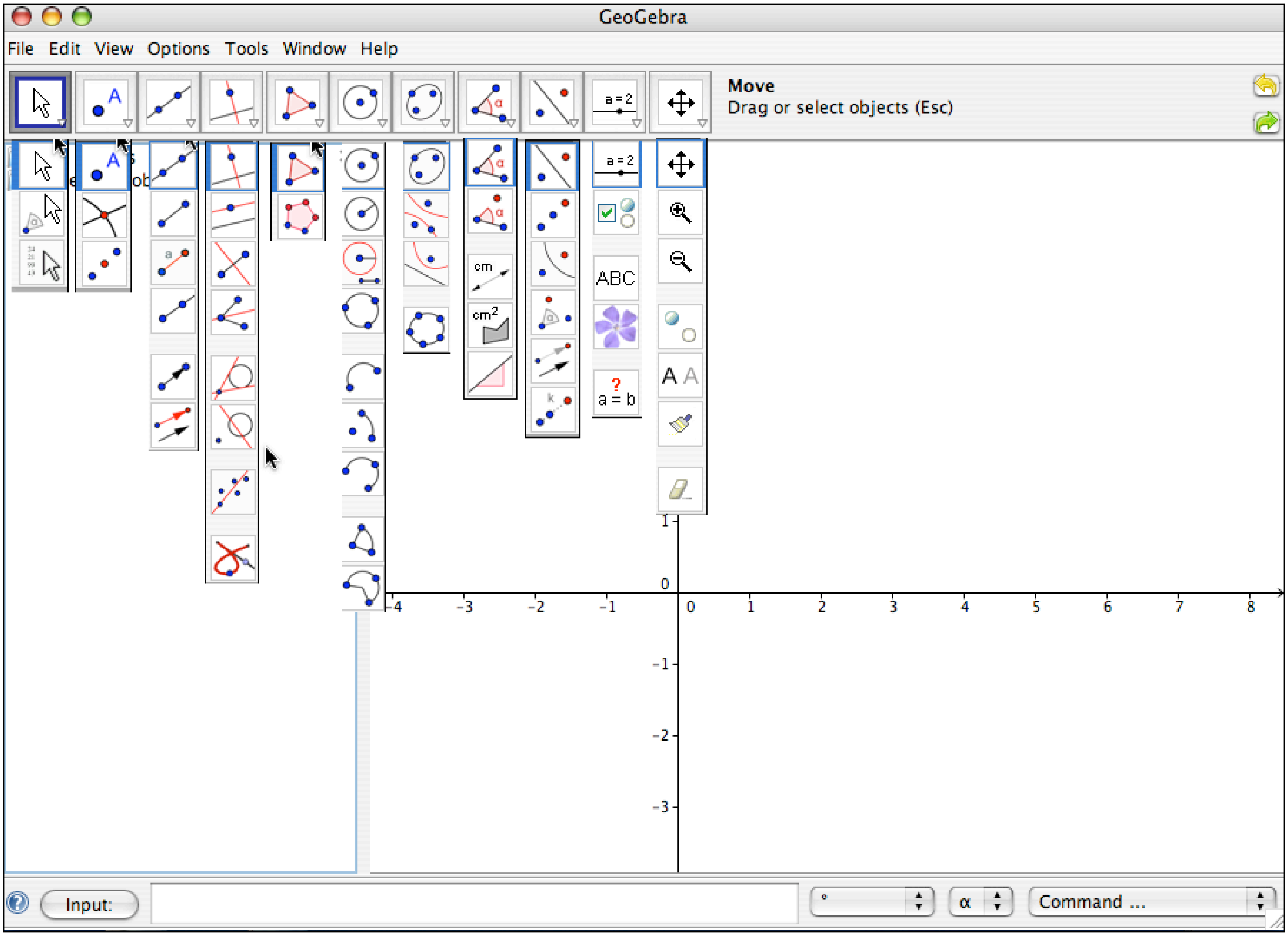
Changes with the next revease

Since GoeGebra is preparing to release a new version it is worthwhile to look at the new version, 3.2, that should come out shortly. The menus are slightly modified.



- **New Point**
- Intersect two objects
- Midpoint or center

- **Move**
- Rotate around point
- Record to spreadsheet

- **Polygon**
- Regular polygon

- **Perpendicular line**
- Parallel line
- Perpendicular bisector
- Angle bisector
- Tangents
- Polar or diameter line
- Best fit line
- Locus

- **Slider**
- Check box to show and hide objects
- Insert text
- Insert image
- Relation between two objects

- **Line through two points**
- Segment between two points
- Segment with given length from point
- Ray through two points
- Vector between two points
- Vector from point

- **Circle with center through point**
- Circle with center and radius
- Compass
- Circle through three points
- Semicircle through two points
- Circular arc with center through two points
- Circumcircular arc through three points
- Circular sector with center through two points
- Circumcircular sector through three points

- **Ellipse**
- Hyperbola
- Parabola
- Conic through five points

- **Mirror object at line**
- Mirror object at point
- Mirror point at circle
- Rotate object around point by angle
- Translate object by vector
- Dilate object from point by factor

- **Angle**
- Angle with given size
- Distance or length
- Area
- Slope

- **Move drawing pad**
- Zoom in
- Zoom out
- Show / hide object
- Show / hide label
- Copy visual style
- Delete object

New commands in the new version

Statistics

* Covariance[<list1 of numbers> , <list2 of numbers>]
Calculates the covariance using the elements of both lists
* Covariance[<list of points>]
Calculates the covariance using the x- and y-coordinates of the points
* InverseNormal[<mean>, <standard deviation>, <number>]
Calculates the function inversephi(x) * (standard deviation) + (mean)
where inversephi(x) is the inverse of the pdf for N(0,1)
(pdf = probability density function, ie a non-negative function with area 1)
* Mean[<list>]
Calculates the mean of the list elements
* MeanX[<list of points>]
Mean of the x coordinates of the points in the list
* MeanY[<list of points>]
Mean of the y coordinates of the points in the list
* Median[<list>]
Determines the median of the list elements
* Mode[<list>]
Determines the mode(s) of the list elements
Mode[{1,2,3,4}] returns {}
Mode[{1,1,1,2,3,4}] returns {1}
Mode[{1,1,2,2,3,3,4}] returns {1,2,3}
* Normal[<mean>, <standard deviation>, <number>]
Calculates the function (phi(x) - mean) / (standard deviation)
where phi(x) is the pdf for N(0,1)
(pdf = probability density function, ie a non-negative function with area 1)
* Q1[<list>]
Determines the lower quartile of the list elements
* Q3[<list>]
Determines the upper quartile of the list elements
* RandomBetween[<min>, <max>]
Generates a random integer between min and max (inclusive)
The numbers min and max need to be integers.
* RandomBinomial[<number n>, <number p>]
Generates a random number from a binomial distribution
* RandomNormal[<mean>, <standard deviation>]
Generates a random number from a normal distribution
* RandomPoisson[<mean>]
Generates a random number from a poisson distribution
* SD[<list>]
Calculates the standard deviation of list elements
* SigmaXX[<list of numbers>]
* SigmaXX[<list of points>]

Calculates the sum of squares (of list elements, or x coordinates of points)
  * SigmaXY[<list of x-coordinates> , <list of y-coordinates>]
  * SigmaXY[<list of points>]
Calculates the sum of (the product of the x and y coordinates).
For bivariate data, SigmaXY works out sum of (x coord times y coord)
  * SigmaYY[<list of points>]
Calculates the sum of squares of y coords
For bivariate data, SigmaYY = sum of (y coord ^2)
  * Sxx[<list of numbers> , <list of numbers>]
Calculates the statistic sigma(x^2) - sigma(x) * sigma(x)/n
  * Sxx[<list of points>]
Calculates the statistic sigma(x^2) - sigma(x) * sigma(x)/n
  * Sxy[<list of numbers> , <list of numbers>]
Calculates the statistic sigma(xy) - sigma(x) * sigma(y)/n
  * Sxy[<list of points>]
Calculates the statistic sigma(xy) - sigma(x) * sigma(y)/n
  * Syy[<list of numbers> , <list of numbers>]
Calculates the statistic sigma(y^2) - sigma(y) * sigma(y)/n
  * Syy[<list of points>]
Calculates the statistic sigma(y^2) - sigma(y) * sigma(y)/n
  * Take[<list> , <number m>, <number n>]
Returns a list containing the elements from positions m to n of the list.
  * Variance[<list>]
Calculates the variance of list elements

Working With Lists

  * Append[<list>, <object>]
Appends the object to the list
e.g. Append[{1, 2, 3}, (5, 5)] gives you {1, 2, 3, (5, 5)}
  * Append[<object>, <list>]
Appends the list to the object
e.g. Append[(5, 5), {1, 2, 3}] gives you {(5, 5), 1, 2, 3}
  * CountIf[<condition>, <list>]
Counts the number of elements in the list satisfying the condition
e.g. CountIf[x < 3, {1, 2, 3, 4, 5}]
e.g. CountIf[x<3, A1:A10] where A1:A10 is a range of cells in the spreadsheet
  * First[<list>, n]
Returns a list containing just the first n elements of the list.
  * Insert[<list 1>, <list 2>, <position>]
  * Intersection[<list 1>, <list 2>]
Gives you all elements that are part of both lists
  * Join[<list 1>, <list 2>, ...]
Joins the two (or more) lists (no re-ordering of elements, keeps all elements even if they are the same)
e.g. Join[{1,2,3}, {4,5,6}]
  * Join[<list of lists>]

Joins the sub-lists into one longer list (no re-ordering of elements, keeps all elements even if they are the same)
e.g. Join[{ {1,2,3}, {4,5,6}, {7,8,9} }]
* KeepIf[<condition>, <list>]
e.g. KeepIf[x<3, {1,2,3,4,1,5,6}] returns {1,2,1}
* Last[<list> , <number n>]
Returns a list containing just the last n elements of the list.
* Product[<list>]
Calculates the product of all list elements
* RemoveUndefined[<list>]
Removes undefined objects from a list
e.g. RemoveUndefined[Sequence[(-1)^i, i, -3, -1, 0.5]]
* Reverse[<list>]
Reverses the order of a list
* Sort[<list>]
Sorts a list of numbers, text objects or points (sorts points by x-coordinate)
e.g. Sort[{3, 2, 1}]
e.g. Sort[{"pears", "apples", "figs"}]
e.g. list1 = Sort[{A, B, C}] list2 = Sequence[Segment[Element[list1, i], Element[list1, i + 1]], i, 1, Length[list1] - 1]
* Sum[<list>]
Calculates the sum of all list elements
Works for numbers, points & vectors, text and functions
e.g. Sum[{1,2,3}] gives you a = 6
e.g. Sum[{x^2,x^3}] gives you f(x)=x^2 + x^3
e.g. Sum[Sequence[i,i,1,100]] gives you a = 5050
e.g. Sum[Sequence[1 / (2 k - 1) sin((2 k - 1) x), k, 1, 20]]
e.g. Sum[{(1, 2), (2, 3)}] gives you point A = (3, 5)
e.g. Sum[{(1, 2), 3}] gives you point B = (4, 2)
e.g. Sum[{"a","b","c"}] gives "abc"
* Sum[<list>, <number n>]
Calculates the sum of the first n list elements
Works for numbers, points & vectors, text and functions
e.g. Sum[{1, 2, 3, 4, 5, 6}, 4] gives you 10
* Union[<list1>, <list2>]
Joins lists and removes items that appear multiple times

Plotting Data

* BarChart[<start>, <end>, <list of heights>]
e.g. 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>, <end>, <expression>, <variable>, <from>, <to>]
* BarChart[<start>, <end>, <expression>, <variable>, <from>, <to>, <step>]
e.g. p = 0.1
q = 0.9

n = 10
BarChart[-0.5, n + 0.5, BinomialCoefficient[n,k]*p^k*q^(n-k), k, 0, n]
* BarChart[<raw data>, <width>]
e.g. BarChart[{1,1,1,2,2,2,2,2,3,3,3,5,5,5,5}, 1]
* BarChart[<data>, <frequencies>]
<data> must be a list where the numbers go up by a constant amount
e.g. BarChart[{10,11,12,13,14}, {5,8,12,0,1}]
e.g. BarChart[{5, 6, 7, 8, 9}, {1, 0, 12, 43, 3}]
e.g. BarChart[{0.3, 0.4, 0.5, 0.6}, {12, 33, 13, 4}]
* BarChart[<data>, <frequencies>, <width>]
<data> must be a list where the numbers go up by a constant amount
e.g. leaves gaps between bars: BarChart[{10,11,12,13,14}, {5,8,12,0,1}, 0.5]
e.g. line graph: BarChart[{10,11,12,13,14}, {5,8,12,0,1}, 0]
* BoxPlot[<yOffset>, <yScale>, <raw data>]
e.g. BoxPlot[0, 1, {2,2,3,4,5,5,6,7,7,8,8,8,9}]
* BoxPlot[<yOffset>, <yScale>, <start>, <Q1>, <median>, <Q3>, <end>]
e.g. BoxPlot[0, 1, 2, 3, 4, 5, 6]
* Histogram[<class boundaries>, <heights>]
e.g. Histogram[{1,2,4,8}, {3,5,7}]
* Histogram[<class boundaries>, <raw data>]
e.g. Histogram[{1,1.5,2,4},{1.0,1.1,1.1,1.2,1.7,1.7,1.8,2.2,2.5,4.0}]

Curve Fitting

* FitExp[<list of points>]
Calculates the exponential regression curve
* FitLine[<list of points>]
Calculates the y on x regression line of the points.
* FitLineX[<list of points>]
Calculates the x on y regression line of the points.
* FitLog[<list of points>]
Calculates the logarithmic regression curve
* FitPoly[<list of points>, <number n>]
Calculates the regression polynomial of degree n
* FitPow[<list of points>]
Calculates the regression curve in the form a x^b.
All points used need to be in the first quadrant of the coordinate system.
* PMCC[<list of x-coordinates> , <list of y-coordinates>]
* PMCC[<list of points>]
Product moment correlation coefficient
* Polynomial[<list of points>] (in 3.0, undocumented)
Interpolation polynomial of degree (n-1) through n points.

Number Theory

* BinomialCoefficient[<Number n>, <Number r>]

Calculates the binomial coefficient "n choose r".
* GCD[<number a>, <number b>]
* GCD[<list>]

Greatest common divisor
(UK_English HCF Highest common factor)
* LCM[<number a>, <number b>]
* LCM[<list>]

Lowest common multiple (UK) of two numbers a and b or elements of the list
Least common multpile (US)

Calculus and Pre-calculus

* Ellipse[<point A>, <point B>, <point C>]

Draws an ellipse with foci A and B passing through C
* Expand[<function>]

Multiplies out the brackets and simplifies
e.g. Expand[(x+3)(x-4)] gives you f(x) = x^2 - x - 12
e.g. Expand[x^3 + x^3] gives f(x) = 2 x^3
* Factor[<polynomial>]

Factors the polynomial
e.g. Factor[x^2+x-6] gives you f(x) = (x-2)(x+3)
* Hyperbola[<point A>, <point B>, <point C>]

Draws a hyperbola with foci A and B passing through C
* Simplify[<function>]

e.g. Simplify[x + x + x]
* TrapezoidalSum[<function>, <start>, <end>, <# steps>]

Works the same way as UpperSum[] and LowerSum[]
e.g. TrapezoidalSum[x^2, 1, 2, 5]

Technical Controls

* AxisStepX[]
* AxisStepY[]

Return the current step for the x-axis or y-axis respectively.
Together with the Corner[n] and Sequence[] commands, these allow you to create custom axes.
* AxisStepX[]
* AxisStepY[]

Return the current step for the x-axis or y-axis respectively.
Together with the Corner[n] and Sequence[] commands, these allow you to create custom axes.
* TableText[<list1>, <list2>, <list3>, ...]

Creates a text that contains the table of list objects.
* TableText[<list1>, <list2>, <list3>, ... , <orientation>]

The optional text controls the orientation and alignment of the table.
Possible values: "vl", "vc", "vr", "v", "h", "hl", "hc", "hr"

v = vertical, i.e. lists are columns
h = horizontal, i.e. lists are rows
l = left aligned
r = right aligned
c = centered
Default is "vl"
e.g. TableText[{ x^2, x^3, x^4 }] 1 column, left aligned
e.g. TableText[Sequence[i^2, i, 1, 10]] 1 column, left aligned
e.g. TableText[{1,2,3,4},{1,4,9,16},"v"] 2 columns, left aligned
e.g. TableText[{1,2,3,4},{1,4,9,16},"h"] 2 rows, left aligned
e.g. TableText[{11.2,123.1,32423.9,"234.0"},"r"] 1 column right aligned

* Text[<object>]
* Text[<object>, <substitute values for variables>]
* Text[<object>, <point>]
* Text[<object>, <point>, <substitute values for variables>]

Returns the formula for the object as a text object, with or without variables substituted
Point defines where the text will be drawn
e.g. a = 2
c = a^2
Text[c] and Text[c, true] both return "4"
Text[c, false] returns "a^2"
Text["hello", (2,3)] draws the text at (2,3)