of a function.

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: At first, define function *f* as `f(x) = 3 x^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 command Translate) and a free function can be moved with the mouse (see mode Move).

Limit Function to Interval

In order to limit a function to an interval [*a*, *b*], please use the command `Function` (see command Function).

4.2.6. Lists of Objects

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

Examples:

- `L = {A, B, C}` gives you a list that consists 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, as well as these nameless points.

4.2.7. Arithmetic Operations

To enter numbers, coordinates, or equations (see Direct Input) you may use arithmetic expressions with parentheses. The following operations are available in GeoGebra:

Operation	**Input**
addition	+
subtraction	–

Operation	Input
multiplication	`*` or *space* key
scalar product	`*` or *space* key
division	`/`
exponentiation	`^` or `2`
factorial	`!`
Gamma function	`gamma( )`
parentheses	`( )`
x-coordinate	`x( )`
y-coordinate	`y( )`
absolute value	`abs( )`
sign	`sgn( )`
square root	`sqrt( )`
cubic root	`cbrt( )`
random number between 0 and 1	`random( )`
exponential function	`exp( )` or e^x
logarithm (natural, of *e*)	`ln( )` or `log( )`
logarithm of 2	`ld( )`
logarithm of 10	`lg( )`
cosine	`cos( )`
sine	`sin( )`
tangent	`tan( )`
arc cosine	`acos( )`
arc sine	`asin( )`
arc tangent	`atan( )`
hyperbolic cosine	`cosh( )`
hyperbolic sine	`sinh( )`
hyperbolic tangent	`tanh( )`
antihyperbolic cosine	`acosh( )`
antihyperbolic sine	`asinh( )`
antihyperbolic tangent	`atanh( )`
greatest integer less than or equal	`floor( )`
least integer greater than or equal	`ceil( )`
round	`round( )`

Examples:

- Midpoint *M* of two points *A* and *B* might be entered as `M = (A + B) / 2`
- The length of a vector *v* might be calculated using `l = sqrt(v * v)`

Note: In GeoGebra you can do calculations with points and vectors as well.

4.2.8. Boolean Variables

You can use the Boolean variables “true” and “false” in GeoGebra.

Example: Type `a = true` or `b = false` into the input field and press the *Enter* key.

Check Box and Arrow Keys

Free Boolean variables can be displayed as check boxes on the drawing pad (see mode Check box to show and hide objects). By means of the arrow keys you may change Boolean variables in the algebra window too (see Animation).

4.2.9. Boolean Operations

You can use the following Boolean operations in GeoGebra:

	Operation	Example	Types
equal	≟ or ==	`a ≟ b` or `a == b`	numbers, points, lines, conics *a*, *b*
unequal	≠ or !=	`a ≠ b` or `a != b`	numbers, points, lines, conics *a*, *b*
less than	<	`a < b`	numbers *a*, *b*
greater than	>	`a > b`	numbers *a*, *b*
less or equal than	≤ or <=	`a ≤ b` or `a <= b`	numbers *a*, *b*
greater or equal than	≥ or >=	`a ≥ b` or `a >= b`	numbers *a*, *b*
and	∧	`a ∧ b`	Booleans *a*, *b*
or	∨	`a ∨ b`	Booleans *a*, *b*
not	¬ or !	`¬a` or `!a`	Booleans *a*
parallel	‖	`a ‖ b`	lines *a*, *b*
perpendicular	⊥	`a ⊥ b`	lines *a*, *b*

4.3. Commands

Using commands we can produce new and modify existing objects. A command's result may be named by entering a label followed by "=". 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 = Intersect[g, h]` (see command Intersect).

Note: You can also use indices within the names of objects: A_1 or S_{AB} is entered as `A_1` or `s_{AB}`.

4.3.1. General Commands

Relation

`Relation[object a, object b]`: shows a message box that tells us the relation of object *a* and object *b*.
Note: This command lets us find out whether two objects are equal, if a point lies on a line or conic, or if a line is tangent or a passing line to a conic.

Delete

`Delete[object a]`: Deletes an object *a* and all its dependants.

Element

`Element[list L, number n]`: n^{th} element of a list *L*

4.3.2. Boolean Commands

`If[condition, a, b]`: gives a copy of object *a* if *condition* evaluates to `true`, and a copy of object *b* if it evaluates to `false`.
`If[condition, a]`: gives a copy of object *a* if *condition* evaluates to `true`, and an undefined object if it evaluates to `false`.

4.3.3. Number

Length

`Length[vector v]`: Length of a vector v
`Length[point A]`: Length of the position vector of *A*
`Length[function f, number x1, number x2]`: Length of the function graph of *f* between numbers *x1* and *x2*
`Length[function f, point A, point B]`: Length of the function graph of *f* between two points *A* and *B* on the graph
`Length[curve c, number t1, number t2]`: Length of curve *c* between numbers *t1* and *t2*
`Length[curve c, point A, point B]`: Length of curve *c* between two points *A* and *B* on the curve
`Length[list L]`: Length of list *L* (number of elements in the list)

Area

`Area[point A, point B, point C, ...]`: Area of the polygon defined by the given points *A*, *B*, and *C*
`Area[conic c]`: Area of a conic section *c* (circle or ellipse)

Distance

`Distance[point A, point B]`: Distance of two points *A* and *B*
`Distance[point A, line g]`: Distance of point *A* and line *g*
`Distance[line g, line h]`: Distance of lines *g* and *h*.
Note: The distance of intersecting lines is 0. This function is interesting for parallel lines.

Modulo Function

`Mod[number a, number b]`: Remainder when number *a* is divided by number *b*

Integer Division

`Div[number a, number b]`: Integer quotient when number *a* is divided by number *b*

Slope

`Slope[line g]`: Slope of a line *g*.
Note: This command also draws the slope triangle whose size may be changed (see Properties dialog).

Curvature

`Curvature[point A, function f]`: Curvature of function *f* in point *A*
`Curvature[point A, curve c]`: Curvature of curve *c* in point *A*

Radius

`Radius[circle c]`: Radius of a circle *c*

Circumference

`Circumference[conic c]`: Returns the circumference of a conic section *c* (circle or ellipse)

Perimeter

`Perimeter[polygon poly]`: Perimeter of a polygon *poly*

Parameter

`Parameter[parabola p]`: Parameter of a parabola *p* (distance of directrix and focus)

FirstAxisLength

`FirstAxisLength[conic c]`: Length of a conic section *c*'s principal axis

SecondAxisLength

`SecondAxisLength[conic c]`: Length of a conic section *c*'s second axis

Excentricity

`Excentricity[conic c]`: Excentricity of a conic section *c*

Integral

`Integral[function f, number a, number b]`: Definite integral of function *f(x)* from number *a* to *b*.
Note: This command also draws the area between the function graph of *f* and the *x*-axis.

`Integral[function f, function g, number a, number b]`: Definite integral of the difference of the functions *f(x) - g(x)* from number *a* to number *b*.
Note: This command also draws the area between the function graphs of *f* and *g*.

Note: See Indefinite integral

LowerSum

`LowerSum[function f, number a, number b, number n]`: Lower sum of function *f* on the interval [*a*, *b*] with *n* rectangles.
Note: This command draws the rectangles of the lower sum too.

UpperSum

`UpperSum[function f, number a, number b, number n]`: Upper sum of function *f* on the interval [*a*, *b*] with *n* rectangles.
Note: This command draws the rectangles of the upper sum too.

Iteration

`Iteration[function f, number x0, number n]`: Iterates function *f* *n* times using the given start value *x0.*
Example: After defining `f(x) = x^2` the command `Iteration[f, 3, 2]` gives you the result $(3^2)^2 = 81$

Minimum and Maximum

`Min[number a, number b]`: Minimum of the given numbers *a* and *b*
`Max[number a, number b]`: Maximum of the given numbers *a* and *b*

Affine Ratio

`AffineRatio[point A, point B, point C]`: Returns the affine ratio λ of three collinear points *A*, *B*, and *C*, where *C = A + λ * AB*

Cross Ratio

`CrossRatio[point A, point B, point C, point D]`: Cross ratio λ of four collinear points *A*, *B*, *C*, and *D*, where
λ = AffineRatio[B, C, D] / AffineRatio[A, C, D]

4.3.4. Angle

Angle

`Angle[vector v1, vector v2]`: Angle between two vectors *v1* and *v2* (between 0 and 360°)
`Angle[line g, line h]`: Angle between the direction vectors of two lines *g* and *h* (between 0 and 360°)
`Angle[point A, point B, point C]`: Angle enclosed by *BA* and *BC* (between 0 and 360°). Point *B* is the apex.
`Angle[point A, point B, angle alpha]`: Angle of size α drawn from point *A* with vertex *B*.
Note: Point *Rotate[A, alpha, B]* is created too.
`Angle[conic c]`: Angle of twist of a conic section *c*'s principle axis (see command Axes)
`Angle[vector v]`: Angle between *x*-axis and vector *v*
`Angle[point A]`: Angle between *x*-axis and position vector of point *A*
`Angle[number n]`: Converts a number *n* into an angle (result between 0 and 2pi)
`Angle[polygon poly]`: All inner angles of a polygon *poly*

4.3.5. Point

Point

`Point[line g]`: Point on line *g*
`Point[conic c]`: Point on conic section *c* (e.g. circle, ellipse, hyperbola)
`Point[function f]`: Point on function *f*
`Point[polygon poly]`: Point on polygon *poly*
`Point[vector v]`: Point on vector *v*
`Point[point P, vector v]`: Point *P* plus vector *v*

Midpoint and Center

`Midpoint[point A, point B]`: Midpoint of points *A* and *B*
`Midpoint[segment s]`: Midpoint of segment *s*
`Center[conic c]`: Center of a conic section *c* (e.g. circle, ellipse, hyperbola)

Focus

`Focus[conic c]`: (All) foci of a conic section *c*

Vertex

`Vertex[conic c]`: (All) vertices of a conic section *c*

Centroid

`Centroid[polygon poly]`: Centroid of a polygon *poly*

Intersect

`Intersect[line g, line h]`: Intersection point of lines *g* and *h*
`Intersect[line g, conic c]`: All Intersection points of line *g* and conic section *c* (max. 2)
`Intersect[line g, conic c, number n]`: n^{th} intersection point of line *g* and conic section *c*
`Intersect[conic c1, conic c2]`: All intersection points of conic sections *c1* and *c2* (max. 4)
`Intersect[conic c1, conic c2, number n]`: n^{th} intersection point of conic sections *c1* and *c2*
`Intersect[polynomial f1, polynomial f2]`: All intersection points of polynomials *f1* and *f2*
`Intersect[polynomial f1, polynomial f2, number n]`: n^{th} intersection point of polynomials *f1* and *f2*
`Intersect[polynomial f, line g]`: All intersection points of polynomial *f* and line *g*
`Intersect[polynomial f, line g, number n]`: n^{th} intersection point of polynomial *f* and line *g*
`Intersect[function f, function g, point A]`: Intersection point of functions *f* and *g* with initial point *A* (for Newton's method)
`Intersect[function f, line g, point A]`: Intersection point of function *f* and line *g* with initial point *A* (for Newton's method)
Note: Also see mode Intersect two objects

Root

`Root[polynomial f]`: All roots of polynomial *f* (as points)
`Root[function f, number a]`: One root of function *f* with initial value *a* (Newton's method)
`Root[function f, number a, number b]`: One root of function *f* on interval [*a*, *b*] (regula falsi)

Extremum

`Extremum[polynomial f]`: All local extrema of polynomial *f* (as points)

InflectionPoint

`InflectionPoint[polynomial f]`: All inflection points of polynomial *f*

4.3.6. Vector

Vector

`Vector[point A, point B]`: Vector from point *A* to point *B*
`Vector[point A]`: Position vector of point *A*

Direction

`Direction[line g]`: Direction vector of line *g*.
Note: A line with equation *ax* + *by* = *c* has the direction vector *(b, - a)*.

UnitVector

`UnitVector[line g]`: Direction vector with length 1 of a line *g*
`UnitVector[vector v]`: Vector with length 1, same direction and orientation as the given vector *v*

PerpendicularVector

`PerpendicularVector[line g]`: Perpendicular vector of line *g*.
Note: A line with equation *ax* + *by* = *c* has the perpendicular vector *(a, b)*.
`PerpendicularVector[vector v]`: Perpendicular vector of a vector *v*.
Note: A vector with coordinates *(a, b)* has the perpendicular vector *(- b, a)*.

UnitPerpendicularVector

`UnitPerpendicularVector[line g]`: Perpendicular vector with length 1 of a line *g*
`UnitPerpendicularVector[vector v]`: Perpendicular vector with length 1 of a vector *v*

Curvature Vector

`CurvatureVector[point A, function f]`: Curvature vector of function *f* in point *A*
`CurvatureVector[point A, curve c]`: Curvature vector of curve *c* in point *A*

4.3.7. Segment

Segment

`Segment[point A, point B]`: Segment between two points *A* and *B*
`Segment[point A, number a]`: Segment with length *a* and starting point *A*.
Note: The endpoint of the segment is created too.

4.3.8. Ray

Ray

`Ray[point A, point B]`: Ray starting at point *A* through point *B*
`Ray[point A, vector v]`: Ray starting at point *A* with direction vector *v*

4.3.9. Polygon

Polygon

`Polygon[point A, point B, point C,...]`: Polygon defined by the given points *A*, *B*, *C*,…
`Polygon[point A, point B, number n]`: Regular polygon with *n* vertices (including points *A* and *B*)

4.3.10. Line

Line

`Line[point A, point B]`: Line through two points *A* and *B*
`Line[point A, line g]`: Line through point *A* parallel to line *g*
`Line[point A, vector v]`: Line through point *A* with direction vector *v*

Perpendicular

`Perpendicular[point A, line g]`: Line through point *A* perpendicular to line *g*
`Perpendicular[point A, vector v]`: Line through point *A* perpendicular to vector *v*

LineBisector

`LineBisector[point A, point B]`: Line bisector of the line segment *AB*
`LineBisector[segment s]`: Line bisector of the segment *s*

AngularBisector

`AngularBisector[point A, point B, point C]`: Angular bisector of the angle defined by points *A*, *B*, and *C*.
Note: Point *B* is apex of this angle.
`AngularBisector[line g, line h]`: Both angular bisectors of lines *g* and *h*.

Tangent

`Tangent[point A, conic c]`: (All) tangents through point *A* to conic section *c*
`Tangent[line g, conic c]`: (All) tangents to conic section *c* that are parallel to line *g*
`Tangent[number a, function f]`: Tangent to function *f(x)* at *x = a*
`Tangent[point A, function f]`: Tangent to function *f(x)* at *x = x(A)*

`Tangent[point A, curve c]`: Tangent to curve *c* in point *A*

Asymptote

`Asymptote[hyperbola h]`: Both asymptotes of a hyperbola *h*

Directrix

`Directrix[parabola p]`: Directrix of a parabola *p*

Axes

`Axes[conic c]`: Principal and second axis of a conic section *c*

FirstAxis

`FirstAxis[conic c]`: Principal axis of a conic section *c*

SecondAxis

`SecondAxis[conic c]`: Second axis of a conic section *c*

Polar

`Polar[point A, conic c]`: Polar line of point *A* relative to conic section *c*

Diameter

`Diameter[line g , conic c]`: Diameter parallel to line *g* relative to conic section *c*
`Diameter[vector v, conic c]`: Diameter with direction vector *v* relative to conic section *c*

4.3.11. Conic Section

Circle

`Circle[point M, number r]`: Circle with midpoint *M* and radius *r*
`Circle[point M, segment s]`: Circle with midpoint *M* and radius equal to *Length[s]*
`Circle[point M, point A]`: Circle with midpoint *M* through point *A*
`Circle[point A, point B, point C]`: Circle through three points *A*, *B* and *C*

Osculating Circle

`OsculatingCircle[point A, function f]`: Osculating circle of function *f* in point *A*
`OsculatingCircle[point A, curve c]`: Osculating circle of curve *c* in point *A*

Ellipse

`Ellipse[point F, point G, number a]`: Ellipse with focal points *F* and *G* and principal axis length *a*.
Note: Condition: 2*a* > *Distance[F, G]*

`Ellipse[point F, point G, segment s]`: Ellipse with focal points *F* and *G* where the length of the principal axis equals the length of segment *s* (*a* = *Length[s]*).

Hyperbola

`Hyperbola[point F, point G, number a]`: Hyperbola with focal points *F* and *G* and principal axis length *a*.
Note: Condition: 0 < 2*a* < *Distance[F, G]*

`Hyperbola[point F, point G, segment s]`: Hyperbola with focal points *F* and *G* where the length of the principal axis equals the length of segment *s* (*a* = *Length[s]*)

Parabola

`Parabola[point F, line g]`: Parabola with focal point *F* and directrix *g*

Conic

`Conic[point A, point B, point C, point D, point E]`: Conic section through five points *A*, *B*, *C*, *D*, and *C*. Note: No four of the points lie on one line.

4.3.12. Function

Derivative

`Derivative[function f]`: Derivative of function *f(x)*
`Derivative[function f, number n]`: n^{th} derivative of function *f(x)*

Note: You can use `f'(x)` instead of `Derivative[f]`as well as `f''(x)` instead of `Derivative[f, 2]`.

Integral

`Integral[function f]`: Indefinite integral for function *f(x)*

Note: See Definite integral

Polynomial

`Polynomial[function f]`: Expanded polynomial function *f*.
Example: `Polynomial[(x - 3)^2]` yields $x^2 - 6x + 9$

TaylorPolynomial

`TaylorPolynomial[function f, number a, number n]`: Power series expansion for function *f* about the point *x* = *a* to order *n*

Function

`Function[function f, number a, number b]`: Function, that is equal to *f* on the interval [*a*, *b*] and not defined outside of [*a*, *b*]

Conditional Function

You can use the Boolean command `If` (see command If) in order to create a conditional function.
Note: You can use derivatives and integrals of such functions and intersect them like “normal” functions.

Example:
`f(x) = If[x < 3, sin(x), x^2]` gives you a function that equals

- sin(x) for x < 3 and
- x^2 for $x \geq 3$.

4.3.13. Parametric Curves

`Curve[expression e1, expression e2, parameter t, number a, number b]`: Cartesian parametric curve for the given *x*-expression *e1* and *y*-expression *e2* (using parameter *t*) within the given interval [*a*, *b*]
Example: `c = Curve[2 cos(t), 2 sin(t), t, 0, 2 pi]`

`Derivative[curve c]`: Derivative of the curve *c*

Note: Parametric curves can be used like functions in arithmetic expressions.
Example: Input `c(3)` returns the point at parameter position 3 on curve *c*.

Note: Using the mouse you can also place a point on a curve using the mode  *New point* (see mode New point; also see command Point). Since the parameters *a* and *b* are dynamic you could use slider variables there (see mode Slider).

4.3.14. Arc and Sector

Note: The algebraic value of an arc is its length and the value of a sector is its area.

Semicircle

`Semicircle[point A, point B]`: Semicircle above the segment AB.

CircularArc

`CircularArc[point M, point A, point B]`: Circular arc with midpoint *M* between points *A* and *B*.
Note: Point *B* does not have to lie on the arc.

CircumcircularArc

`CircumcircularArc[point A, point B, point C]`: Circular arc through three points *A*, *B*, and *C*

Arc

`Arc[conic c, point A, point B]`: Conic section arc between two points *A* and *B* on the conic section *c* (circle or ellipse)

`Arc[conic c, number t1, number t2]`: Conic section arc between two parameter values *t1* and *t2* on the conic section *c* for the following parameter forms:

- 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 first and second axis

CircularSector

`CircularSector[point M, point A, point B]`: Circular sector with midpoint *M* between two points *A* and *B*.
Note: point *B* does not have to lie on the arc.

CircumcircularSector

`CircumcircularSector[point A, point B, point C]`: Circular sector through three points *A*, *B*, and *C*

Sector

`Sector[conic c, point A, point B]`: Conic section sector between two points *A* and *B* on the conic section *c* (circle or ellipse)

`Sector[conic c, number t1, number t2]`: Conic section sector between two parameter values *t1* and *t2* on the conic section *c* for the following parameter forms:

- 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 first and second axis

4.3.15. Image

Corner

`Corner[image pic, number n]`: n^{th} corner of an image pic with a maximum of 4 corners

4.3.16. Text

Name

`Name[object]`: Text showing the name of the given object
Note: Use this command in dynamic texts for objects that might be renamed

4.3.17. Locus

Locus

`Locus[point Q, point P]`: Locus line of point *Q* that depends on point *P*.
Note: Point *P* has to be a point on an object (e.g. line, segment, circle).

4.3.18. Sequence

Sequence

`Sequence[expression e, variable i, number a, number b]`: List of objects created using expression *e* and index *i* that ranges from number *a* to number *b*.
Example: `L = Sequence[(2, i), i, 1, 5]` creates a list of points whose *y*-coordinates range from 1 to 5

`Sequence[expression e, variable i, number a, number b, number s]`: List of objects created using expression *e* and index *i* that ranges from number *a* to number *b* with given step size *s*.
Example: `L = Sequence[(2, i), i, 1, 5, 0.5]` creates a list of points whose *y*-coordinates range from 1 to 5 with a step size of 0.5.

Note: Since the parameters *a* and *b* are dynamic you could use slider variables there.

Other Sequence Commands

`Element[list L, number n]`: n^{th} element of a list *L*
`Length[list L]`: Length of a list *L*
`Min[list L]`: Minimal element of a list *L*
`Max[list L]`: Maximal element of a list *L*

Iteration

`IterationList[function f, number x0, number n]`:
List *L* of length *n*+1 whose elements are iterations of function *f* starting with the value *x0.*
Example: After defining function `f(x) = x^2` the command `L = IterationList[f, 3, 2]` gives you the list
$L = \{3, 3^2, (3^2)^2\} = \{3, 9, 81\}$

4.3.19. Geometric Transformations

If you assign one of the following commands to a new name, a copy of the moved object will be produced.
Note: The command `Mirror[A, g]` mirrors point *A* at line *g* and changes point *A*'s location. Entering `B = Mirror[A, g]` will produce a new point *B* while point *A* is left unchanged.

Translate

`Translate[point A, vector v]`: Translates point *A* by vector *v*
`Translate[line g, vector v]`: Translates line *g* by vector *v*
`Translate[conic c, vector v]`: Translates conic *c* by vector *v*
`Translate[function c, vector v]`: Translates function *f* by the vector *v*
`Translate[polygon poly, vector v]`: Translates polygon *poly* by vector *v*.
Note: New vertices and segments are created too.
`Translate[image pic, vector v]`: Translates image *pic* by vector *v*
`Translate[vector v, Point P]`: Translates vector *v* to point *P*
Note: Also see mode Translate object by vector

Rotate

`Rotate[point A, angle phi]`: Rotates point *A* by angle *φ* around the axis origin
`Rotate[vector v, angle phi]`: Rotates vector *v* by angle *φ*
`Rotate[line g, angle phi]`: Rotates line *g* by angle *φ* around the axis origin
`Rotate[conic c, angle phi]`: Rotates conic section *c* by angle *φ* around the axis origin
`Rotate[polygon poly, angle phi]`: Rotates polygon *poly* by angle *φ* around the axis origin.
Note: New vertices and segments are created too.
`Rotate[image pic, angle phi]`: Rotates image *pic* by angle *φ* around the axis origin
`Rotate[point A, angle phi, point B]`: Rotates point *A* by angle *φ* around point *B*
`Rotate[line g, angle phi, point B]`: Rotates line *g* by angle *φ* around point *B*
`Rotate[conic c, angle phi, point B]`: Rotates conic section *c* by angle *φ* around point *B*
`Rotate[polygon poly, angle phi, point B]`: Rotates polygon *poly* by angle *φ* around point *B*.
Note: New vertices and segments are created too.
`Rotate[image pic, angle phi, point B]`: Rotates image *pic* by angle *φ* around point *B*
Note: Also see mode Rotate object around point by angle

Mirror

`Mirror[point A, point B]`: Mirror point *A* at point *B*

`Mirror[line g, point B]`: Mirror line *g* at point *B*
`Mirror[conic c, point B]`: Mirror conic section *c* at point *B*
`Mirror[polygon poly, point B]`: Mirror polygon *poly* at point B.
Note: New vertices and segments are created too.
`Mirror[image pic, point B]`: Mirror image *pic* at point *B*
`Mirror[point A, line h]`: Mirror point *A* at line *h*
`Mirror[line g, line h]`: Mirror line *g* at line *h*
`Mirror[conic c, line h]`: Mirror conic *c* at line *h*
`Mirror[polygon poly, line h]`: Mirror polygon *poly* at line *h*.
Note: New vertices and segments are created too.
`Mirror[image pic, line h]`: Mirror image *pic* at line *h*
Note: Also see mode Mirror object at point; mode Mirror object at line

Dilate

`Dilate[point A, number f, point S]`: Dilates point *A* from point *S* using factor *f*
`Dilate[line h, number f, point S]`: Dilates line *h* from point *S* using factor *f*
`Dilate[conic c, number f, point S]`: Dilates conic section *c* from point *S* using factor *f*
`Dilate[polygon poly, number f, point S]`: Dilates polygon *poly* from point *S* using factor *f*.
Note: New vertices and segments are created too.
`Dilate[image pic, number f, point S]`: Dilates image *pic* from point *S* using factor *f*
Note: Also see mode Dilate object from point by vector

5. Printing and Export

5.1. Printing

5.1.1. Drawing Pad

You find the item *Print Preview* for the drawing pad in the *File* menu. Here, you can specify title, author, date and the scale of your print-out (in cm).

Note: Press Enter after any change to update the preview window.

5.1.2. Construction Protocol

In order to open the print preview window of the construction protocol you need to open the *Construction Protocol* first (menu *View*). There you find the item *Print Preview* in the *File* menu of the appearing window.

Note: You may switch on and off the different columns *Name*, *Definition*, *Command*, *Algebra*, and *Breakpoint* of the construction protocol here (see menu *View* of the construction protocol).

In the print preview window of the construction protocol you can enter title, author, and date before printing your construction.

There is a navigation bar at the bottom of the construction protocol window. It allows you to navigate step-by-step through your construction (see Navigation bar).
Note: Using the column *Breakpoint* (menu *View*) you are able to define certain construction steps as breakpoints allowing you to group objects. When navigating through your construction by means of the navigation bar groups of objects are shown at the same time.

5.1.3. Drawing Pad as Picture

You can find the item *Drawing Pad as Picture* in the *File* menu, *Export*. Here, you can specify the scale (in cm) and the resolution (in dpi) of the output file. The true size of the exported image is shown at the bottom of the window.
When exporting the drawing pad as 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 with Microsoft Word.
Note: Whenever you insert a PNG graphic file into a Word 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 like Corel Draw and professional text processing systems like LaTeX.

The resolution of an EPS graphic is always 72dpi. This value is only used to calculate the true size of an image in cm and has no effect on the image's quality.

Note: The transparency effect with filled polygons or conic sections is not possible with EPS.

SVG – Scaleable Vector Graphic

(see EPS format above)

EMF – Enhanced Meta Format

(see EPS format above)

PSTricks

for LaTeX

5.2. Drawing Pad to Clipboard

You can find the item *Drawing Pad to Clipboard* in the *File* menu, *Export*. This feature copies a screenshot of the drawing pad to your system's clipboard as a PNG (see PNG format) picture. This picture can be pasted into other programs (e.g. a Microsoft Word document).

Note: In Order to export your construction at a certain scale (in cm) please use the menu item *Drawing Pad as Picture* in the *File* menu, *Export* (see Drawing Pad as Picture).

5.3. Construction Protocol as Webpage

In order to open the window *Export Construction Protocol* you first need to open the Construction protocol from the *View* menu. There you find the item *Export as Webpage* in the *File* menu.

Note: You may switch on and off the different columns of the construction protocol before exporting it as a webpage (see menu *View* of the construction protocol).

In the export window of the construction protocol you can enter title, author, and date of the construction and choose whether or not you want to export a picture of the drawing pad and the algebra window along with the protocol.

Note: The exported HTML file can be viewed with any internet browser (e.g. Mozilla, Internet Explorer) and edited with many word processing systems (e.g. FrontPage, Word).

5.4. Dynamic Worksheet as Webpage

In the File menu, Export you find the item Dynamic Worksheet as Webpage (html).

At the top of the export window you can enter title, author, and date for your dynamic worksheet.

The tab *General* allows you to add some text above and below the dynamic construction (e.g. a description of the construction and some tasks). The construction itself may be included directly into a webpage or opened by clicking a button.

The tab Advanced allows you to change the functionality of the dynamic construction (e.g. reset icon, double click to open the application window) as well as to modify the user interface (e.g. show toolbar, modify height and width).
Note: Don't take too big values for the width and height of the dynamic construction in order to make it fully visible in the internet browser.

Several files are created when exporting a dynamic worksheet:

- html file (e.g. *circle.html*) – this file includes the worksheet itself
- GGB file (e.g. *circle_worksheet.ggb*) – this file includes your GeoGebra construction
- *geogebra.jar* (several files) – these files include GeoGebra and make your worksheet interactive

All files (e.g. *circle.htm*l, *circle_worksheet.ggb* and the *geogebra.jar* files) have to be in one folder (directory) to let the dynamic construction work. Of course, you may copy all files to another folder too.

Note: The exported HTML file (e.g. *circle.html*) can be viewed with any internet browser (e.g. Mozilla, Internet Explorer, Safari). In order to let the dynamic construction work, Java has to be installed on the computer. You can get Java from http://www.java.com without charge. If you want to use your worksheet in your school's computer network, ask your local network administrator to install Java on the computers.

Note: You can edit the dynamic worksheet's text with many word processing systems (e.g. FrontPage, Word) by opening the exported HTML file.

6. Options

Global options may be changed in the menu *options*. To change object settings, please use the Context menu.

6.1. Point Capturing

Determines if the *Point capturing* is on / off or if points are captured by the grid.

6.2. Angle Unit

Determines whether angles are displayed in degree (°) or radian (rad).
Note: Input is always possible in both ways (degree and radian).

6.3. Decimal Places

Allows you to adjust the number of decimal places from 0 to 5.

6.4. Continuity

GeoGebra allows you to turn on / off the continuity heuristic in the *Options* menu. The software uses a near-to-heuristic to keep moving intersection points (line-conic, conic-conic) close to their old positions and avoid jumping intersection points.

Note: By default this heuristic is turned off. For user defined tools (see User Defined Tools) continuity is always turned off too.

6.5. Point Style

Determines whether points are displayed as dots or crosses.

6.6. Right Angle Style

Determines whether right angles are displayed as a rectangle, a dot, or like all other angles.

6.7. Coordinates

Determines whether coordinates of points are displayed as *A = (x, y)* or *A(x | y)*.

6.8. Labeling

You can specify whether the label of a newly created object should be shown or not.
Note: The setting *Automatic* shows labels when the algebra window is open while creating new objects.

6.9. Font Size

Determines the font size for labels and text in points (pt).

6.10. Language

GeoGebra is multilingual. Here you can change the current language setting. This affects all inputs including command names and all outputs.

6.11. Drawing Pad

Opens a dialog where properties of the drawing pad (e.g. coordinate grid and axes, background color) can be set.

6.12. Save Settings

GeoGebra remembers your favorite settings (settings in the *Options* menu, current toolbar and drawing pad) if you select *Save settings* in the menu *Options*.

7. Tools and Toolbar

7.1. User Defined Tools

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 names for the toolbar icon and command.

<u>Example</u>: Square-Tool

- 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 in the drop down menu.
- 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. The *tool name* will appear in GeoGebra's toolbar, the *command name* can be used in GeoGebra's input field.
- You can also choose an image for the toolbar icon. GeoGebra resizes your image automatically to fit on a toolbar button.

<u>Note</u>: Your tool can be used both with the mouse and as a command in the input field. All tools are automatically saved in your "GGB" construction file.

Using the *Manage tools* dialog (menu *Tools*) 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 its tools into another construction.
<u>Note</u>: Opening a "GGT" file doesn't change your current construction, but opening a "GGB" file does.

7.2. Customizable Toolbar

You can customize the tools in GeoGebra's toolbar by selecting *Customize toolbar* in the menu *Tools*. This is especially useful for dynamic worksheets where you want to restrict the available tools in the toolbar.
<u>Note</u>: The current toolbar setting is saved with your construction in a "GGB" file.

8. JavaScript Interface

Note: GeoGebra's JavaScript interface is interesting for users who have some experience in HTML editing.

In order to enhance your dynamic worksheets and increase their interactivity, GeoGebra applets provide a JavaScript interface. For example, you could create a button to randomly generate new configurations of a dynamic construction.

Please, see the document GeoGebra Applets and JavaScript (http://www.geogebra.org in "Help") for examples and information about using JavaScript with GeoGebra applets.

Index

V

X

Y

Z