

# Dr. Geo, be a geometer

---

The GNU interactive geometry, version 16.10

Hilaire Fernandes ([hilaire@drgeo.eu](mailto:hilaire@drgeo.eu))  
Edward Cherlin ([echerlin@gmail.com](mailto:echerlin@gmail.com))

---

This manual is for GNU Dr.Geo (version 16.10), an environment for interactive geometry and programming.

Copyright © 2002, 2003, 2004, 2005, 2009, 2010, 2011, 2012, 2015, 2016 Hilaire Fernandes

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

# Short Contents

Preliminary .....	1
<b>Partie I Getting started</b>	
1 Basic functionality .....	7
2 Files and documents .....	20
<b>Partie II Advanced functionalities</b>	
Introduction .....	25
3 Macro Construction .....	26
4 Dr.Geo Smalltalk script .....	31
5 Dr.Geo Smalltalk sketch .....	44
<b>Partie III Applications</b>	
6 Didactic applications .....	63
7 Various tips .....	74
A Dr.Geo Glossary .....	79
B GNU Free Documentation License .....	84
Conceptual index .....	92
Method index .....	94

# Table of Contents

Preliminary .....	<b>1</b>
 <b>Partie I Getting started</b>	
<b>1 Basic functionality .....</b>	<b>7</b>
1.1 Tools for construction.....	7
1.1.1 Point tools.....	8
1.1.1.1 Free point .....	8
1.1.1.2 Middle.....	8
1.1.1.3 Intersection.....	8
1.1.1.4 Point defined by coordinates .....	8
1.1.2 Line tools.....	9
1.1.2.1 Line .....	9
1.1.2.2 Parallel line.....	9
1.1.2.3 Perpendicular line.....	9
1.1.2.4 Perpendicular bisector .....	9
1.1.2.5 Angle bisector .....	9
1.1.2.6 Ray .....	10
1.1.2.7 Segment .....	10
1.1.2.8 Vector.....	10
1.1.2.9 Circle.....	10
1.1.2.10 Arc by three points .....	10
1.1.2.11 Arc, centre .....	10
1.1.2.12 Polygon .....	10
1.1.2.13 Polygon regular.....	11
1.1.2.14 Locus .....	11
1.1.3 Transformation tools.....	11
1.1.3.1 Reflection.....	11
1.1.3.2 Symmetry .....	11
1.1.3.3 Translation .....	12
1.1.3.4 Rotation.....	12
1.1.3.5 Homothety (scale).....	12
1.1.4 Numerics and text tools .....	12
1.1.4.1 Distance, length & value.....	12
1.1.4.2 Angle.....	13
1.1.4.3 Coordinates .....	13
1.1.4.4 Free text .....	13
1.1.4.5 Inserting a Smalltalk script .....	13
1.1.4.6 Editing a Smalltalk script .....	13
1.1.5 macro construction tools.....	13
1.1.5.1 Create a macro construction .....	14
1.1.5.2 Execute a macro construction.....	14
1.1.6 Misc tools .....	14
1.1.6.1 Delete an object .....	14
1.1.6.2 Change the style of an object .....	14
1.1.6.3 Object property .....	16
1.2 Misc functions.....	18

1.2.1	Moving the Sketch.....	18
1.2.2	Scaling the sketch .....	18
1.2.3	Moving an object .....	18
1.2.4	Merging points .....	18
1.2.5	Cloning curve.....	19
1.2.6	Grid.....	19
<b>2</b>	<b>Files and documents.....</b>	<b>20</b>
2.1	Rename a sketch .....	20
2.2	Save a sketch.....	20
2.3	Export a sketch .....	20
2.4	Save a session .....	20
2.5	Save a macro construction .....	21
2.6	Open a file .....	21
 <b>Partie II Advanced functionalities</b> 		
	<b>Introduction.....</b>	<b>25</b>
<b>3</b>	<b>Macro Construction .....</b>	<b>26</b>
3.1	Creating a macro construction .....	27
3.2	Play a macro construction .....	28
3.2.1	With the dialog box .....	29
3.2.2	With the macro construction menu.....	29
<b>4</b>	<b>Dr.Geo Smalltalk script .....</b>	<b>31</b>
4.1	Script by the example .....	33
4.1.1	Script without parameter .....	33
4.1.2	Script with one input parameter .....	36
4.1.3	Script with two input parameters:.....	37
4.1.4	Detailed example with several scripts .....	37
4.2	Reference methods for Dr.Geo scripts .....	40
4.2.1	Math Item.....	40
4.2.2	Point .....	41
4.2.3	Straight or curved line .....	42
4.2.4	Line, ray, segment, vector .....	42
4.2.5	Segment .....	42
4.2.6	Circle, arc, polygon .....	43
4.2.7	Value.....	43
4.2.8	Angle .....	43
<b>5</b>	<b>Dr.Geo Smalltalk sketch .....</b>	<b>44</b>
5.1	Smalltalk sketches by example .....	44
5.2	Reference methods for the Dr.Geo Smalltalk sketch .....	45
5.2.1	Various messages .....	45
5.2.2	Modification of object attributes .....	56
5.2.3	Complementary methods .....	57
5.3	Gallery of examples.....	58
5.3.1	Animate a figure.....	58
5.3.2	Sierpinski triangle .....	59

## Partie III Applications

<b>6</b>	<b>Didactic applications</b> .....	<b>63</b>
6.1	Perimeter and area .....	63
6.2	Theorem and conjectures .....	64
6.3	Irrational numbers.....	66
6.4	Baravelle spiral.....	68
6.5	Pappus Chain .....	70
6.6	$\pi$ calculus .....	71
<b>7</b>	<b>Various tips</b> .....	<b>74</b>
7.1	Programming.....	74
7.1.1	Workspace.....	74
7.1.2	Profiler.....	76
7.1.3	Debugger .....	77
7.1.4	Inspector .....	78
<b>Appendix A</b>	<b>Dr.Geo Glossary</b> .....	<b>79</b>
<b>Appendix B</b>	<b>GNU Free Documentation License</b> .....	<b>84</b>
	<b>Conceptual index</b> .....	<b>92</b>
	<b>Method index</b> .....	<b>94</b>

# Preliminary

Dr.Geo permits its users to create geometric figures, called sketches within Dr. Geo, and to manipulate them interactively, observing assigned geometric constraints. It also offers a gentle path into programming. Dr.Geo is usable in teaching situations from the primary to advanced levels. Its user interface is designed as a harmonious combination of simplicity and ease of use with advanced capabilities.

The Dr.Geo user interface, under the form of great simplicity, allows the beginner to get acquainted very quickly with the basic functions of the software. Then, as the user progresses, the more advanced aspects of the interface and the capabilities of Dr.Geo will become apparent, such as multiple methods of construction of each kind of object,<sup>1</sup> macro construction, multiple recording, scripts, Smalltalk sketches, and inheritance of Smalltalk in Dr.Geo. These advanced features generate little overload on the interface, which is why Dr.Geo is very pleasant for use in primary education; however they also make it very interesting for use in high school and university.

In the following sections, we explain the basic tools. Then the advanced features are presented in detail. We begin with a blank sketch. Open Dr.Geo, revealing a menu bar, a tool bar, and the welcome graphic. Click the new sketch button at the left of the toolbar.

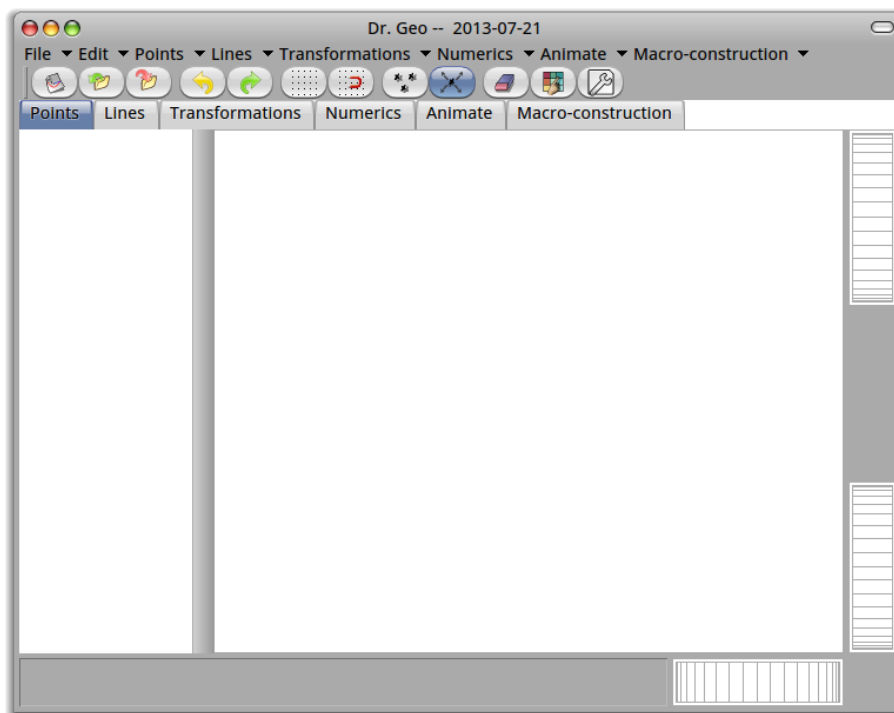


Figure 1: Dr.Geo window with a blank figure

---

<sup>1</sup> A particular command can create an object from different parameters. For example with the command for constructing a circle, the user can specify its centre and a point on the circumference or the length of its radius, among other options. Of course this command is represented only by one button, with Dr.Geo anticipating the construction intended by the user from the context. The immediate effect is thus a decrease of the cognitive load in the user interface, while offering a significant expansion of capabilities.

The layout of the interface is as follows:

1. A *menu bar* with: File – Edit – Points – Lines – Transformations – Numerics – Animate – Macro construction
2. An *editing toolbar* New figure; open and save figures; undo last action; redo last action; display grid; snap to grid; mode toggle for single or multiple object creation; select and move an object; erase an object; modify the style of an object; modify object properties.
3. A *construction toolbar*, which groups the tools for constructing objects in several tabs. It has the same functions as corresponding items on the menu bar.
4. In the right bottom corner, two wheels for moving the sketch horizontally and vertically.
5. In the right top corner, a wheel for scaling the sketch.

To create a new sketch, the user selects the New command in the File menu. We will indicate such menu choices in the form File→New from here on. For each new sketch, a distinct window opens with its own menu and toolbar. The user can then create points, lines, circles, and so on, and control their properties.

## Philosophy of the application

Dr.Geo II is Free Software<sup>1</sup> for interactive geometry on multiple platforms. It is a complete rewrite of Dr.Geo 1.1 in Pharo Smalltalk – <http://pharo-project.org>.

Dr.Geo 1.1 was written in C++ and integrated with a Scheme interpreter for writing scripts to make interactive programmed sketches. Dr.Geo II also provides the integration of scripts in geometric sketches so that interactive sketches can be written in a programming language.

The rewrite in Smalltalk was motivated by its unique dynamic qualities; it in fact allows us to push very far in our investigations of the interactive possibilities between the user and the application. So Dr.Geo is not only a convivial interactive geometry application but it is also, as distributed, a complete programming environment to study, to modify and to improve it.

For get an idea of these capabilities, the user is invited to click in the background of Dr.Geo – outside any window – and to press the key combination *control-b*. Choose **Tools>System Browser** from the menu. The system browser that appears lets the user explore and modify Dr.Geo source code while it is running.

This access to the source code is part of the Dr.Geo DNA; it is part of the free software philosophy as well, for a completely open approach, unlocked, with great opportunities for learners. Far be it from us to pretend that Dr.Geo is enough for building mental capacity, but it certainly can help.

---

<sup>1</sup> Free Software is software under a Free license such as GPL, which requires that its source code be made available for study, modification, and redistribution under the same license.



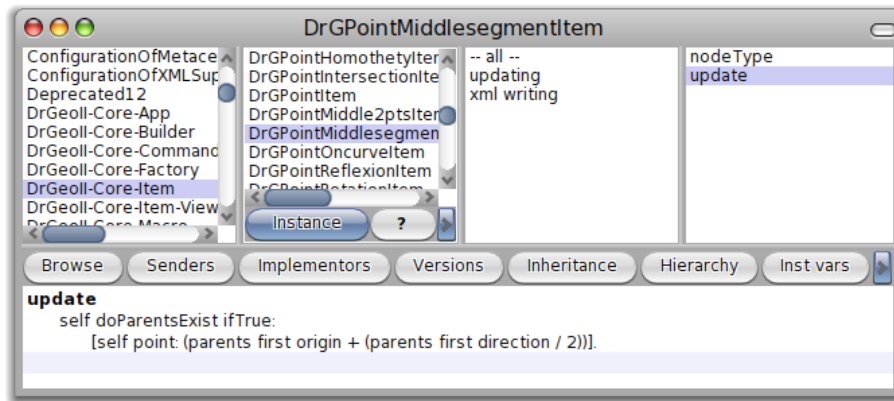


Figure 2: System browser opened on the Dr.Geo source code from Dr.Geo itself

With this same spirit of openness, programmed sketches and scripts – presented in the advanced tools section – rely on an advanced set of developer tools: browser, debugger, inspector, and more.

In the following we will not distinguish between the application names Dr.Geo II or Dr.Geo.

## Dr.Geo on the web

Dr.Geo has its own web site at <http://drgeo.eu>. Here you will find the following information:

- how to get Dr.Geo,
- software documentation,
- information about the Dr.Geo project,
- reference material on applications of Dr.Geo in teaching.



# Partie I

## Getting started



# 1 Basic functionality

This chapter describes the tools used to construct a geometric sketch.

## 1.1 Tools for construction

These tools are ordered on six tabs. Clicking any of these tabs brings up an appropriate toolbar.

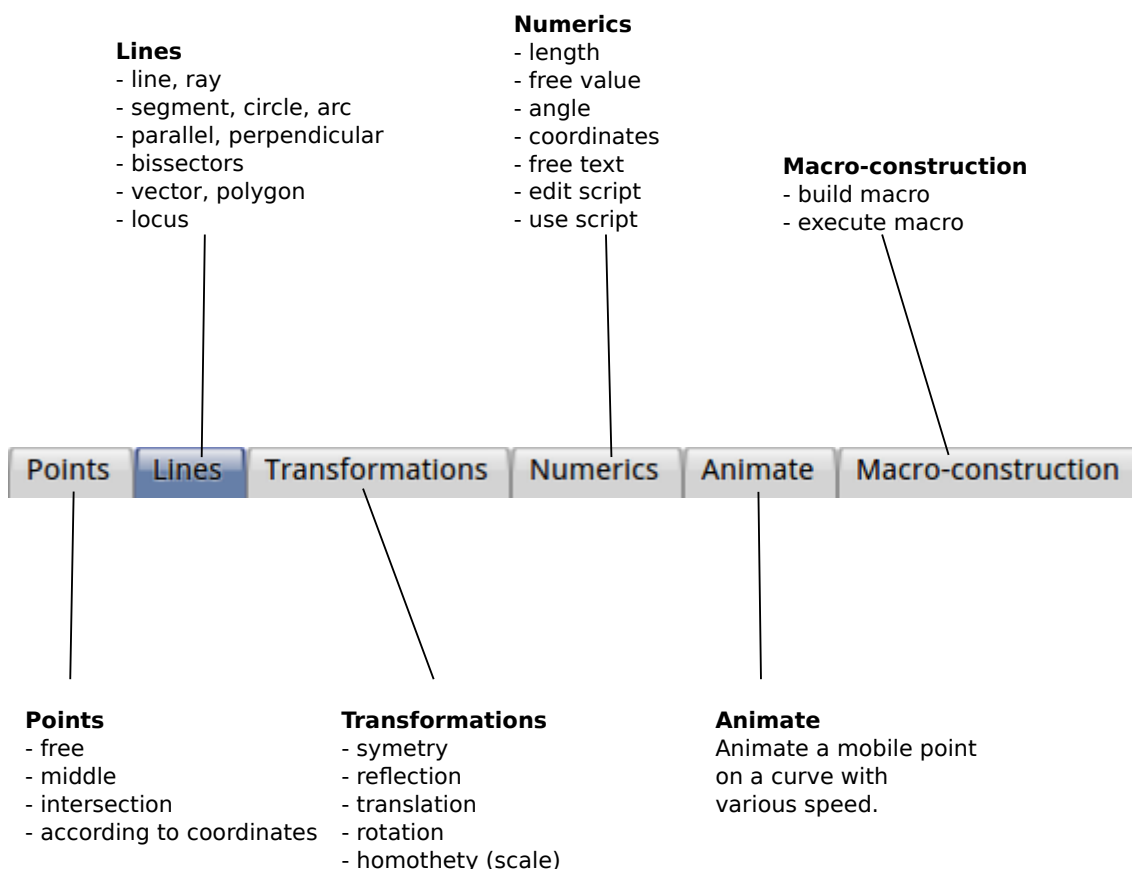


Figure 1.1: Tool categories and the descriptions

Dr.Geo defaults to a functional mode of operation. The user must click on a function tab and then a function button, or select a function from a menu, before selecting the object or objects to apply it to, and executing the function once. At this point, the application returns to its default state, in which the user can select and move objects. The user can abort any function before making the final selection with the Select and move button on the toolbar, or the Edit→Select and Move command.

This is in contrast to a modal application, in which clicking a function button puts the application into a mode that allows the command to be used multiple times in succession. To activate the modal mode the user selects Edit→Create multiple or the corresponding button in the toolbar. The commands for editing object styles and properties are always modal, allowing any number of objects to be modified without having to select the function repeatedly to apply it to each one. The user has to explicitly exit them, for example by returning to the select and move mode or starting some other command.

There is a mode toggle button for single/multiple creation, which switches from this functional mode to this modal mode, in which function selection is persistent. That is,

selecting the line function allows the user to create multiple lines without having to select that function over again for each one.

When the user clicks one of these tabs, an additional toolbar displays immediately. It groups functions of the same family.

From left to right, we have access to tools to build points and lines, use transformations, calculate values, animate sketches, and manage macro construction and execution.

These functions also appear in the menus at the top of every Dr.Geo window.

### 1.1.1 Point tools

#### 1.1.1.1 Free point

##### Point

Create a free point in the plane or on a line object (segment, ray, straight line, arc, circle, locus):

1. In the first case, the created point can be moved anywhere in the plane of the sketch. To create it the user simply clicks in the background.
2. In the second case, the point can move freely only in the line where it was created; it is stuck on the object. To create this type of point, the user clicks on a line (i.e. straight line, ray, segment, circle, arc, etc.).

From this command, you can also create one **intersection point** between two lines (i.e. straight line, ray, segment, arc, circle). It is enough to click in the intersection of these. Moreover, Dr.Geo indicates it by the help bubble: *Intersection*.

#### 1.1.1.2 Middle

##### Middle

Create the middle point between two points, or the midpoint of a segment:

1. In the first case, the user selects two points.
2. In the second case, the user selects a segment.

#### 1.1.1.3 Intersection

##### Intersection

Create one or several intersection(s) of two lines (i.e. straight line, ray, segment, arc, circle). The user has to select two lines. When one or both of the chosen lines is an arc or a circle then two intersection points may be created.

#### 1.1.1.4 Point defined by coordinates

##### Coordinates

## How to place a point by specifying its coordinates

A possibility is to place two free values in the sketch – See [\[numeric\\_tool\]](#), page 12 – then build the point with these two values as coordinates – See [\[coordinate\\_point\]](#), page 8. This possibility has an advantage over the previous one for some purposes. The point so constructed cannot be directly moved with the mouse. The point is constrained in its position.

Create a point defined by its coordinates. The user selects a script returning a pair of x-y coordinates – See [\[scriptCoordinates\]](#), page 38 – or two values: the first one is the abscissa (the x coordinate) and second one the ordinate (the y coordinate).

## How to construct a point constrained by its coordinates?

This function is mainly used to construct a locus of point.

This construction supposes beforehand the existence of two values – See [\[numeric\\_tool\]](#), page 12 – the point is then constructed by selecting these two values.

### 1.1.2 Line tools

#### 1.1.2.1 Line

Line

Create a straight line defined by two points. The user clicks on two points, or clicks on one point and drags to another.

#### 1.1.2.2 Parallel line

Parallel

Create a line parallel to a given direction and going through a point. The user clicks a point and a direction (i.e. straight line, ray, segment or vector).

#### 1.1.2.3 Perpendicular line

Perpendicular

Create a line perpendicular to a given direction and going through a point. The user clicks a point and a direction (i.e. straight line, ray, segment or vector).

#### 1.1.2.4 Perpendicular bisector

Perpendicular bisector

Create a perpendicular bisector given a segment or two points. The user clicks on a segment or two points, or clicks and drags.

#### 1.1.2.5 Angle bisector

Angle bisector

Create the angle bisector of an angle formed by three points. The users clicks on a geometric angle (defined by three points)

or clicks on three points. The line will bisect the angle at the second point.

### 1.1.2.6 Ray

#### Ray

Create a ray defined by two points. The user clicks on two points. The first one is the origin, the second one a point on the ray.

### 1.1.2.7 Segment

#### Segment

Create a segment given two points.

### 1.1.2.8 Vector

#### Vector

Create a vector given two points. The user select two points, the origin, the second one the terminal point.

Once the vector is created, it can be moved independently of the two points. This remains true when the vector is built from a transformation – See [\[transformations\]](#), page 11.

### 1.1.2.9 Circle

#### Circle

Create a circle. The user can create it from different selections:

1. the centre and a point on the circle,
2. the centre and a number (the radius),
3. the centre and a segment whose length is the radius.

### 1.1.2.10 Arc by three points

#### Arc

Create an arc going through three points. The first one is the starting point, the third one is the end point, and the second one is a point on the arc.

### 1.1.2.11 Arc, centre

#### Arc (center)

Create an arc defined by its center and by starting and ending points. The first one is the centre, the second one is the starting point, and the third one is the end point.

### 1.1.2.12 Polygon

#### Polygon

Create a polygon defined by  $n$  points. The user selects  $n+1$  points defining the polygon submits. The first and last ones are the same point, it indicates to Dr.Geo the selection is over. Mobile point can be added on polygon, but it is not possible to compute an intersection with another line. However geometric transformation of polygon are allowed.



### 1.1.2.13 Polygon regular

#### Regular polygon

Create a regular polygon defined by two points and a numeric value. The user selects its center, one vertex, and a value indicating the number of vertices. If the value selected is not an integer, it is truncated to the next lower integer.

### 1.1.2.14 Locus

#### Locus

Locus defined by two points. The user selects two points, one free on a line, the other one depending on it, so that moving the first point makes the location of the second point change. This dependent point is called a relative point.

For example, we can construct an ellipse based on its property that the sum of the distances from each point on the ellipse to two given points is constant.

1. Create free points O and A, which will be the given points.
2. With center O and radius greater than the length OA, create a circle.
3. Create a free point B on the circle.
4. Join the segments OA and OB.
5. Construct the point C, the midpoint of AB.
6. Create a perpendicular bisector to AB.
7. Construct the point D, the intersection of OB and the perpendicular bisector to AB.
8. Construct the locus with free point B and relative point D.

Since CD is the perpendicular bisector of AB, the triangles ACD and BCD are congruent (side-angle-side). Therefore  $AD = BD$ . Then  $AD + DO = BD + DO$ , which is the radius BO, which is constant. D lies on the ellipse. Moving B all around the circle moves D all around the ellipse.

## 1.1.3 Transformation tools

### 1.1.3.1 Reflection

#### Reflection

Transform an object by a reflection. The user clicks on the object to transform and the axis – straight line – of the reflection. When the object to transform is a straight line, the first clicked is the axe of the reflection.

### 1.1.3.2 Symmetry

#### Symmetry

Transform an object by a central symmetry. The user clicks on the object to transform and the centre of symmetry (a point). When the object to transform is a point, the first clicked is the centre of the symmetry.

### 1.1.3.3 Translation

#### Translation

Transform an object by a translation. The user clicks on the object to transform and a vector. When the object to transform is a vector, the first clicked is the vector of translation.

### 1.1.3.4 Rotation

#### Rotation

Transform an object by a rotation. The user clicks on the object to rotate, a centre (point) and an angle. When the object to transform is a point, the first clicked is the centre.

The rotation angle can be defined by three different means:

- **numeric value:** the angle is therefore the value interpreted in radians (Examples of numeric value: free value, distance between two points, segment length, a coordinate, a value returned by a Dr.Geo script, etc.),
- **a geometric angle defined by three points:** its measure is displayed in degrees. Beware, in this case the measure ranges only in the interval  $[0 ; 180]$ . The angle has to be put in numeric form using the angle function described below.
- **an oriented angle between two vectors:** its measure is displayed in degrees in the interval  $]-180 ; 180]$ . The angle has to be put in numeric form using the angle function described below.

### 1.1.3.5 Homothety (scale)

#### Homothety (scale)

Transform an object by scaling it. The user clicks on the object to transform, the centre and the factor (i.e. a number). When the object to transform is a point, the first clicked is the centre.

## 1.1.4 Numerics and text tools

### 1.1.4.1 Distance, length & value

#### Distance, length, value

Create a numeric value. The value, depending on the user selection, may be computed or user edited:

1. for two points it is the distance between these two,
2. for a segment it is its length,
3. for a vector it is its norm,
4. for a circle it is its perimeter,
5. for an arc it is its length,
6. for a line it is its slope,
7. for a point and a line it is the distance between these two,
8. a **mouse click on the background** creates a free value, initially 0, that the user can edit.

A newly created free value can be edited by clicking it. Later on, it can be edited by clicking the property button on the toolbar, described below, and then selecting the value.

This last possibility is very interesting for some situations. It lets the user set an arbitrary value to use for a given length, the radius of a circle, an angle measure (in radians) or the coordinates of a point. The value is then used by other specific tools to construct a circle, a rotation or a point defined by its coordinates.

#### 1.1.4.2 Angle

Angle

Compute the angle defined by three points or two vectors. In the first case, the angle is considered as non oriented (i.e. angle whose measure belongs to the interval  $[0 ; 180]$ ). In the second case, the angle is oriented and its measure belongs to the interval  $]-180 ; 180]$ .

#### 1.1.4.3 Coordinates

Coordinates, equation

Create the coordinates (abscissa and ordinate) of a point or vector.

#### 1.1.4.4 Free text

Text

Add a block of text in the sketch. The user clicks at the desired position, then edits the text. To edit the text again, the user clicks the property button described below, then the text.

#### 1.1.4.5 Inserting a Smalltalk script

Use a script

Insert a Dr.Geo Smalltalk script in the sketch. The script expects a number of objects as arguments, and requires a target to attach the output to. It returns an object whose text representation is printed in the sketch, at the position clicked by the user. A script can be used for its side effects or for the value it returns, ready to be used for subsequent constructions. See the Smalltalk Script chapter for more details.

#### 1.1.4.6 Editing a Smalltalk script

Edit a script

Edit or create a Smalltalk script. A script editor is opened for the user to edit any existing script or to create a new script.

The Dr.Geo Smalltalk scripting system is reviewed in detail in the chapter on advanced functionalities (See [\[script\]](#), page 30).

#### 1.1.5 macro construction tools

### 1.1.5.1 Create a macro construction

#### Build macro

Extract a construction sequence from the sketch and compile it as a macro construction.

macro constructions are covered in detail in its section (See [\[macroConstruction\]](#), page 26).

### 1.1.5.2 Execute a macro construction

#### Execute Macro

Execute a macro construction. The macro construction can be newly created or loaded from a file.

## 1.1.6 Misc tools


### 1.1.6.1 Delete an object



An object can be deleted using the eraser tool or the Edit→Eraser command. Objects that depend on the deleted object will also disappear. When deleting a curve, the points used to define it remain behind. The action is undoable with the undo button in the toolbar or the undo menu item in the Edit menu.

### 1.1.6.2 Change the style of an object



Each geometric object has its own style attributes, which can include colour, line style, thickness, label, size and shape, and properties such as Fill, Translucent, and Hidden. An object that is hidden remains in the sketch, but is not seen except when editing properties, so that it can be found. Hiding intermediate construction is often useful to avoid clutter in the sketch. All these attributes are edited from a side panel displayed when the user enters style mode. To do so, click the  button from the toolbar, and then select a particular item in the sketch.

### Style of point.

The lateral panel for style of point is for points within every kind of object. Name, Colour, shape (x, dot, or block), size, visibility, and locking are adjusted from there. Only the position of free points on the plane or on a curve is lockable.

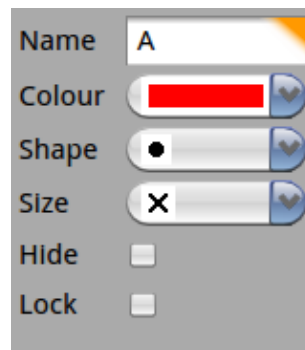


Figure 1.2: Side panel to edit style of point

### Style of line.

The side panel for style of line is for straight lines, rays, segments, vectors, circles, arcs, loci, and polygons. Name, colour, thickness, solid or dashed styles, and visibility are adjusted from there. Some properties only apply to particular kinds of line, as in the case of marks and arrowheads on segments.



Figure 1.3: Lateral panel to edit style of line

### Style of value.

The lateral panel for style of value is for all kinds of numeric value: free value edited by the user, measure, computed results from a Dr.Geo Smalltalk script. Colour, label and visibility are adjusted from here. The value can also be locked.

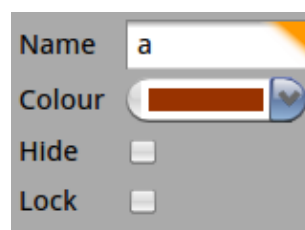


Figure 1.4: Lateral panel to edit style of value

### 1.1.6.3 Object property



Some objects' numeric or text values are editable. This includes the coordinates of free points on the plane or on a curve, free values and scripts. To edit the property, select this tool, then choose one of these objects in the sketch. A dialog box will appear:

#### Free point.

Selecting a free point, a dialog box let you edit its abscissa and ordinate.

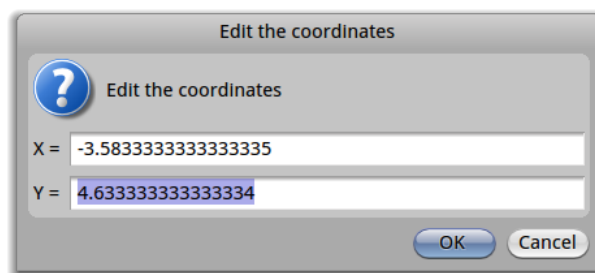


Figure 1.5: Edit the coordinates of a free point

#### Free point on curve.

Selecting a free point on curve, a dialog box let you edit its curvilinear abscissa. It is normalised in the interval  $[0 ; 1]$ . This allows a curve to be treated as a parametric curve.

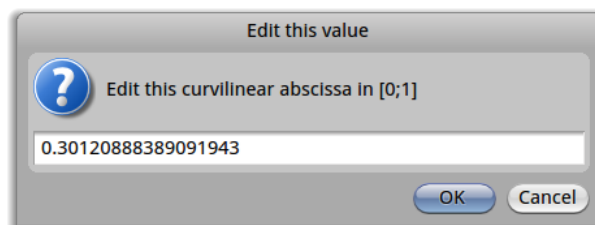


Figure 1.6: Edit the curvilinear abscissa of free point on a curve

#### Free value.

Selecting a free value, a dialog box let you edit its value.

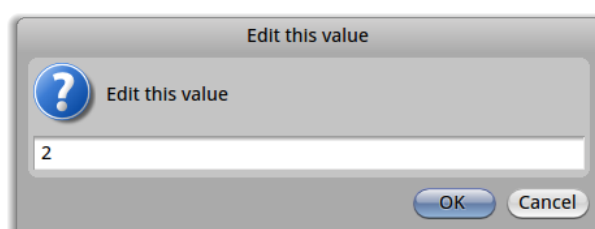


Figure 1.7: Edit a free value

## Script.

Selecting the value displayed from a script, an editor is opened to let you study and edit it. To save any modification to a script, use the key shortcut *Ctrl-s* or the entry *Do-it* in the contextual menu over the script (right mouse click to display it).

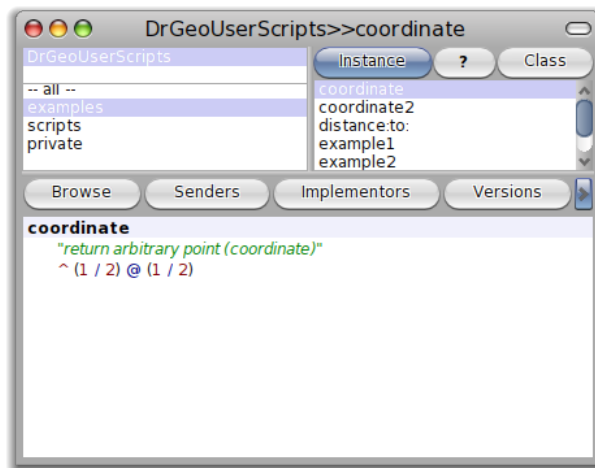


Figure 1.8: Edit a script

## Text.

Selecting a script, a dialog box let you edit it. The text can be formatted in several lines with carriage return – keyboard key *Return*.

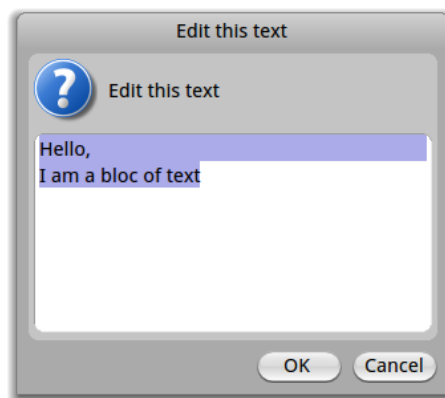


Figure 1.9: Edit a text

## 1.2 Misc functions

### 1.2.1 Moving the Sketch

The sketch can be moved with the wheels in the right bottom corner or directly with the right mouse button.

### 1.2.2 Scaling the sketch

The sketch scale can be adjusted with the wheel on the top right of the window. The mouse wheel offers this same service; press simultaneously the keyboard key *Shift* to speed up the scaling.

### 1.2.3 Moving an object



An object is moved by clicking and dragging with the mouse. The whole sketch is then recomputed and redrawn to respect the constraints specified. Almost all geometric objects can be moved. If necessary, Dr.Geo moves the associated free points. For example, when the user drags a line defined by two points, the points are moved simultaneously.

In this mode, it is possible to change the nature of a point from:

- a free point on the plane
- a free point on a curve
- an intersection point

to a point as:

- a free point on the plane
- a free point on a curve
- an intersection point

For example, transforming a free point on the plane into an intersection point. However, there is one constraint: it is not possible to mutate a point toward a “younger” line (as free on the line or intersection with the line). Younger line means created chronologically after the point.

Pressing the keyboard key *Shift* while grabbing a point will display a balloon help text about the point you can mutate and the possible destinations.

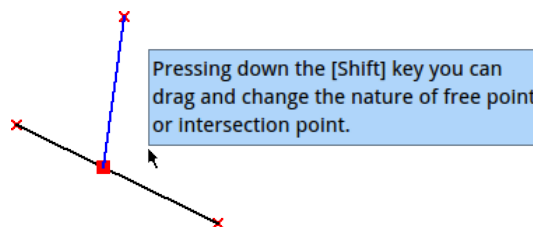


Figure 1.10: Pressing [Shift] to mutate a point

### 1.2.4 Merging points


Two free points can be merged: to do so drag one point over the second destination point and hold for a few seconds.




### 1.2.5 Cloning curve

Various type of curve can be cloned to quickly duplicate identical objects. To do so press and hold the mouse button over the curve to clone. The curve will blink, and then a cloned curve is displayed, slightly offset from the original, ready to be positioned as the user chooses.

### 1.2.6 Grid

It is possible to show or hide a grid in every sketch with the  button in the toolbar. The scale of the grid adjusts itself when the sketch is enlarged or shrunk. When saving a sketch, the grid state is saved as well.

When creating or moving points, it is possible to constrain them to snap to the grid, so that the point will appear at the grid position closest to the mouse pointer. To do so, activate the magnet tool from the  button in the toolbar.

## 2 Files and documents

Sketches are saved in two different ways, one sketch per file or a set of sketches and macro constructions in one file (i.e. a Dr.Geo session). We recall that documents are saved with the extension **.fgeo**. When providing the file name, you don't need to write this extension. Dr.Geo will do it for you.

### 2.1 Rename a sketch

From the menu File→Change title, the user changes the name and the title of the active window. This function is useful when the user saves a set of sketches in a single session file.

### 2.2 Save a sketch

From the menu File→Save, the sketch of the active window is saved. The user is prompted for a name.



Dr.Geo can work with several sketches at the same time. The user switches from one sketch to another one with the task bar in the bottom of the Dr.Geo environment.

With the menu File→Save as..., the user saves the document under another name.

The documents are saved in the Dr.Geo application structure, in the folder **DrGeo.app/MySketches**. To save at an arbitrary location in the hard disk structure, use the menu File→Save at...



When saving a document, a network option is available in the dialog box. If selecting this option, the user must also provide a network share name – arbitrarily chosen by the user – and the document name. This option lets a group of networked users share sketches

### 2.3 Export a sketch

The menu File→Export as bitmap exports the sketch of the active window in a picture file in the standard format PNG<sup>1</sup>.

The pictures are saved in the Dr.Geo structure application, in the folder **DrGeo.app/MyExports**.

### 2.4 Save a session

A session is a set of Dr.Geo documents – sketches and macro constructions – the user saves in **one** file. It eases the preparation of pedagogical sequences.

From the menu File→Save multiple, the users accesses a dialog box to select the document to integrate in the session file.

---

<sup>1</sup> <http://www.w3.org/Graphics/PNG>

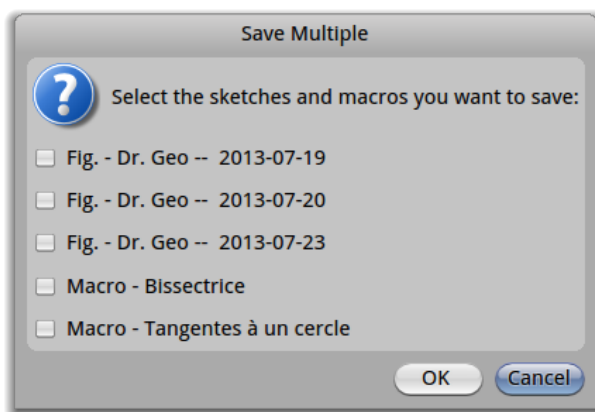


Figure 2.1: The Dr.Geo session dialog box

In this dialog, the list of all active documents is presented in column. Each line is prefixed with a tag **Fig.** or **Macro** according to the document type. The second part is the name of the document.



As for now, a session knows about two types of document: 2D interactive sketch and macro construction.

The user selects the documents to integrate in the session then provides a name for the composite file, and clicks OK.



The menu **File**→**Save multiple** is the only method for saving a macro construction in a file.

## 2.5 Save a macro construction

To save one or several macro constructions in a file, proceed the same way as for saving a session – multiple save. From the dialog box to save a session, select one or several macro construction to save, then input the file name. That's all.

Therefore, it is possible to build macro construction libraries, one per file or several per file following a given theme.

## 2.6 Open a file

Whenever the user saved an individual sketch or a session in a file, the procedure to open it is the same from the menu **File**→**Open**.

If the session contains macro constructions, they are automatically added to the macro menus and player, ready to use in any existing Dr.Geo window.



# **Partie II**

## **Advanced functionalities**



## Introduction

In this part, we present features to extend Dr.Geo's capabilities, or to adapt it to a given pedagogical situation.

The first feature is the *macro construction*. It lets the user record a sequence of construction steps in a single command. Once recorded, it can be replayed and saved in a file to be opened later in another sketch.

the Dr.Geo Smalltalk script is another feature to extend Dr.Geo behaviour. A script appears as a sketch item like any other. As input, it can expect zero, one, or more references to geometric items from the sketch, and it returns a reference whose print representation is placed in the sketch. It is in fact a programmed method<sup>1</sup> placed in a sketch, and evaluated each time the whole sketch is updated (i.e. when the sketch need to be redrawn). A Dr.Geo Smalltalk script may be useful for its returned object or for its side effects, depending on the wishes of the user.

Going a step further, Dr.Geo offers Smalltalk sketches. In this case the whole interactive geometry sketch is described in Smalltalk source code. Its interest is the functional approach to describing a sketch<sup>2</sup> compared to the linear, declarative method of constructing a sketch with the mouse.

---

<sup>1</sup> Comparable to a procedure for users of other programming languages.

<sup>2</sup> For example recursively

### 3 Macro Construction

A macro construction is similar to a programmed procedure or function, receiving as input items from the sketch and providing as outputs another set of items in the sketch. A macro is built according to a model defined by the user. That means that the user first needs to do the complete construction sequence in the sketch, then ask to Dr.Geo to record it in a macro construction. The macro construction can then be replayed and saved in a file like any sketch.

To record a construction sequence, Dr.Geo needs to know the initial items of the sequence and the final – output – items. Of course, the final items must depend *only* on the input items, otherwise Dr.Geo cannot deduce the final items from the initial ones.

Thus, Dr.Geo traces the logic of the construction sequence for the specified outputs, and records it in a macro construction. Then, the user can repeat this sequence. When playing the macro construction, it only asks for the initial items – of the types specified – in the sketch and constructs the resulting – output – items.



Intermediate invisible items are constructed by the macro construction. They are necessary to reproduce to complete the construction sequence and the resulting output items.

To illustrate the use of a macro construction, we take the example of a circle going through three points.

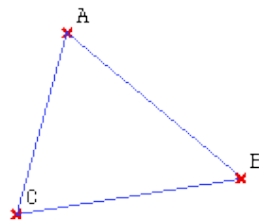


Figure 3.1: The initial sketch

Before creating the macro construction, the user has to construct the final sketch to serve as a model, as shown in the figure below.

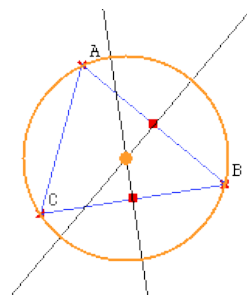



Figure 3.2: The sketch with the final construction



### 3.1 Creating a macro construction

The user now tells Dr.Geo he wants to define a macro construction from this sequence.

He activates the function Build a macro from the toolbar  or from the menu macro construction→Build macro.

In the newly displayed dialog box, the user selects the input and output objects, and enters a name and description of the macro construction.

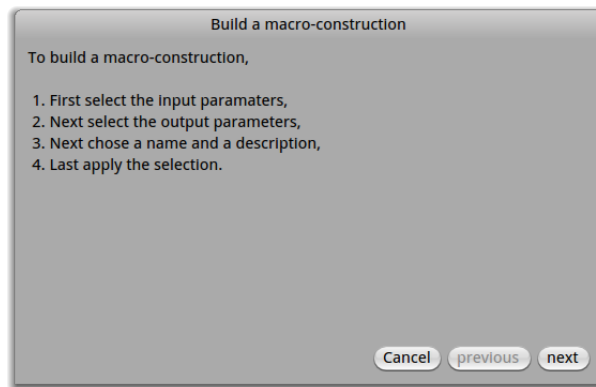


Figure 3.3: Introduction page in the dialog box to construct a macro construction

The second page in the dialog box lets the user select the input objects. In our example, it is the three initial points. The user just needs to go to this second page and then click on the three points A, B and C in the sketch. The selected points blink and the names are displayed in the dialog box.

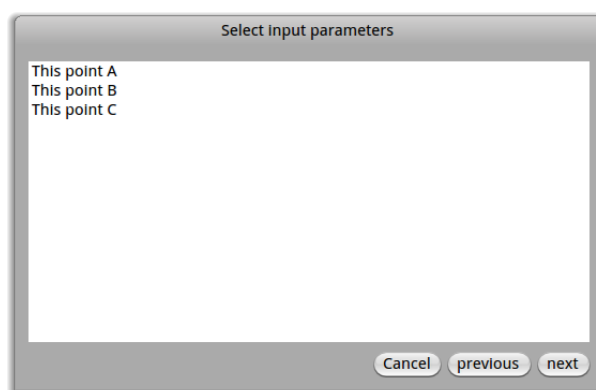


Figure 3.4: The second page with the three points selected

In the third page, the user selects the output objects. In our example, we want the circle and its centre as the result of the macro construction. The user clicks on these two objects in the sketch. When selected they blink, and their names are displayed in the dialog box.

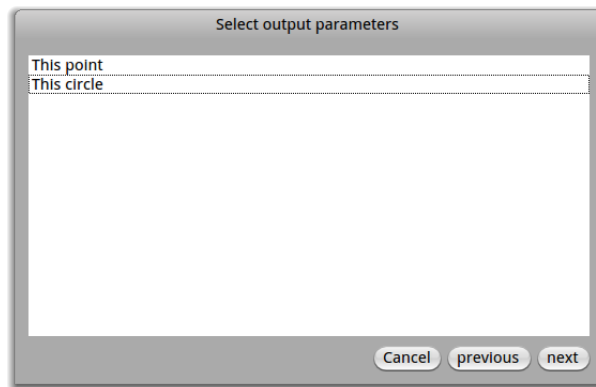


Figure 3.5: The third page with the circle and the centre selected

In the fourth page, the user inputs a name and a description for the macro construction. This information is displayed when the user plays a macro construction. It helps to disambiguate macro constructions.

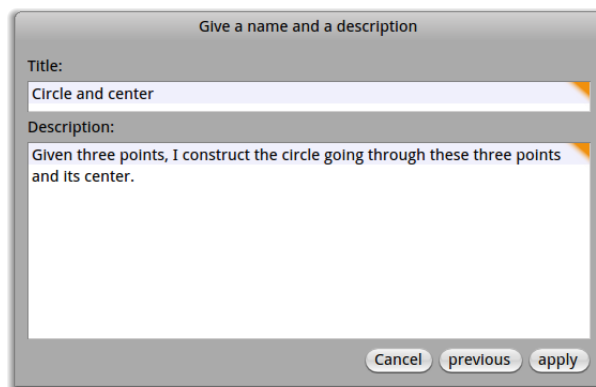


Figure 3.6: The fourth page with the name and description of the macro construction

Next, the user validates the definition of the macro construction by pressing the **Apply** button and examining the results. He can also step back to the previous steps to adjust the parameter objects.




If the selections of the input and output objects do not match (Dr.Geo cannot extract a construction sequence, because an output depends on an object not selected as an input), the macro construction cannot be created. In this case the user needs to reconsider the input and output parameters. He or she can step back to the second or third page in the dialog box to adjust the choices.

At this stage, the macro construction is created and recorded in Dr.Geo. In the next section, we will see how to use it.

## 3.2 Play a macro construction

### 3.2.1 With the dialog box

To execute a macro construction, the user clicks on the  button in the toolbar or selects the menu macro construction → Execute macro in the sketch window. A dialog box describing the procedure appears.

On the second page of this dialog box the user selects the macro construction in the upper list, and then the input parameters in the sketch. Once all are selected, the macro construction is executed automatically and the output parameters are built and displayed in the sketch.

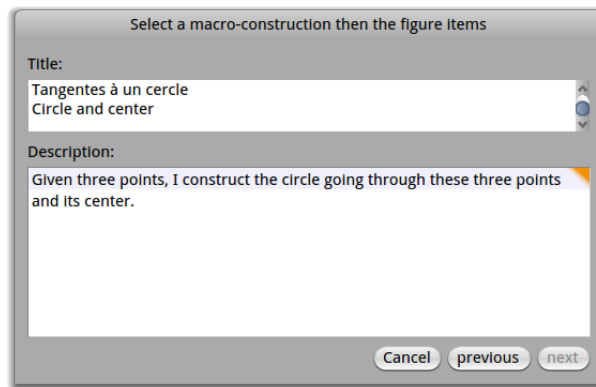


Figure 3.7: The user selects the input parameters in the sketch

In our example, the macro construction expects three input parameters (three points) and constructs a circle and its centre. Therefore, to execute it, you need a sketch with at least three points.



Figure 3.8: Sketch with three points

Once we feed three points to the macro we get the expected circle and centre.

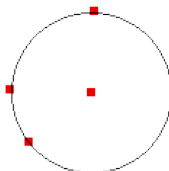


Figure 3.9: A final sketch with the circle and its centre

### 3.2.2 With the macro construction menu

The macro construction loaded in memory are listed in the window menu macro construction. Selecting a macro construction from this menu let you execute it directly. After

choosing one in the menu, select the input parameters in the sketch. The macro construction is executed as soon as all the input parameters are selected.

It is a dialog-less mode of execution.

As described in the Files and documents chapter, to save a macro or collection of macros to a single file, select File→ Save multiple, and then select the items to include in the saved file. The same process works for saving a sketch with multiple macros.

To add a macro from a saved file to a sketch, create or open the sketch and open a file containing just the macro.

## 4 Dr.Geo Smalltalk script

Dr.Geo is a dynamic application written in Pharo Smalltalk. This means that it is possible to modify it within itself as it is running. We have exploited this feature with the Numerics→Use a script command to define sketch items in Dr.Geo which are in fact Smalltalk scripts – code snippets – to extend the possibilities of Dr.Geo dynamically, without limit. But what exactly is Smalltalk?

Smalltalk is an object-oriented, dynamically typed, reflective programming language. Smalltalk was created as the language to underpin the "new world" of computing exemplified by "human–computer symbiosis." It was designed and created in part for educational use, more so for constructionist learning, at the Learning Research Group (LRG) of Xerox Palo Alto Research Center (PARC) by Alan Kay, Dan Ingalls, Adele Goldberg, Ted Kaehler, Scott Wallace, and others during the 1970s. The language was first generally released as Smalltalk-80. Smalltalk-like languages are in continuing active development, and have gathered loyal communities of users around them. ANSI Smalltalk was ratified in 1998 and represents the standard version of Smalltalk.<sup>1</sup>

In this definition, object-oriented means that everything in Smalltalk is an object defined in a class, with its behavior determined by methods defined in that class. This is in contrast with the notion of a variable associated with a data type in most programming languages, where the type system sets few limits on what can be done with values.

In statically typed languages, the type of a variable has to be declared before it can be used, and that type cannot be changed within the program. In particular, the sizes of arrays and other composite data structures must be declared in advance. In dynamically typed languages, a variable name can be repurposed to refer to an object of a different class with a simple assignment, items can be added to or removed from data structures at will, and the class definition can be changed while a program is running.

Reflection in programming languages refers to the ability to examine and modify properties of objects within the live system, rather than having to read the source code in which variables are declared to determine their types and infer their properties. Reflection tools in Smalltalk include the system browser, the inspector, the object explorer, and the method finder.

This abstract from the book preface *Pharo By Example*<sup>2</sup> describes the Smalltalk environment used by Dr.Geo:

Pharo is a modern Open Source development environment for the classic Smalltalk-80 programming language.

Pharo strives to offer a lean, Open Source platform for professional software development, and a robust and stable platform for research and development into dynamic languages and environments.

Dr.Geo uses the Smalltalk environment to create a comfortable environment for writing scripts providing access to the programming interfaces for geometric objects. These interfaces are the set of methods in the definitions of the types of the objects.

Thus the user can write scripts to manipulate the sketch items, and as scripts pinned in the sketch are also sketch items, they do not need to be in separate files, but can be saved in the sketch file.

<sup>1</sup> Wikipedia Smalltalk article (<http://en.wikipedia.org/wiki/Smalltalk>). Page view the 24 July 2013. Contents under the license CC-BY-SA 3.0 (<http://creativecommons.org/licenses/by-sa/3.0/deed.fr>).

<sup>2</sup> <http://pharobyexample.org>

Scripts are not edited as sketch items but as methods stored globally in Smalltalk. Using a script and pinning the result in a sketch makes it part of the sketch so that it can be saved and reloaded with the sketch. However, there is no way to keep scripts in different sketches separate if they are open at the same time. This is a design bug, because users will often have all of their work polluted with scripts meant to be used in other sketches, unless they exercise great care to avoid this. In fact, the default state of Dr.Geo contains a number of useless scripts that could have been put into the examples category or even omitted, so that any new session would start with an empty scripts category.

A workaround is to close Dr.Geo and open a new session after working with a sketch containing multiple scripts and before creating any new sketch, which can then begin without any extraneous scripts.

While it is possible to put scripts for a sketch into a new category for creating them, it is not possible to select by category when using a script. Furthermore, in the use a script dialog, scripts seem to be offered in random order. It is certainly not alphabetical or temporal order.

Scripts in Dr.Geo are executed in the Pharo Smalltalk environment. The user thus benefits from the excellent developer tools of the environment: class browser, inspector, debugger, etc. The user wishing to explore the power of scripts is invited to study the book *Pharo By Example*, where he or she will learn the Smalltalk language and its environment.

For our purposes in this chapter, it is necessary mainly to understand how to use a Workspace. Open a new Workspace by clicking in the background and selecting Tools→Workspace. The workspace provides

- Basic text editing capability
- Syntax highlighting
- Context-sensitive selector (method name) completion
- Access to a menu of editing commands and development tools

Text editing capabilities in a Workspace consist of the usual Cut, Copy, and Paste; Find and Replace; Undo and Do again. An Extended search function displays information about selected items using various development tools. There are also Accept and Cancel functions. Typing in the Workspace causes a highlight to appear at the upper right of the window. Accept checkpoints the contents of the window, empties the undo stack, and causes the highlight to disappear. Editing and then cancelling takes the window back to the last checkpoint without having to undo each action individually.

When editing, object names appear in blue, method names and symbols such as := in black, and unrecognized text in red. This includes incomplete names, undeclared local variables, and also syntax errors. When the user is typing a selector, a menu appears showing selectors starting with the typed text that are usable in the current context, that is, for the object that will receive this message. Move up or down in the menu with the arrow keys, and accept the highlighted selector by pressing the tab key. If the selector has more than one part, all of the parts are entered into the workspace.

Development tools provided on the workspace menu, accessed by right-clicking in the workspace, are

- **Do it** Execute the selected text or the current line otherwise
- **Print it** Execute text and display the result.
- **Inspect it** Open an Inspector on a selected class name, or execute selected text and open an Inspector on the results.
- **Explore it** Open an object Explorer on a selected class name, or execute selected text and open an Explorer on the results.

- **Debug it** Open a Debugger on selected code.
- **Profile it** Open a Profiler on selected code.

Development tools accessible in the Extended search menu are the following.

- **browse it**
- **senders of it**
- **references to it**
- **selectors containing it**
- **method strings with it**
- **method source with it**
- **class names containing it**
- **class comments with it**
- **change sets with it**

Each one opens the tool specified on the text selected or on an object that that text names, if any. If there is none, a message may appear saying so.

Some of the functions of these development tools can be discovered through exploration, clicking on every button to see what happens. For further details see Pharo Smalltalk documentation.

## 4.1 Script by the example

**WARNING:** THE FOLLOWING SECTIONS ARE OBSOLETE AND NEED TO BE SYNCHRONISED WITH THE FRENCH USER GUIDE

There are two phases to using scripts:

- creating the script
- inserting the script in a sketch one or more times with different parameters

The tool to create or edit a script is on the menu Numeric→Edit a script. It is also reachable from the toolbar.

The tool to use a script is on the menu Numeric→Use a script. It is also reachable from the toolbar.

A script can be defined to receive 0 or more arguments (input parameters). After selecting a script to insert in the sketch, the user clicks on the items used as arguments. Then a final click somewhere in the background sets the location for the returned value.

In the following section, we present a few script examples so that the function and power of scripting will be more easily understood. The script and the macro construction give a special dimension to Dr.Geo – with a different positioning<sup>3</sup> – to let the user go into areas that the software was not initially planned for.

It is important to understand that most Pharo Smalltalk resources are accessible in scripts. This is particularly true for the class and method libraries<sup>4</sup>, which we will use a lot.

### 4.1.1 Script without parameter

<sup>3</sup> A macro construction is geometry oriented whereas a script is numerically/computation oriented.

<sup>4</sup> For example, mathematical functions.

## First example:

The procedure to create a script without an input parameter is as follows:

### 1. Edit the script

1. Select Numeric→Edit a script in the menu to open the script editor:

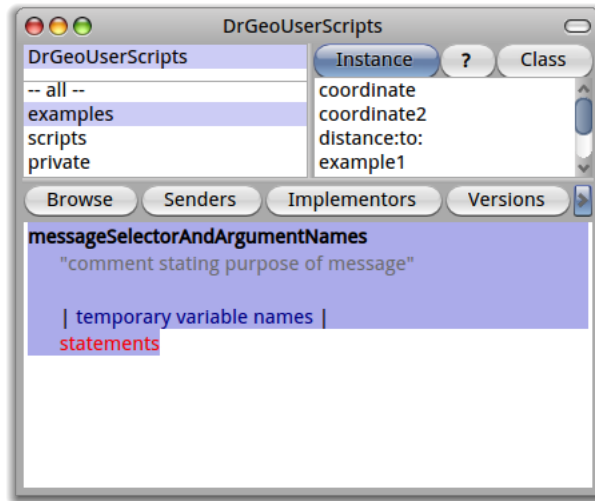


Figure 4.1: Script editor

Within the script editor window there are three panes:

- In the top left, the script categories to keep in good order: `examples`, `private`, and `scripts`. It is in this last one that you put your scripts.
  - In the top right, the script names in the selected category. When selecting one, its source code is displayed in the editor pane at the bottom.
  - In the bottom, the source code editor for the selected script. It is there you create or modify the scripts. To accept a modification, press the keyboard keys `Ctrl-s`.
2. In the script editor, select the category `scripts`. The text editor in the bottom will display a script source code template (See [scriptEditor], page 34).
  3. Input the source code as below (See [firstScript], page 35):

```
myFirstScript
"Hello, World is the traditional first program in any language.
It simply displays a text message."
^ 'Hello, World!'
```

Save the script with `Ctrl-s`. Dr.Geo will ask your last name and first name to track the history of the script modifications. The first script line, `myFirstScript` is the name of the script, followed by the source code. It is followed by an optional comment of one or more lines between quotation marks. The comment should usually explain the purpose of the script, what are the expected parameters, and what are the conditions for its use. We strongly encourage you to document carefully to prevent confusion and error.



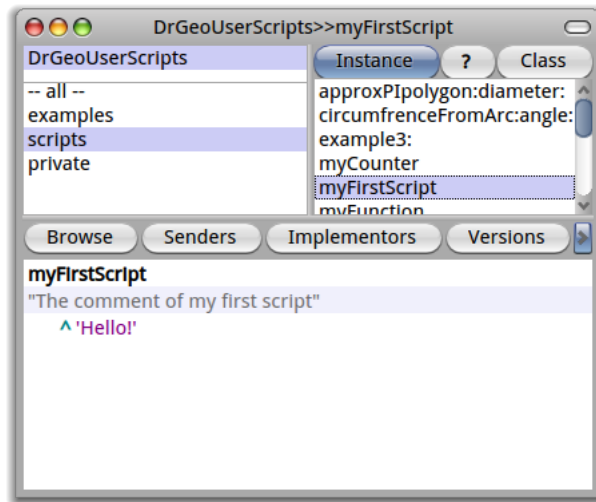


Figure 4.2: My first script

The script editor can now be closed.

## 2. Using the script in the sketch

Select Numeric→Use a script in the menu. In the displayed dialogue box, choose the script `myFirstScript`. Note: each time a script is selected, its descriptive comment is displayed in the bottom of the dialogue.

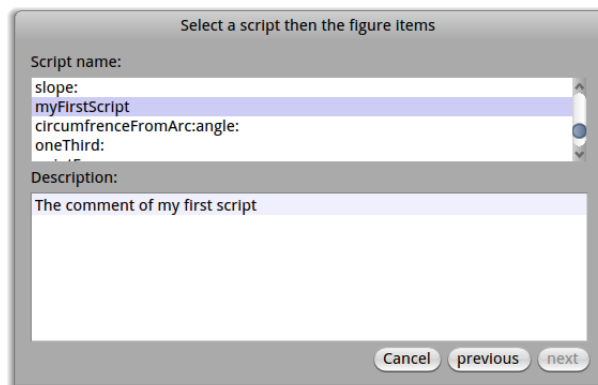


Figure 4.3: Select a script

Once the script selected, click on the sketch at the place to pin it. In this example the script only returns the message “ Hello, World ! ”. The returned value is always printed in the sketch. If necessary it is converted to a text representation.

In the following examples, we only give the source code of the Smalltalk scripts. To use them in Dr.Geo, just follow the steps explained previously.

## A random number generator and more:

If you need a simple random generator for numbers between 1 and 10, the following script exactly does that:

```
random
"I return a random number continuously updated"
~ 10 atRandom.
```

At each sketch update, it generates a different random number in the interval  $[1 ; 10]$ . Updates occur when the sketch is changed by dragging an object.

In case you prefer a random decimal number in the interval  $[0 ; 1]$ , use this script:

```
random2
"I return a random decimal number continuously updated from 0 to 1"
  ^ (101 atRandom -1) / 100.
```

Some details:

- The value returned by the script is the result of the expression after the `^`.
- The returned value can be of any class. Dr.Geo prints its text representation (a string).
- To return the value of a variable, it is enough to put its name after the symbol `^`.

## Calculate common values

To access an approximate value of  $\pi$  with the maximum precision available in Smalltalk:

```
pi
  ^ Float pi
```

This returns 3.141592653589793.

or  $e$  :

```
e
  ^ Float e
```

This returns 2.718281828459045.

These returned values are usable in the same way as any other value generated by Dr.Geo. Even for such small things, the script is your friend. But it can do much more interesting things when it receives input parameters.

Indeed, so far the scripts have had no arguments, so that it was not necessary to select items in the sketch when inserting the script in the construction. Of course the real interest in scripts comes from returning the values of numerical computation, pinned in the sketch for use with other constructions or scripts. In the following sections we show the use of scripts in a cascade.

### 4.1.2 Script with one input parameter

The procedure to create a script with one input parameter is mostly the same:

1. When editing the script (Numeric→Edit a script), the argument is inserted in the first line with the script name.

For example to calculate the distance from the origin to a point, we write the following script:

```
distanceToOrigin: item
"Return the distance from origin to a point"
  ^ item point dist: 0@0
```

A few explanations:

- `item` is the argument of our script, which must be a point. It is in fact an instance of the class `DrGPointItem`<sup>5</sup>

<sup>5</sup> To learn about the protocol of this class, write its name in a workspace, select it with the mouse then press the keys `Ctrl-b`. A class browser opens on this class to navigate in its protocol and source code.

- `DrGPointItem` has an instance method `#point` returning its coordinates. Thus from the argument we can extract information, here its coordinates, and calculate with it.
  - `O@O` is an instance of the class `Point` with coordinates `(0,0)`.
  - `#dist:` is a keyword message<sup>6</sup> from the class `Point` expecting as its unique argument another instance of `Point`. It calculates the distance between these two instances. It can be understood as: “distance between item point and `(0,0)`”. The keyword message syntax is very specific to Smalltalk. The arguments are declared in the message name line of a method.
2. To use a script (Numeric→Use a script), Dr.Geo expects from the user to click on appropriate objects, then somewhere in the sketch.

**Attention:** If an item other than a point is selected when a point is expected, Dr.Geo throws an error, opening a debugger window stating the problem and offering further information. The debugger window can be safely closed. To continue, select the script again.

Depending on the type of argument received by the script, various methods are available: to get its value, its coordinate, etc. A list of the methods you can use is presented below (See [\[api-dgs\]](#), page 40).

### 4.1.3 Script with two input parameters:

Let’s say we want to calculate the distance between two points. To do so we create a script with two input parameters inserted on the name line of the script. We name it `distance:to:`, where each “:” indicates the place of the parameter. Therefore in the script editor we write the following source code:

```
distance: item1 to: item2
"Calculate the distance between two points"
^ item1 point dist: item2 point
```

`item1` and `item2` are the names of the two parameters, which can be chosen freely. To use this script, proceed as in the previous example: select two points in the construction and click somewhere in the sketch to pin the result of the script.

### 4.1.4 Detailed example with several scripts

In the following section, we present a more complex sketch, integrating a cascading use of scripts to construct the curve of a function and its tangent at a mobile point of the curve.

The final sketch is in the Dr.Geo folder `examples`. It is named `Curve` and `slope.fgeo`.

---

<sup>6</sup> As we can see because its name ends in “:”

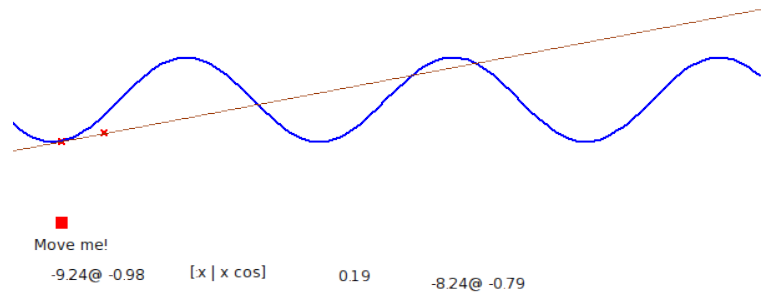


Figure 4.4: Curve and tangent to a point

In a new empty sketch, we construct a horizontal segment, then add to this segment a free point named “Move me!”. This point will be the base to construct the curve as a locus.

## Define a function: A script can return any type of object,

so that our first construction simply defines a function. To do so, we use a Smalltalk object named block of code (an anonymous function in Scheme and some other languages). We name this script `myFunction`, without parameter:

```
myFunction
  "Definition of our function"
  ^ [:x | x cos]
```

Next we insert the function in the sketch<sup>7</sup> So the block of code returned by `myFunction` expects one argument `:x` and it returns the cosine of that argument. We see later how to manipulate this script.

## Image of a value by function:

Now we calculate the coordinates of a point on the curve. We use our point “Move me!” abscissa and our function previously defined. Our script will have one unique point argument; as we will see we do not need to pass `myFunction` as argument:

```
pointM: item
  "Return a point on the curve of myFunction and driven by item point"
  ^ item point x @(self myFunction value: item point x)
```

The point returned by this script has the same abscissa as the argument, and its ordinate is the image of that abscissa by the function.

Note:

- the direct access to the function: `self myFunction`,
- the way to pass an argument to this block of code, which can be understood as `myFunction(item point x)`.

<sup>7</sup> It is important to add it to the construction to get it included in the file when the sketch is saved.

Now we use this script `pointM:` with the point “Move me!” as argument; the result of this script is of the form `1.2@0.5` representing a pair of coordinates.

With the tool `Coordinates`, `Points→Coordinates`, we create a point with its coordinates specified by the result of this script.

The locus tool `Locus`, `Lines→Locus`, gives the curve after selecting our two points.

## Slope at a point of the function’s curve:

We calculate an approximation of the slope at a given point thus:

$$p = (f(x + 0.001) - f(x)) / 0.001$$

This is directly translated in the script `slope:` with a single argument, the point at which to approximate the slope:

```
slope: item
"Compute the slope at the given x point position for myFunction"
| f x |
  f := self myFunction.
  x := item point x.
  ^ (f value: x + 0.001) - (f value: x) / 0.001
```

Next we insert this script in the construction.

Note:

- The declaration of temporary variables `| f x |`. As explained variables are not typed, so only the name is required.
- A reference to a block of code in a variable `f := self myFunction`. The symbol to assign a value to a variable is “:=”.
- The parenthesis! Smalltalk has no concept of operator priority; in fact operators do not exist in this language. The reader is encouraged to study the section about Smalltalk messages in the book *Pharo By Example*.

## Calculate and display the tangent to a curve:

To construct our tangent we need two points: one on the curve – we have it already – and a second one on the line. For this last one we use the previously calculated slope. Let’s write a second script computing this second point’s coordinates:

```
pointN: item
"Given an initial point, calculate the coordinates
of a point on the tangent"
| p x |
  p := self slope: item.
  x := item point x + 1.
  ^ x (item point y + p)
```

With the protocol of the class `Point` – message `+ sent to a point` – this script can be written in one line:

```
pointN: item
"Given an initial point, calculate the coordinates
of a point on the tangent"
  ^ item point + (1@ (self slope: item))
```

We use this script with argument the point on the curve. We get as result the coordinates of the second point on the tangent. With these inputs we construct the second point<sup>8</sup>. The tangent is the line defined by this point and the point on the curve.

<sup>8</sup> Tool `Points→Coordinates`

When moving the point “Move me!”, the tangent is recomputed. Equally interesting: modifying the script `myFunction` updates the whole construction accordingly. A few examples of modifications to this script:

```

^ [:x | x * x / 10]

^ [:x | x cos + (10 * x) sin]

^ [:x | (x * 5) cos + x abs]

```

## 4.2 Reference methods for Dr.Geo scripts

An argument passed to a script is always a reference to an instance of a class from the hierarchy `DrGMathItem`, the base class of all items used in a construction. So to know about the messages you can send to an argument passed to a script, it is wise to examine the hierarchy of `DrGMathItem`. It contains more than 80 classes, but because of class inheritance, only a few classes are interesting to explore:

- `DrGMathItem`
- `DrGpointItem`
- `DrGDirectionItem` for segment, line, ray, vector
- `DrGArcItem`
- `DrGCircleItem`
- `DrGLocus2ptsItem`
- `DrGPolygonNptsItem`
- `DrGValueItem`

To explore these classes, open a workspace – *Ctrl-k* after a click in the background of the environment – input and mouse select the class name to explore then press the shortcut *Ctrl-b* to open a class browser on it.

The following sections contain descriptions of some useful messages, ordered by class.

### 4.2.1 Math Item

The protocol of the class `DrGMathItem` concerns all types of argument passed to a script.

```

<String> safeName [Method on DrGMathItem]
⇒ text representing the item name

    name := point1 safeName.
    ^ name asUppercase.

```

To use this method in a script, create a Dr.Geo math item and give it a name with the style function. Then create the script

```

myName: item
^item safeName

```

and use it as usual, clicking on the point and then a place to put the result. Note that changing the name of the item and then forcing an update, for example by moving the item, updates the result of the script.

```

<Boolean> exist [Method on DrGMathItem]
⇒ boolean indicating whether the item exists in the current state of the sketch

    line exist ifTrue: [ position := line origin ].

```

**<Collection> parents** [Method on DrGMathItem]  
 ⇒ collection of the parents of this item  
`point1 := segment parents first.`

In this example, the parents of a segment joining two points are returned as an array containing the two points. In contrast, the parents of a free point are returned as nil.

**move: vector** [Method on DrGMathItem]  
*vector*, a vector (x,y) representing the displacement  
 Move an item in a given direction, if possible while respecting its constraints.  
`circle move: 2@1.`

**<Point> closestPointTo: aPoint** [Method on DrGMathItem]  
*aPoint*, coordinates ⇒ coordinates of the point on “curve” the closest to “aPoint”.  
 This works when the curve is a line, a segment, a vector, an arc, a circle, a polygon, or a locus.  
`position := segment closestPointTo: 2@1.`  
`[..]`  
`position := arc closestPointTo: 2@1.`

## 4.2.2 Point

A point item – point object defined in a Dr.Geo construction – passed as argument to a script is a complex object. It can be a free point on the plane, on a line, an intersection, etc. A few methods can be useful in scripts.

**<Point> point** [Method on DrGPointItem]  
 ⇒ coordinates of this item  
`abscissa := pointItem point x.`

For example, with this script

```
showOrdinate: aDrGeoPoint
"ordinate of point"
^ aDrGeoPoint point y
```

The user would invoke this script, pick a point, and pin the result in the sketch showing the ordinate of the selected point.

**point: aPoint** [Method on DrGPointItem]  
*aPoint*, coordinates  
 ⇒ modify the coordinate of “item”  
`pointItem point: 5@2.`

**<Float> abscissa** [Method on DrGPointOnCurveItem]  
 ⇒ curvilinear abscissa of this point, in the interval [0 ; 1]  
`a := pointItem abscissa.`

**abscissa: a** [Method on DrGPointOnCurveItem]  
*a*, decimal number in the interval [0 ; 1]  
 ⇒ modify the curvilinear abscissa of a free point on a line  
`pointItem abscissa: 0.5.`

**moveAt: aPoint** [Method on DrGPointItem]  
*aPoint*, coordinates where to displace “item”  
 ⇒ displace “item” to the given position  
`point moveAt: 2@1.`

### 4.2.3 Straight or curved line

<Float> *abscissaOf: aPoint* [Method on DrGCurveItem]  
*aPoint*, coordinates (x,y) of a point

⇒ number in [0 ; 1] curvilinear abscissa of “aPoint” on “curve”

`a := curve abscissaOf: 2@1.`

<Point> *pointAt: anAbscissa* [Method on DrGCurveItem]  
*anAbscissa*, number in [0 ; 1]

⇒ coordinates of a point on “curve” with curvilinear abscissa equal to “anAbscissa”

`myPoint := curve pointAt: 0.5.`

<Boolean> *contains: aPoint* [Method on DrGCurveItem]  
*aPoint*, coordinates (x, y) of a point

⇒ boolean indicating if the “aPoint” is on “curve”

`(curve contains: 0@1) ifTrue: [^ 'Yes!']  
 ifFalse: [^ 'No!'].`

### 4.2.4 Line, ray, segment, vector

<Point> *origin* [Method on DrGDirectionItem]

⇒ coordinates of the origin of this item

`item origin.`

<Vector> *direction* [Method on DrGDirectionItem]

⇒ vector (x, y) indicating the item direction

`v := item direction.  
 slope := v y / v x.`

<Vector> *normal* [Method on DrGDirectionItem]

⇒ unit vector normal to the item direction

`n := item normal.`

### 4.2.5 Segment

<Float> *item length* [Method on DrGSegmentItem]

⇒ length of the segment

`segment := canvas segment: 0@0 to: 5@5.  
 l := segment length`

<Point> *extremity1* [Method on DrGSegmentItem]

⇒ coordinates of the extremity 1

`segment := canvas segment: 0@0 to: 5@5.  
 p := segment extremity1.`

<Point> *extremity2* [Method on DrGSegmentItem]

⇒ coordinates of the extremity 2

`segment := canvas segment: 0@0 to: 5@5.  
 p := segment extremity2`

<Point> *middle* [Method on DrGSegmentItem]

⇒ coordinates of the middle of the segment

`segment := canvas segment: 0@0 to: 5@5.  
 m := segment middle`



### 4.2.6 Circle, arc, polygon

<Point> center [Method on DrGCircleItem|DrGArcItem]  
 ⇒ coordinates of the centre of the circle or arc

c := item center.

<Float> radius [Method on DrGCircleItem|DrGArcItem]  
 ⇒ radius of the circle or arc

r := item radius

<Float> length [Method on DrGCircleItem|DrGArcItem|DrGPolygonItem]  
 ⇒ length of the circle, arc or polygon

l := arc length

### 4.2.7 Value

<Float> valueItem [Method on DrGValueItem]  
 ⇒ value of this item

n1 := item2 valueItem.

n2 := item2 valueItem.

n1 + n2.

valueItem: v [Method on DrGValueItem]

v, decimal number

⇒ modify the value of a free value item

item valueItem: 5.2.

position: aPoint [Method on DrGValueItem]

aPoint, coordinates (x, y) of a point

⇒ displace at the screen the position of a value item

item position: 0.5@2.

### 4.2.8 Angle

<Integer> degreeAngle [Method on DrGAngleItem]  
 ⇒ measure of this angle item in degrees

angle1 := a1 degreeAngle.

<Float> radianAngle [Method on DrGAngleItem]

⇒ measure of this angle item in radians

angle1 := a1 radianAngle.

## 5 Dr.Geo Smalltalk sketch

*Dr.Geo Smalltalk sketches* – (DSS) – are sketches entirely defined in the Smalltalk language. This is not about constructing a sketch with the Dr.Geo graphical interface, but about describing a sketch with the Smalltalk language. We provide a nice programming interface with an easy and light syntax.

### 5.1 Smalltalk sketches by example

Smalltalk itself is a high level language, carefully crafted iteratively for about 10 years at Xerox Parc Research Labs. When a sketch is defined with it, we can use all the power of the language to build a sketch, or to position some objects randomly to get a slightly different sketch at each execution of its Smalltalk code. Therefore, a Smalltalk sketch is freed from the constraints of the graphic user interface while reinforced by the Smalltalk language.

A Smalltalk sketch is source code to execute in a **workspace**, which is a text editor where source code can be written and executed. To open one, use the shortcut **Ctrl-k** when no window is selected – or click on the background and in the menu select Tools→Workspace. It is possible to paste text in with the shortcut **Ctrl-v**. (See [workspace], page 74 to learn more about this tool).

Smalltalk sketches are not saved within Dr.Geo. They must be pasted to an external text editor and saved to external files in order to be available for reuse in Dr.Geo. Then the text must be copied from the external file and pasted into a Dr.Geo workspace in order to be executed again.

To introduce the concepts of Smalltalk sketches, we will study a few examples. Each one will be written in a workspace and its source code will be selected with mouse, then executed with the shortcut **Ctrl-d** to *Do-it!*<sup>1</sup>.

Let's start with the simplest possible Smalltalk sketch:

```
DrGeoCanvas new
```

When executing it, it simply creates a new empty sketch. The Dr.Geo canvas displayed is simplified as there is no toolbar, only the menu bar.

A second example that begins to do something:

```
| c item |
c := DrGeoCanvas new.
item := c point: 1.2 @ -2.
item name: 'A'.
```

Here we define a sketch containing a free point A of coordinates (1,2 ; -2). An object is added to the construction by sending a message to the canvas, here **#point:** to create a free point given its coordinates. The result is a point object. We can also modify the point by sending a message to it. Here we rename it 'A'.

Let's continue with a third, more interesting example:

```
| c triangle ourRandom m n p |

triangle := [:p1 :p2 :p3 |
c segment: p1 to: p2.
c segment: p2 to: p3.
c segment: p3 to: p1].
```

---

<sup>1</sup> Alternatively, this is the entry to select in the contextual menu of the workspace.

```

ourRandom := [6 - 11 atRandom].

c := DrGeoCanvas new.
m := c point: ourRandom value @ 0.
n := c point: 5 @ 0.
p := c point: ourRandom value @ 3.
triangle value: m value: n value: p.

```

This example shows us three interesting things:

1. The introduction of more elaborate constructions, not initially implemented in Dr.Geo. Here we define a block of code `triangle` between the square brackets: given three points, construct a triangle. We can compare this to the macro construction method but with a different, programming oriented approach.
2. The definition of a block of code – `ourRandom` – to give us integer random number between -5 and 5. It is used for a random positioning of the points. Therefore each time the sketch is executed, it is slightly different.
3. Assigning the result of a construction – the object we got when sending a construction message to the canvas – is not mandatory. We do it when we need to keep a reference to the constructed object for later use. Here in the block of code `triangle`, we don't keep references to the constructed segments. However, concerning the defined points, we keep references in the local variables `m`, `n` and `p`, for later use as arguments when executing the block of code `triangle`.

To finish with this introduction by example, here is the last one:

```

| c a b d |

c := DrGeoCanvas new.
a := c point: 1@0.
b := c point: 5@0.
d := c line: a to: b.
a color: Color yellow;
  round;
  large.
b hide.
d dashed.

```

Two points and a line are constructed. Then messages are sent to them to modify their styles, including hiding one.

We have finished our small guided tour of *Dr.Geo Smalltalk sketches*. In the following sections, we expose the commands available.

## 5.2 Reference methods for the Dr.Geo Smalltalk sketch

To add an object to a construction, you send a message to the canvas. The resulting constructed object can be modified as well by sending messages to it. So before adding any object to a canvas, we need to create the canvas with the command `DrGeoCanvas new`.

### 5.2.1 Various messages

`<DrGeoCanvas> new` [Method on DrGeoCanvas]  
 $\Rightarrow$  Return a canvas and open it. The result is normally assigned to a variable for later use to add constructions.

```
| canvas |
canvas := DrGeoCanvas new.
```

**fullscreen** [Method on DrGeoCanvas]

The sketch is set to fill the Dr.Geo window.

```
| canvas |
canvas := DrGeoCanvas new.
canvas fullscreen
```

**do: block** [Method on DrGeoCanvas]

*block*, Smalltalk block of code with instructions for construction and/or animation of the sketch.

Execute the block of code in a specific background process. Use it when a construction needs to be done step by step in front of the user or when the sketch is animated.

```
| canvas point |
canvas := DrGeoCanvas new.
canvas fullscreen.
point := canvas point: 0@0.
canvas do: [
  -5 to: 5 by: 0.1 do: [:x |
    point moveTo: x @ (x cos * 3).
    (Delay forMilliseconds: 100) wait.
    canvas update]
]
```

**update** [Method on DrGeoCanvas]

Update the canvas after some objects were modified. Used mainly for animation, or for modifications that do not cause an update themselves.

**gridOn** [Method on DrGeoCanvas]

Display the grid in the canvas.

**centerTo: aPoint** [Method on DrGeoCanvas]

*aPoint*, coordinates of a point

The canvas is displaced so the point given by the argument is at the centre of the canvas window.

```
canvas centerTo: 5@0
```

**scale: anInteger** [Method on DrGeoCanvas]

*anInteger*, scale of the canvas

Modify the scale of the canvas.

```
canvas scale: 10
```

## Point.

**<DrGeoWrappedPoint> point: aPoint** [Method on DrGeoCanvas]

*aPoint*, coordinates (x,y)

⇒ free point in the plane with coordinates *aPoint*.

```
canvas point: 5@2.
```

<DrGWrappedPoint> `pointX:Y: v1 v2` [Method on DrGeoCanvas]  
*v1*, value object  
*v2*, value object

⇒ point determined by its coordinates in the form of values placed in the sketch

```
canvas
  pointX: (canvas freeValue: 2) hide
  Y: (canvas freeValue: 5) hide.
```

<DrGWrappedPoint> `pointOnCurve:at: curve` [Method on DrGeoCanvas]  
*abscissa*  
*curve*, line (straight line, ray, segment, etc.)

*abscissa*, curvilinear abscissa of the free point, abscissa in interval [0 ; 1]

⇒ free point on a line

```
myPoint := canvas pointOnCurve: s1 at: 0.5.
```

<DrGWrappedPoint> `middleOf:and: p1 p2` [Method on DrGeoCanvas]  
*p1*, point item or coordinates  
*p2*, point item or coordinates

⇒ middle of two points

```
| canvas a i |
canvas := DrGeoCanvas new.
a := canvas point: 1@1.
i := canvas middleOf: a and: 4@4.
```

<DrGWrappedPoint> `middleOf: s` [Method on DrGeoCanvas]  
*s*, segment

⇒ middle of a segment

```
canvas middleOf: s.
```

<DrGWrappedPoint> `intersectionOf:and: l1 l2` [Method on DrGeoCanvas]  
*l1*, line  
*l2*, line

⇒ point of intersection of two lines

```
canvas intersectionOf: line1 and: line2
```

<DrGWrappedPoint> `altIntersectionOf:and: l1 l2` [Method on DrGeoCanvas]  
*l1*, line  
*l2*, line

⇒ second point of intersection of two lines, when it exists

```
canvas altIntersectionOf: line and: circle.
```

<DrGWrappedPoint> `point:parents block item` [Method on DrGeoCanvas]  
*block*, block of code returning coordinates

*item*, math item

⇒ pointBlockItem whose coordinates are calculated with the block of code with “item” as argument

```

| figure s mobile c block |
figure := DrGeoCanvas new.
figure fullscreen.
s:=figure
  segment: (figure point: -5@0)
  to: (figure point: 5@0).
mobile := figure pointOnCurve: s at: 0.1.
block := [:mathItem | |x|
  x := mathItem point x.
  x (x * x * x / 25 - x)].
c := figure point: block parent: mobile.
figure locusOf: c when: mobile.

```

<DrGWrappedPoint> point:parents *block collection* [Method on DrGeoCanvas]  
*block*, block of code returning coordinates

*collection*, math item collection

⇒ point whose coordinates are calculated with the block of code with “collection” as argument

```

| figure a b d m p |
figure:=DrGeoCanvas new.
a:=figure point: (-2)@1.
b:=figure point: 3@3.
d:=figure line: a to: b.
d color: Color blue.
m:=figure point: 1@(-1).
p:= figure
  point: [:parents | parents first closestPointTo: parents second point]
  parents: {d . m}.

```

This example uses array notation { . } to provide the collection needed.

## Line.

<DrGWrappedCurve> line:to: *p1 p2* [Method on DrGeoCanvas]  
*p1*, point or coordinates

*p2*, point or coordinates

⇒ line passing through these two points

```

| canvas p1 |
canvas := DrGeoCanvas new.
p1 := canvas point: 0@0.
canvas line: p1 to: 1@2.

```

Note that if one or both arguments are given as literals, those points are not drawn, and are not available in the sketch to click and drag. The direction of the line thus cannot be changed interactively.

<DrGWrappedCurve> parallel:at: *d p* [Method on DrGeoCanvas]  
*d*, direction (line, segment, vector,...)

*p*, point or coordinates

⇒ line parallel to direction *d* and passing through point *p*

```
| canvas a d |
canvas := DrGeoCanvas new.
a := canvas point: 1@5.
d := canvas line: (-2)@1 to: 3@3.
canvas parallel: d at: a.
```

<DrGWrappedCurve> perpendicular:at: *d p* [Method on DrGeoCanvas]  
*p*, point or coordinates

*d*, direction (line, segment, vector, ...)

⇒ line perpendicular to direction *d* and passing through point *p*

```
canvas perpendicular: d at: 1@5.
```

<DrGWrappedCurve> perpendicularBisector: *s* [Method on DrGeoCanvas]  
*s*, segment

⇒ perpendicular bisector to *s*

```
canvas perpendicularBisector: (canvas segment: 0@0 to: 4@4)
```

<DrGWrappedCurve> perpendicularBisector:to: *a* [Method on DrGeoCanvas]  
*b*

*a*, point or coordinates

*b*, point or coordinates

⇒ perpendicular bisector to segment joining *a* and *b*

```
canvas perpendicularBisector: 0@0 to: 4@4
```

<DrGWrappedCurve> angleBisector: *a* [Method on DrGeoCanvas]  
*a*, geometric angle defined by **three points**

⇒ angle bisector of the angle *a*

```
canvas angleBisector: angle
```

<DrGWrappedCurve> [Method on DrGeoCanvas]  
angleBisectorSummit:side1:side2: *a b c*

*a,b,c*, points defining a geometric angle BAC

⇒ angle bisector of the angle BAC

```
canvas angleBisectorSummit: 0@0 side1: 1@0 side2: 0@1
```

## Ray.

<DrGWrappedCurve> ray:to: *o p* [Method on DrGeoCanvas]  
*o*, point or coordinate, the origin

*p*, point or coordinates, point anywhere on the ray

⇒ ray defined by its origin and a second point

```
| canvas a |
canvas := DrGeoCanvas new.
a := canvas point: 1@5.
canvas ray: 0@0 to: a.
```

**Segment.**

```
<DrGWrappedSegment> segment:to: p1 p2 [Method on DrGeoCanvas]
  p1, points or coordinates
  p2, points or coordinates
  ⇒ segment defined by two points
    | canvas a |
    canvas := DrGeoCanvas new.
    a := canvas point: 5@5.
    canvas segment: 10@10 to: a.
```

**Circle.**

```
<DrGWrappedFilledCircle> circleCenter:to: c p [Method on DrGeoCanvas]
  c, point or coordinates, centre of the circle
  p, point or coordinates, point on the circle
  ⇒ circle defined by its centre and a point
    | canvas a |
    canvas := DrGeoCanvas new.
    a := canvas point: 1@5.
    canvas circleCenter: a to: 10@4.
```

```
<DrGWrappedFilledCircle> circleCenter:radius: [Method on DrGeoCanvas]
  c r
  c, point or coordinates, centre of the circle
  r, numeric item or numeric value, radius
  ⇒ circle defined by its centre and radius
    | canvas a r |
    canvas := DrGeoCanvas new.
    a := canvas point: 1@5.
    r := canvas freeValue: 4.
    canvas circleCenter: a radius: r.
    canvas circleCenter: 4@4 radius: 5
```

**Arc.**

```
<DrGWrappedFinitCurve> arc:to:to: p1 p2 p3 [Method on DrGeoCanvas]
  p1, point or coordinates, 1st extremity of the arc
  p2, point or coordinates representing a point on arc
  p3, point or coordinates, 2nd extremity of the arc
  ⇒ arc defined by its extremities and a point
    | canvas a b |
    canvas := DrGeoCanvas new.
    a := canvas point: 1@5.
    b := canvas point: 0@5.
    canvas arc: a to: b to: -1 @ -2.
```

```
<DrGWrappedFinitCurve> arcCenter:from:to: o a [Method on DrGeoCanvas]
  b
  o, point or coordinates, centre of the arc
```



*a*, point or coordinates, origin of the arc  
*b*, point or coordinates, so the angle is AOB  
 ⇒ arc defined by its centre and angle AOB

```
| canvas a b |
canvas := DrGeoCanvas new.
a := canvas point: 1@5.
b := canvas point: 0@5.
canvas arcCenter: a from: b to: -1 @ -2.
```

## Polygon.

<DrGWrappedFinitCurve> *polygon: collection* [Method on DrGeoCanvas]  
*collection*, collection of points or coordinates; vertices of the polygon

⇒ polygon defined by its vertices

```
| canvas b |
canvas := DrGeoCanvas new.
b := canvas point: 1@3.
canvas polygon: {1@2. b. 0@0. d}
```

<DrGWrappedFinitCurve> [Method on DrGeoCanvas]  
*regularPolygonCenter:vertex:sides: c s n*

*c*, point or coordinates, centre of the polygon

*s*, point or coordinate, a vertex of the polygon

*n*, value item or numeric value, number of vertices of the polygon

⇒ regular polygon defined by its centre, one vertex and number of vertices

```
| canvas b |
canvas := DrGeoCanvas new.
b := canvas point: 1@3.
canvas regularPolygonCenter: b vertex: 1@1 sides: 7.
```

## Geometric transformations.

Geometric transformations are to transform any kind of geometric objects: point, segment, line, ray, vector, circle, arc and polygon.

<DrGWrappedCurve> *rotate:center:angle: i c a* [Method on DrGeoCanvas]  
*i*, object to transform (point, segment, line, ray, vector, circle, arc, polygon)

*c*, point or coordinates, rotation centre

*a*, value item or numeric value, rotation angle

⇒ transformed object

```
| canvas c k l |
canvas := DrGeoCanvas new.
c := canvas point: 5@5.
k := 3.14159.
l := canvas line: 0@0 to: 5@5.
canvas rotate: l center: c angle: k.
canvas rotate: l center: 0@0 angle: Float pi / 3.
```

<DrGWrappedCurve> scale:center:factor: *i c k* [Method on DrGeoCanvas]  
*i*, object to transform (point, segment, line, ray, vector, circle, arc, polygon)

*c*, point or coordinates, homothety centre

*k*, value item or numeric value, homothety factor

⇒ transformed object

```
| canvas c k l |
canvas := DrGeoCanvas new.
c := canvas point: 0@5.
k := -3.
l := canvas line: 0@0 to: 5@5.
canvas scale: l center: c factor: k.
canvas scale: l center: 0@4 factor: 5.
```

<DrGWrappedCurve> symmetry:center: *i c* [Method on DrGeoCanvas]  
*i*, point, segment, line, ray, vector, circle, arc, polygon

*c*, point or coordinates, symmetry centre

⇒ transformed object

```
| canvas a |
canvas := DrGeoCanvas new.
a := canvas point: 4@2.
canvas symmetry: a center: 0@0.
```

<DrGWrappedCurve> reflect:axe: *i a* [Method on DrGeoCanvas]  
*item*, point, segment, line, ray, vector, circle, arc, polygon

*axe*, line, symmetry axis

⇒ transformed object

```
| canvas a d polygon |
canvas := DrGeoCanvas new.
a := canvas polygon: {0@0 . 2@0 . 3@2 . 0@4}
d := canvas line: (-1@(-1) to: 4@(-1)).
canvas reflect: a axis: d
```

<DrGWrappedCurve> translate:vector: *i v* [Method on DrGeoCanvas]  
*item*, point, segment, line, ray, vector, circle, arc, polygon

*vector*, vector or coordinates

⇒ transformed object

```
| canvas u a |
canvas := DrGeoCanvas new.
u := canvas vector: (canvas point: 1@1) to: (canvas point: 3@2).
a := canvas translate: (canvas point: 2@1) vector: u.
```

```
| canvas u a |
canvas := DrGeoCanvas new.
a := canvas translate: (canvas point: 2@1) vector: 2@1.
```

## Locus of a point.

```
<DrGWrappedCurve> locusOf:when: m c [Method on DrGeoCanvas]
  m, mobile point on a line
  c, fixed point depending on the mobile point m
  => locus
      canvas locusOf: c when: mobile.
```

## Vector.

```
<DrGWrappedCurve>vector:to: o e [Method on DrGeoCanvas]
  o, point or coordinates, vector origin
  e, point or coordinates, vector extremity
  => vector
      | canvas b |
      canvas := DrGeoCanvas new.
      b := canvas point: 0@5.
      canvas vector: b to: -1 @ -2.
```

```
<DrGWrappedCurve> vector: p [Method on DrGeoCanvas]
  p, point or coordinates, the vector coordinates
  => vector from 0@0 to point
      | canvas p |
      canvas := DrGeoCanvas new.
      p := canvas point: 5@5.
      canvas vector: p.
      canvas vector: -5 @ -5
```

## Number.

When applying a method to a canvas that creates a numeric value, the value is placed as a Dr.Geo math item somewhat at random in the sketch, and is initially hidden. It can be revealed with the show command described below.

```
<Point> coordinates [Method on DrGWrappedPoint]
  pointItem, point
  => coordinates (static) of self
      | canvas c p segment |
      canvas := DrGeoCanvas new.
      segment := canvas segment: 0@0 to: 3@5
      p := canvas pointOnCurve: segment at: 0.5.
      c := p coordinates.
      canvas freeValue: c x
```

```
<DrGWrappedValue> abscissaOf: item [Method on DrGeoCanvas]
  item, point or vector item
  => abscissa (dynamic) item of item
      | canvas m x |
      canvas := DrGeoCanvas new.
      m := canvas middleOf: 10@5 and: 7@8.
      x := canvas abscissaOf: m.
```

The abscissa of a point is its x coordinate. The abscissa of a vector is the difference between the x coordinates of its initial and final points.

```
<DrGWrappedValue> ordinateOf: item [Method on DrGeoCanvas]
  item, point or vector item
⇒ ordinate (dynamic) item of item
  | canvas m x |
  canvas := DrGeoCanvas new.
  m := canvas middleOf: 10@5 and: 7@8.
  x := canvas ordinateOf: m
```

The ordinate of a point is its y coordinate. The ordinate of a vector is the difference between the y coordinates of its initial and final points.

```
<DrGWrappedValue> freeValue: v [Method on DrGeoCanvas]
  v, initial value
⇒ free value item, initially hidden, randomly placed
  v := canvas freeValue: (-1 arcCos).
  v show
```

```
value: aNumber [Method on DrGWrappedValue]
  item, a free value item
  aNumber, a number
Eedit the value of a free number item
  v := canvas freeValue: 3.
  v show.
  v value: Float pi
```

```
<DrGWrappedValue> lengthOf: item [Method on DrGeoCanvas]
  item, segment, circle, arc or vector
⇒ number item, the item length
  canvas lengthOf: v1.
```

```
<DrGWrappedValue> distance:to: item point [Method on DrGeoCanvas]
  item, line or point
  point, point
⇒ number, distance between two points or a point and a line
  canvas distance: l1 to: a.
```

```
<DrGWrappedValue> slopeOf: line [Method on DrGeoCanvas]
  line, line
⇒ number, slope of the line
  | canvas p |
  canvas := DrGeoCanvas new.
  p := canvas slopeOf: d.
```

Dr.Geo does not handle infinite slope for vertical lines well. If a slope is placed in a sketch, and then the line is made vertical, the slope disappears from the sketch, and its value does not update in the data pane.

**Angle.**

<DrGWrappedValue> angle:to:to: *a b c* [Method on DrGeoCanvas]

*a*, point

*b*, point, angle summit

*c*, point

⇒ geometric angle ABC in the interval  $[0 ; \pi]$

```
| canvas a b c angle |
canvas := DrGeoCanvas new.
a := canvas point: 0@0.
b := canvas point: 0@4.
c := canvas point: 3@0.
angle := canvas angle: b to: a to: c.
angle show.
```

<DrGWrappedValue> angle:to: *v1 v2* [Method on DrGeoCanvas]

*v1*, vector

*v2*, vector

⇒ oriented angle given two vectors in the interval  $]-\pi ; \pi]$

```
| canvas v1 v2 a b c angle |
canvas := DrGeoCanvas new.
a := canvas point: 0@0.
b := canvas point: 0@4.
c := canvas point: 3@0.
v1 := canvas vector: a to: b.
v2 := canvas vector: a to: c.
angle := canvas angle: v2 to: v1.
angle show.
```

**Equation.**

<DrGWrappedValue> equationOf: *item* [Method on DrGeoCanvas]

*item*, line or circle

⇒ equation of the line or circle

```
| canvas e d |
canvas := DrGeoCanvas new.
d := canvas line: 0@0 to: 15@13.
e := canvas equationOf: d.
e show.
```

**Text.**

<DrGWrappedText> text: *string* [Method on DrGeoCanvas]

*string*, a string

⇒ a text arbitrary positioned

```
canvas text: 'Hello'
```

<DrGWrappedText> text:at: *string point* [Method on DrGeoCanvas]

*string*, a string

*point*, coordinates

⇒ a text positioned at aPoint

```

    canvas text: 'Hello,
    I am happy!' at: 0@0

```

**text:** *aString* [Method on item]

*item*, an item representing a text

*aString*, a string

Edit the text of a text object

```

    myText := canvas text: 'Hello'.
    myText text: 'Bye'

```

## 5.2.2 Modification of object attributes

To modify object attributes, we send messages to the objects. So the attributes are always modified after creating the objects.

**color:** *aColor* [Method on DrGWrappedItem]

*aColor*, a Color, see methods in this class for existing colors : Color black, Color red, Color blue, Color orange, Color yellow,...

Modify the item color

```

    pointA color: Color green.

```

**backgroundColor:** *aColor* [Method on DrGWrappedText]

*aColor*, a Color

Modify the background color of the text item

```

    myText backgroundColor: Color green.

```

**name:** *aString* [Method on DrGWrappedItem]

*aString*, a string, text

Rename item

```

    segment name: '[AB]'.

```

**hide** [Method on DrGWrappedItem]

Hide an item

```

    point hide

```

**show** [Method on DrGWrappedItem]

Show an item

```

    value show

```

**small** [Method on DrGWrappedCurve]

Set thickness of line to small

```

    circle small.

```

**normal** [Method on DrGWrappedCurve]

Set thickness of line to normal

```

    arc normal.

```

**large** [Method on DrGWrappedCurve]

Set thickness of line to large

```

    polygon large.

```

**plain** [Method on DrGWrappedCurve]

Plain, undotted, style line

```

    polygon plain.

```

<b>dashed</b>	[Method on DrGWrappedCurve]
Dash style line	
polygon dashed.	
<b>dotted</b>	[Method on DrGWrappedCurve]
Dotted style line	
arc dotted.	
<b>cross</b>	[Method on DrGWrappedPoint]
Cross shape for point	
a cross.	
<b>round</b>	[Method on DrGWrappedPoint]
Round shape for point	
a round.	
<b>square</b>	[Method on DrGWrappedPoint]
Square shape for point	
a square.	
<b>small</b>	[Method on DrGWrappedPoint]
Small sized point	
a small.	
<b>medium</b>	[Method on DrGWrappedPoint]
Medium sized point	
a medium.	
<b>large</b>	[Method on DrGWrappedPoint]
Large sized point	
a large.	
<b>moveTo: <i>point</i></b>	[Method on DrGWrappedItem]
<i>point</i> , coordinates	
Move the item to the given position, if its constraints permit	
canvas a	
canvas := DrGeoCanvas new.	
a := canvas point: 0@0.	
a moveTo: 5@5.	
canvas update	

### 5.2.3 Complementary methods

The `DrGeoCanvas` class proposes in the category `helpers` complementary methods to ease the computation of complex, interactive sketches.

<b>plot:from:to: <i>block</i> <i>x0</i> <i>x1</i></b>	[Method on DrGeoCanvas]
<i>block</i> , block of code, with one argument, to describe a function on an interval	
<i>x0</i> , number, lower boundary of the function's domain	
<i>x1</i> , number, upper boundary of the function's domain	
Display the curve representing the function described by the block of code, in the interval [ <i>x0</i> ; <i>x1</i> ], and display a free point on the x axis together with a relative point on the curve, representing the value of the function at that point.	
canvas plot: [:x  x * x] from: -3 to: 3.	

```
<BlockClosure> float:at:from:to:name: f1 p f2 f4 [Method on DrGeoCanvas]
```

```
s
```

*f1*, initial value

*p*, left position of the ruler

*f2*, minimum value

*f3*, maximum value

*s*, name of the value

⇒ block of code returning the current value of the ruler

Construct a ruler at the given position with a **decimal** value in the interval [float2 ; float3]

```
| canvas A F |
canvas := DrGeoCanvas new.
A := canvas float: 1 at: -10@4 from: 0 to: 10 name: 'A'.
F := canvas integer: 3 at: -10@3 from: 0 to: 10 name: 'F' showValue: true.
A value + F value.
```

There are other variants, some for integer values.

## 5.3 Gallery of examples

To illustrate the use of Dr.Geo Smalltalk sketches, we present below a small set of examples. It shows you some possibilities that we hope will inspire you for your own needs. For each example for which we give the Smalltalk source code, we encourage you to copy and paste it into a Dr.Geo workspace and run the code.

### 5.3.1 Animate a figure

These examples rely on time handling and background processes.

We begin with a simple animation in order to understand the concept:

```
| figure p pause |
figure:=DrGeoCanvas new.
p := figure point: 0@0.
pause := Delay forSeconds: 0.2.
figure do: [
  100 timesRepeat: [
    p mathItem moveTo: (p mathItem point + (0.1@0)).
    figure update.
    pause wait]].
```

The Delay class supports setting time intervals for pausing Dr.Geo. The `do:` and `timeRepeat:` control structures repeat a code block that moves its argument item across the sketch. The update is necessary so that the point is displayed in its new position after each move.

A second example with a more elaborate sketch:

```
| figure s r u pause |
figure := DrGeoCanvas new fullscreen.
s := figure segment: 0@ -1 to: 4@ -1.
r := figure pointOnCurve: s at: 0.8.
u := figure segment: 0@0 to: 0@1.
u round small; color: Color blue.
1 to: 100 do: [:n]
```



```

u := figure
  point: [:parents | |y t|
    y := parents first point y.
    t := parents second point x.
    (n / 5) @ t * y * (1 - y)]
  parents: {u . r}.
u round small; color: Color blue].
pause := Delay forSeconds: 0.1.
figure do: [
  0 to: 1 by: 0.05 do: [:x |
    r mathItem setCurveAbscissa: x.
    figure update.
    pause wait]].

```

### 5.3.2 Sierpinski triangle

This example largely relies on a recursive block of code for drawing triangles and subdividing them to a specified number of levels.

```

| triangle c |
triangle := [:s1 :s2 :s3 :n |
  c segment: s1 to: s2;
  segment: s2 to: s3;
  segment: s3 to: s1.
  n > 0 ifTrue:
    [triangle
      value: s1
      value: (c middleOf: s1 and: s2) hide
      value: (c middleOf: s1 and: s3) hide
      value: n-1.
    triangle
      value: (c middleOf: s1 and: s2) hide
      value: s2
      value: (c middleOf: s2 and: s3) hide
      value: n-1.
    triangle
      value: (c middleOf: s1 and: s3) hide
      value: (c middleOf: s2 and: s3) hide
      value: s3
      value: n-1.]].

c := DrGeoCanvas new.
triangle
  value: (c point: 0 @ 3)
  value: (c point: 4 @ -3)
  value: (c point: -4 @ -3)
  value: 3.

```

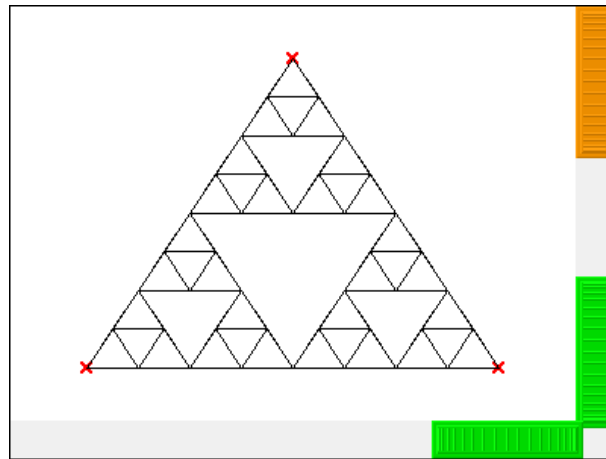


Figure 5.1: Triangle de Sierpinski

# Partie III

## Applications



## 6 Didactic applications

This chapter is an aid to studying Dr.Geo using didactic examples. Although in the previous chapters several examples were presented, here the approach is little more concrete, while still very original. This chapter was created using contributions from various cultural sources.

### 6.1 Perimeter and area

One possible didactic use of Dr.Geo is through its Smalltalk script system to resolve classic geometry exercises.

As an example, we will show the solution of the following classic problem – involving the Pythagorean theorem:

A right trapezoid ABCD where the bases and height are known. Calculate its perimeter and area.

**Solution:**

First we construct the Dr.Geo sketch as follows:

aire = 9.00

périmètre = 12.61

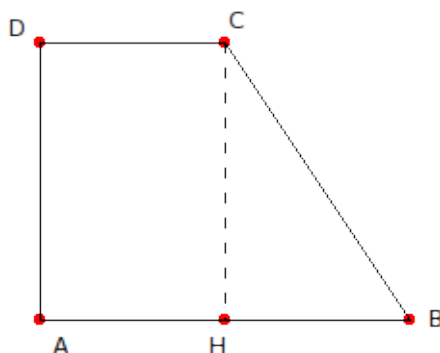


Figure 6.1: Right trapezoid

The sketch contains the data to use to resolve the problem. First we can answer the question of the area. To do so we write a Smalltalk script with arguments the two bases and the height:

```
areaTrapezoidBase1: b1 base2: b2 height: h
```

```
"Calculate the area of a right trapezoid given  
its bases and one height"
```

```
^ h length * (b1 length + b2 length) / 2
```

To calculate the perimeter, we write a script where we calculate the length of BC with the Pythagorean theorem:

```
perimeterTrapezoidBase1: b1 base2: b2 height: h
```

```
"Calculate the area of a right trapezoid given  
its two bases and one height"
```

```
| hb bc |
```

```
hb := (b2 length - b1 length) abs.
```

```
bc := (hb squared + h length squared) sqrt.
```

```
^ b1 length + b2 length + h length + bc
```

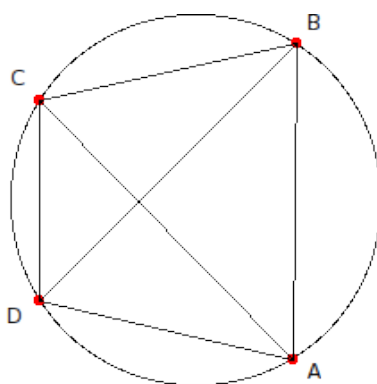
It is not difficult, if you follow the same model, to develop other similar examples.

## 6.2 Theorem and conjectures

With Smalltalk scripts one can solve exercises, but can also understand theorems more deeply and verify conjectures. In this section we start to analyse Ptolemy's theorem:

Given an inscribed quadrilateral, the sum of the products of its opposite sides is equal to the product of its diagonals.

We construct the sketch as below where we implemented two scripts to calculate respectively the sum of the products of its opposite sides and the product of its diagonals.



$$(AD * BC) + (BC * AD) = 31.92$$

$$AC * BD = 31.92$$

Figure 6.2: Ptolemy's theorem: inscribed quadrilateral

The first script:

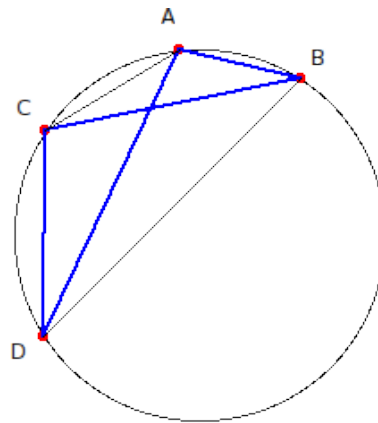
```
ptolemySumS1: ab s2: bc s3: cd s4: ad
"Select the four consecutive sides of the quadrilateral,
then the location for the result"
^ (ad length * bc length) + (ab length * cd length)
```

The second script:

```
ptolemyProductD1: ac d2: bd
"Select the two diagonals of the quadrilateral,
then the location for the result"
^ ac length * bd length
```

As we can see, the values returned by the scripts, as stated by Ptolemy's theorem, are equal<sup>1</sup> When we dynamically modify the sketch, the script values are always equal, except in the following situation where the quadrilateral is not convex:

<sup>1</sup> This is only a numerical example, not a proof.



$$(AD * BC) + (BC * AD) = 26.13$$

$$AC * BD = 13.68$$

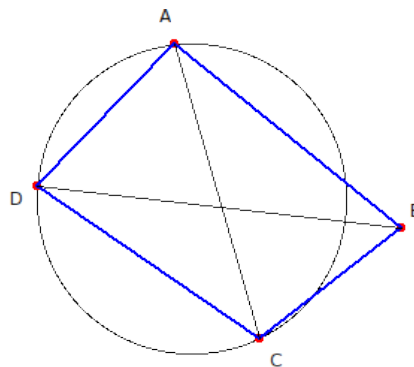
Figure 6.3: Ptolemy's theorem: non convex inscribed quadrilateral

In this case the theorem is not true and the previous text of the theorem is inaccurate. It should be reformulated as follow:

Given an inscribed CONVEX quadrilateral, the sum of the products of its opposite sides is equal to the product of its diagonals.

Now a additional conjecture can be stated: is the theorem still valid when the convex quadrilateral is **non inscribed**?

With Dr.Geo we verify this conjecture is false with the following counterexample bellow. To build this counterexample, we have just detached the point B from the circle by dragging it with the touch *Shift* pressed at the same time.



$$(AD * BC) + (BC * AD) = 39.76$$

$$AC * BD = 39.09$$

Figure 6.4: Counterexample of the conjecture

The reader will easily use Dr.Geo to construct additional didactic examples, perhaps more famous, relative to the Pythagorean and Euclidean theorems.

### 6.3 Irrational numbers

A classic construction of irrational numbers, known as the Teodoro spiral, gives the geometric construction of integer square roots. It begins with an isosceles right triangle.

Let's start with the triangle OAB where  $OA=AB=1$ :

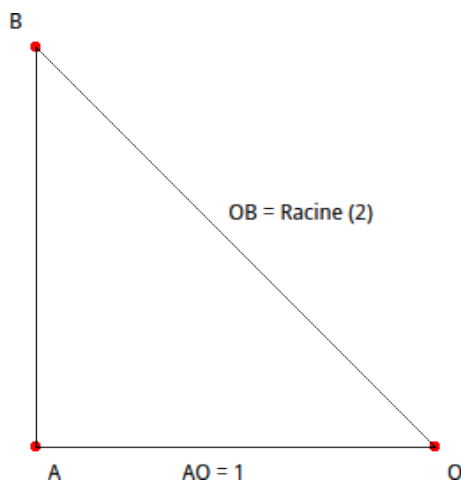


Figure 6.5: Construction of the square root of 2

Using the Pythagorean theorem we have OB equal to the square root of 2. Now, with the sketch, we construct another right triangle with the sides OB and BC so that  $BC=1$ .

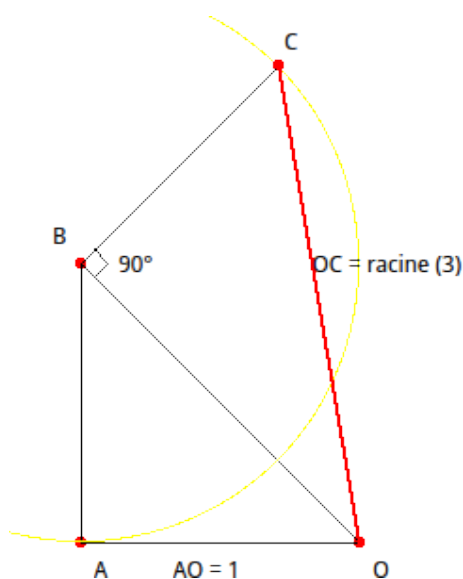


Figure 6.6: Construction of the square root of 3

Still using the Pythagorean theorem, it is clear the hypotenuse OC of OBC has a length equal to the square root of 3. The process can be repeated without limit to get the square roots of all positive integers.

The iterative nature of this construction can be naturally represented in a Smalltalk sketch.



Let's consider the following Smalltalk sketch source code:

```
| sketch triangle |
sketch := DrGeoCanvas new fullscreen.
triangle := [:p1 :p2 :p3 :n |
  |s1 s2 s3 perp circle p4 |
  s1 := sketch segment: p1 to: p2.
  s2 := (sketch segment: p2 to: p3) color: Color red.
  s3 := sketch segment: p3 to: p1.
  perp := (sketch perpendicular: s3 at: p3) hide.
  circle := (sketch circleCenter: p3 to: p2) hide.
  p4 := (sketch altIntersectionOf: circle and: perp) hide.
  n > 0 ifTrue: [triangle value: p1 value: p3 value: p4 value: n -1]].

triangle
  value: 0@0
  value: -1@0
  value: -1@1
  value: 50
```

The triangle at the beginning is defined by the coordinates of its vertices. The source code is a direct transcription of the logic of the construction, integrating its iterative nature with a recursive loop. We use the `#hide` message several times to mask intermediate constructions, in order not to overload the user's perception with background constructions. Once executed, Dr.Geo gives the following sketch.

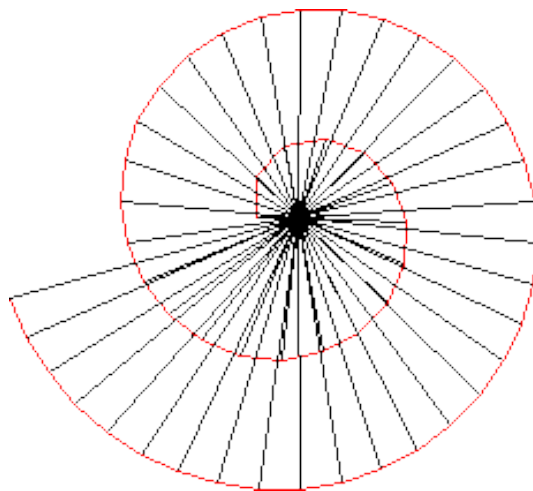


Figure 6.7: Teodoro spiral

The hypotenuse length of each triangle is the square root of an integer in the interval  $[2 ; 52]$ .

The same spiral with intermediate items shown exposes how difficult it will be to *hand construct* such a sketch:

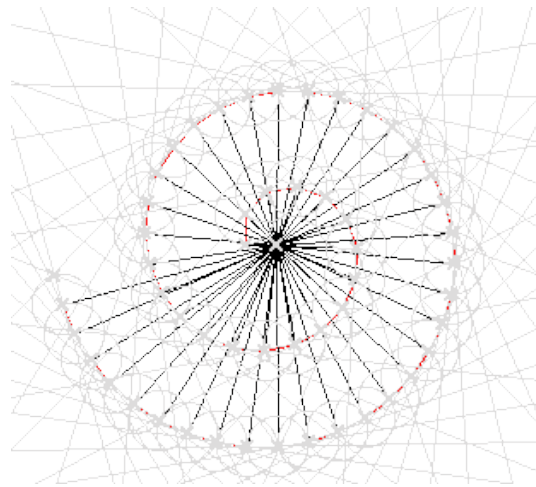


Figure 6.8: Teodoro spiral with hidden item revealed

As a bonus, here is an animated version of the previous Smalltalk sketch. To do this we send to the canvas the messages `#do:` and `#update` (See [\[SmalltalkSketchMethods\]](#), page 45 for more reading about it).

Note the use of the class `Delay` to slow down the construction:

```
| sketch triangle delay |
sketch := DrGeoCanvas new fullscreen.
triangle := [:p1 :p2 :p3 :n |
  |s1 s2 s3 perp circle p4 |
  s1 := sketch segment: p1 to: p2.
  s2 := (sketch segment: p2 to: p3) color: Color red.
  s3 := sketch segment: p3 to: p1.
  perp := sketch perpendicular: s3 at: p3.
  circle := sketch circleCenter: p3 to: p2.
  p4 := sketch altIntersectionOf: circle and: perp.
  sketch update.
  (Delay forMilliseconds: 200) wait.
  perp hide. circle hide. p4 hide.
  n > 0 ifTrue: [triangle value: p1 value: p3 value: p4 value: n -1]].

sketch do: [triangle value: 0@0 value: -1@0 value: -1@1 value: 50]
```

## 6.4 Baravelle spiral

As we saw previously, in Smalltalk sketches it is easy to construct, intuitively and simply, sketches *to visualise* recursive or iterative situations in programming.

We can go one step further by modifying the previous Smalltalk code – used to construct irrational numbers – to get a famous sketch of the mathematics literature: the Baravelle spiral constructed from similar equilateral right triangles.

The code to construct this spiral is as follows:

```
| sketch triangle |
sketch := DrGeoCanvas new fullscreen.
triangle := [:p1 :p2 :p3 :n |
```

```

|s1 s2 s3 m perp circle p4 |
s1 := sketch segment: p1 to: p2.
s2 := sketch segment: p2 to: p3.
s3 := sketch segment: p3 to: p1.
m := (sketch middleOf: p1 and: p3) hide.
perp := (sketch perpendicular: s3 at: p3) hide.
circle := (sketch
  circleCenter: p3
  radius: (sketch distance: m to: p3) hide) hide.
p4 := (sketch altIntersectionOf: circle and: perp) hide.
n > 0 ifTrue: [triangle value: m value: p3 value: p4 value: n -1]
].

triangle
  value: (sketch point: 0@5)
  value: (sketch point: 5@5)
  value: (sketch point: 5@0)
  value: 9.
triangle
  value: (sketch point: 0@ -5)
  value: (sketch point: -5@ -5)
  value: (sketch point: -5@0)
  value: 9

```

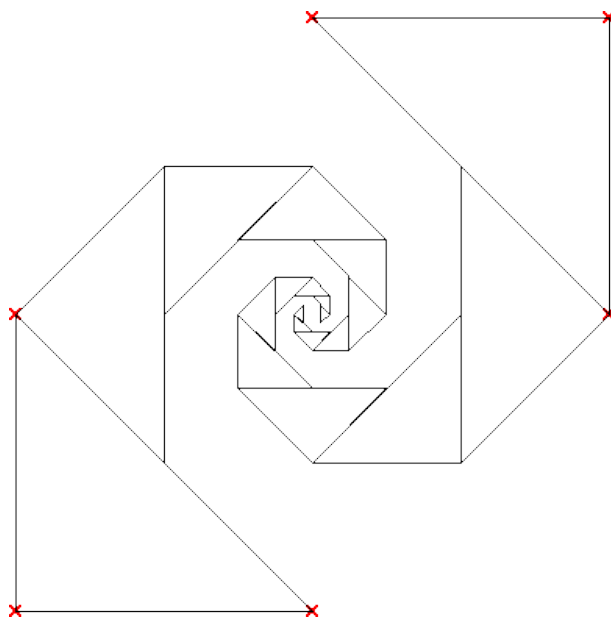


Figure 6.9: The Baravelle spiral after executing the Smalltalk code

With this sketch and the corresponding Smalltalk code we clearly perceive the recursive nature of the construction mechanism. An interesting problem for the reader is to establish when the two branches of the spiral converge.

## 6.5 Pappus Chain

Another classic use of a Smalltalk programmed sketch is based on its numeric ability to reproduce a geometric sketch knowing its analytic characteristics.

The example we propose is the famous “Pappus Chain”.

The successive circles’ centres and radii are analytically known, and in fact rational. It is therefore easy to reproduce this sketch by programming.

```
| sketch circle a o m|
sketch := DrGeoCanvas new fullscreen.
circle := [:n |
  | r c p |
  r := (sketch freeValue: 15 / (n squared + 6)) hide.
  c := sketch point:
    (15 / (n squared + 6) * 5) @
    (15 / (n squared + 6) * n * 2).
  c small; round.
  p := sketch circleCenter: c radius: r.
  n > 0 ifTrue: [circle value: n - 1]].

circle value: 10 .
a := (sketch point: 5@0) name: 'A'.
o := (sketch point: 0@0) name: 'O'.
m := sketch
  middleOf: o
  and: ((sketch point: 15@0) name: 'B').
m name: 'M'.
sketch
  circleCenter: m to: o;
  circleCenter: a to: o;
  line: a to: o.
```

The source code is relatively intuitive and it does not require any comment.

A non-trivial exercise for the reader consists of determining a ruler and compass construction for the figure.

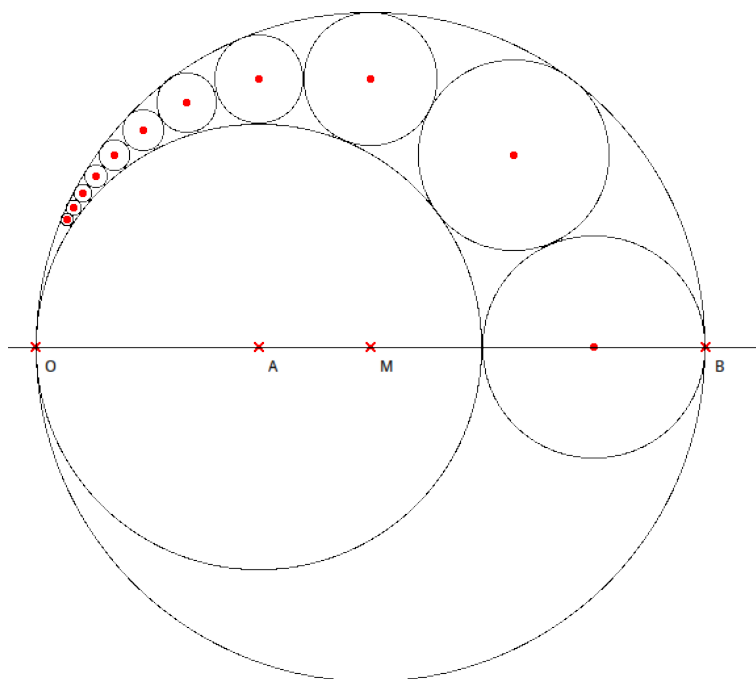


Figure 6.10: Catena di Pappo

## 6.6 $\pi$ calculus

The approximation of  $\pi$  played an important role in the history of mathematics. Numerous methods were proposed, offering a variety of improvements over time, from elementary geometry to calculus. We will examine one of the earliest approaches to the problem, originally carried out by Archimedes, called – although this phrase is not precisely accurate – the **method of exhaustion**. This approach has however the advantage of exposing the essence of the methodology.

Archimedes described a simple ruler and compass construction of regular polygons inscribed inside and outside a circle, and laborious calculations of the lengths of their sides, doubling the number of sides at each step up to 96. In modern terms, this gives three digits of precision, approximately 3.14. We, however, can proceed much more simply, letting Dr.Geo do the hard work, and continuing to polygons of more than a million sides, which takes us nearly to the limits of ordinary computer arithmetic, that is, 16 decimal places for floating point numbers (3.141592653589793). Using much more advanced methods on supercomputers has made it possible to calculate ten trillion digits of  $\pi$ .

We start with the construction of an inscribed regular hexagon. Create a numeric value, and set it to 6. Create a segment (the diameter) and bisect it to get the center. Create a circle with the given center through one of the endpoints of the segment. Create a regular polygon with the given center, one of the endpoints as a vertex, and the numeric value as the number of sides.

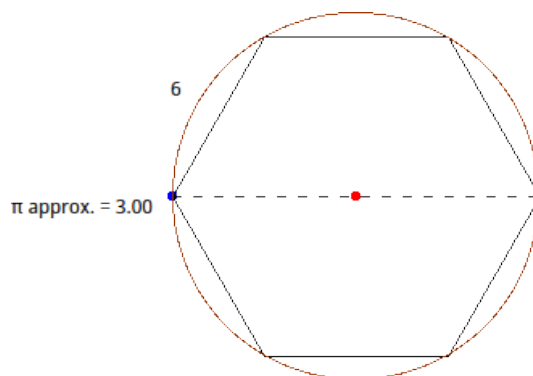


Figure 6.11: Inscribed regular hexagon

The idea of the exhaustion method starts with approximating the length of the circle with the perimeter of the P0 hexagon; then to calculate an approximation of  $\pi$  as the quotient of the perimeter of that hexagon by the diameter of the circle. Clearly, since the side of the hexagon is the same length as the radius of the circle, the resulting initial approximation of  $\pi$  is 3.

In a second step, we construct an inscribed dodecagon. To construct the previous regular hexagon, we used the Dr.Geo tool to construct a regular polygon with 6 vertices. To construct the dodecagon, we just need to change this free value item to 12, and the polygon will update automatically. We calculate the P1 perimeter, then an approximation of  $\pi$  as the quotient of the perimeter by the diameter of the circle.

A small script is written to calculate this quotient; its arguments are the polygon and the circle diameter:

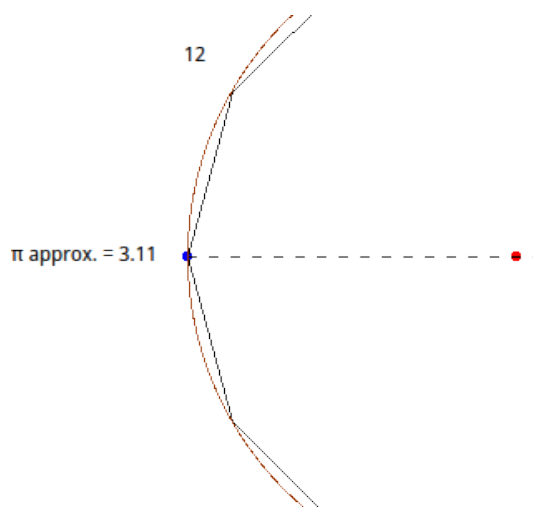
**approxPIpolygon: poly diameter: d**

"PI approximation given an inscribed regular polygon.

Arguments: the polygon and the circle diameter"

^ poly length / d length

Now apply this script to the constructed polygon and its diameter (the segment we started with), and pin the result in the sketch.

Figure 6.12:  $\pi$  approximation

When increasing the number of sides of the polygon, by editing the free value, we improve the approximation.

To display both the approximation of  $\pi$  and its accuracy, we can modify the script to display them together:

```
approxPIpolygon: poly diameter: d  
"PI approximation given an inscribed regular polygon.  
Arguments: the polygon and the circle diameter"  
^ Array  
  with: poly length / d length  
  with: (poly length / d length - Float pi) abs
```

As we increase the number of sides of the polygon, how quickly does this process approach the true value? How far can we go?

## 7 Various tips

Due to Dr.Geo integration with the Pharo Smalltalk environment, there are a few genies we can invoke. Most of them are hidden to the user. It is not that we want to restrain their use, but that we don't want to overload the user when discovering Dr.Geo. As we will show in the following sections, these genies can be invoked from menus, keyboard shortcuts or Smalltalk code.

### 7.1 Programming

In this section we present a few tools to use when writing Smalltalk scripts or composing sketches, beginning with the workspace, the debugger, and the inspector.

#### 7.1.1 Workspace

To show a workspace, click on the Dr.Geo environment background then do *Ctrl-k*<sup>1</sup>.

A workspace, at first glance, is like a text editor. But it is in fact a console to edit Smalltalk code: to write it, to compile it, and to execute it. It is of course possible to paste in code copied from somewhere else.

After invoking a workspace, paste the source code of the Smalltalk sketch below<sup>2</sup>:

```
| sketch function p integral summits |

function := [:x | x * x ].
summits := OrderedCollection new.
sketch := DrGeoCanvas new.
p := sketch point: -1@0.
p hide.
summits add: p.

-1 to: 1 by: 0.1 do: [:x |
  p := sketch point: x @ (function value: x).
  summits add: p hide].

p := sketch point: 1@0.
summits add: p hide.

integral := sketch polygon: summits.
integral color: Color blue.
```

---

<sup>1</sup> Depending on your system, replace *Ctrl* by *Alt*.

<sup>2</sup> To paste a text, try with the shortcut *Ctrl-v* or from its contextual menu (right click).



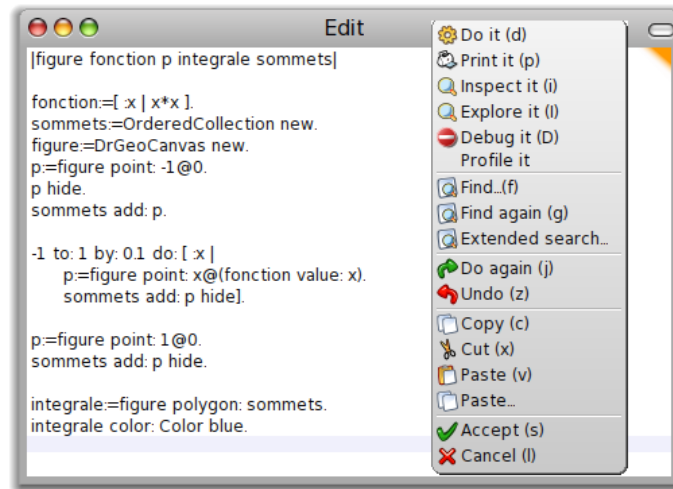


Figure 7.1: Your workspace with the pasted source code and its contextual menu

To compile and to execute this source code, just select it with the mouse and invoke *Do it(d)* in the contextual menu. These two operations can also be done at the keyboard: *Ctrl-a* to select all the source code then *Ctrl-d* to execute it. You immediately get the result of this code, an interactive programmed sketch.



Figure 7.2: Result after executing the source code: the integral of the function in  $[-1 ; 1]$

## 7.1.2 Profiler

When executing complex Smalltalk code, running it with a profiling option let you see its bottlenecks. To do so, in the contextual menu invoke the command *Profile it*. The source code is executed, the sketch is built, and then a profile window informs you about the execution time in different part of the code and the various invoked methods. It is a wonderful way to navigate the execution tree of the code and look for methods consuming too many cycles.

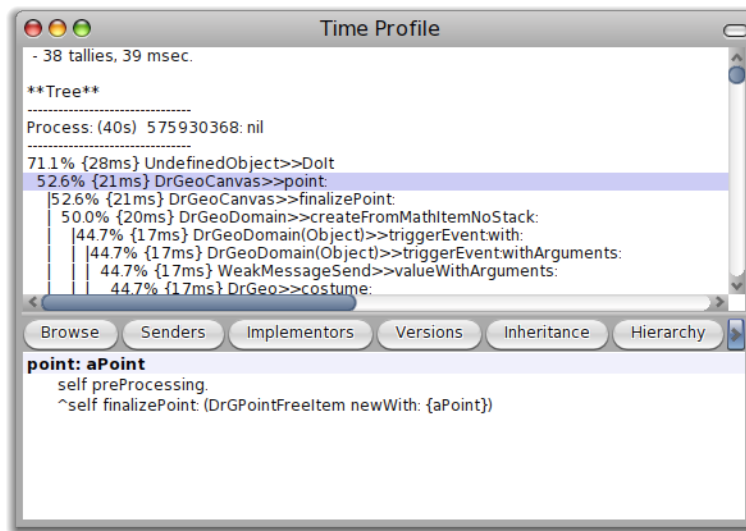


Figure 7.3: Dr.Geo profiling

### 7.1.3 Debugger

Last refinement: stepping through code. This is done through the debugger. Select the code to debug, then in the contextual (right-click) menu, choose the command *Debug it*. The debugger is invoked on the first line of the selected source code, which is executed step by step with the button *Over*, one method call at a time. In the bottom area of the debugger window, at the right, the local variables are shown with type information. Other buttons allow other refinements in step by step execution. You should explore them before looking at Smalltalk documentation.

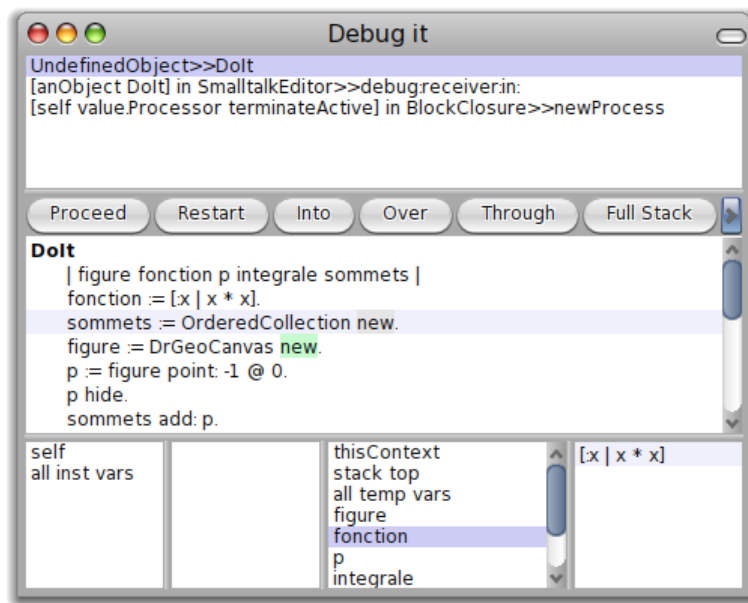


Figure 7.4: The Dr.Geo debugger

As shown in the previous section, the debugger lets you execute the code step by step. In the Dr.Geo environment you can invoke the debugger on any selected code in a workspace with the keyboard shortcut *Alt-D*.

Moreover, the debugger can be invoked in the source code by adding a line `self halt.` – like a breakpoint. In our previous example we modify the source code as follow:

```
...
p := sketch point: -1@0.
p hide.
summits add: p.

self halt.

-1 to: 1 by: 0.1 do: [:x |
  p := sketch point: x @ (function value: x).
  summits add: p hide].
...
```

### 7.1.4 Inspector

In the inspector dialogue, the user examines the attributes of an instance or a variable. The information in the view is updated automatically whenever the inspected object changes.

In our previous example, suppose we want to see the contents of the `summits` collection. In this case, we very simply add a line of code to send the message `inspect` to `summits`. The place in the code where we put it is not very important<sup>1</sup>. It can be at the beginning or the end because we do not have a breakpoint or step by step execution:

```
...
p := sketch point: -1@0.
p hide.
summits add: p.

summits inspect.

-1 to: 1 by: 0.1 do: [:x |
  p := sketch point: x @ (function value: x).
  summits add: p hide].
...
```

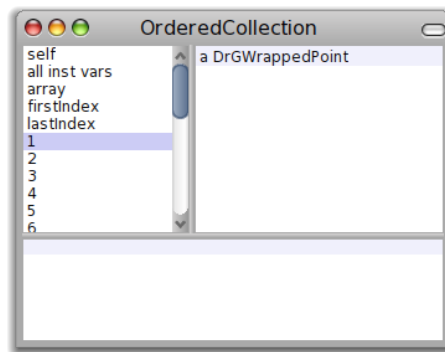


Figure 7.5: The inspector on the variable `summits`

<sup>1</sup> For obvious reasons, we must avoid adding it in the loop.

## Appendix A Dr.Geo Glossary

this Glossary assumes that the learner understands the vocabulary and constructions of elementary Euclidean geometry, and aims to provide simple and clear explanations of terms used in geometry beyond that level, algebra, Dr.Geo, and Smalltalk sufficient for the beginner to proceed with, and also in some significant cases very correct explanations that will help the more advanced learner to understand Dr.Geo and Smalltalk further. Full understanding requires other tools, such as a Smalltalk reference manual and study of Dr.Geo source code, or even extensive study of Computer Science.

Complete understanding is not given to mere mortals, and there are theorems implying that it is actually a self-contradictory idea. Besides, where would the fun be in having nothing left to learn?

**abscissa** In coordinate geometry, the x coordinate of a point. In Dr.Geo, it can also be the difference between the x coordinates of the initial and final points of a vector. See also curvilinear abscissa.

**API** Application Programming Interface; in Dr.Geo, methods usable in scripts to apply to math items, from the class hierarchy `DrGMathItem`.

**argument** A value provided to a function, assigned to a variable that it can act on. In Smalltalk, however, messages are sent to objects rather than applying functions to values. One can view the recipient of a message as an argument of the method invoked by the message. Sometimes called a parameter of a function.

**array** An indexable sequence of Smalltalk objects created with either of two notations. For example, `{}` and `#()` are the empty array; `{1}` and `(1)` are an array with one element, the integer 1; `{'a' . 2}` and `('a' 2)` are an array containing two elements; and so on.

**background process** Define

**balloon help** Explanations of objects in a user interface that appear when the mouse briefly points to the object.

**block** In Smalltalk syntax, a segment of Smalltalk code, optionally including a declaration of local variables, enclosed in brackets.

**block closure** A Smalltalk object of class `BlockClosure` representing a program, defined by a block, that is not an object method, together with the environment it executes in, so that the closure can be a parameter of a control structure such as `whileTrue:`, which is a method of the `BlockClosure` class, or `ifTrue:` which is a method of the `Boolean` class.

**browser** See **class browser**.

**change set** In Smalltalk, a change set is created by saving all new and modified class and method definitions. Although change sets can be accessed via the Extended search submenu in a workspace, they are not relevant to normal use of Dr.Geo.

**class** In Smalltalk, an object representing a type of object (class members) together with the behavior of its members (methods).

**class browser** A Smalltalk development tool that displays the classes and methods of Smalltalk objects in an organized manner, and provides access to detailed information on all objects and code. Also known as System browser.

**class comment** A text string in a class definition usually intended to provide information on the purpose of a class (class intention), any variables it exposes, and how to use the class and its methods (class collaborations, class API).

**closure** See block closure.

**constraint** The relationship between geometric elements in which a change in the position, orientation, or size of one affects the properties of another, or in which properties of

some elements prevent some kinds of change to another. For example, after constructing the intersection of two lines, moving one of the lines generally moves the intersection, but the intersection cannot be moved directly.

**constructionist learning** The terminology of education is vexed and confusing, with the same word used in multiple meanings, and very similar words used in very different ways, all compounded by deep ignorance and misunderstandings about human nature.

Seymour Papert defined constructionism in a proposal to the National Science Foundation entitled "Constructionism: A New Opportunity for Elementary Science Education" as follows: "The word constructionism is a mnemonic for two aspects of the theory of science education underlying this project. From constructivist theories of psychology we take a view of learning as a reconstruction rather than as a transmission of knowledge. Then we extend the idea of manipulative materials to the idea that learning is most effective when part of an activity the learner experiences as constructing is a meaningful product." Geometric constructions and computer programs are but two of many such products.

**control structure** A method of an appropriate class for determining whether or how often to execute a block of code, such as `ifTrue:` or `whileTrue:`.

**Creative Commons** An organization devoted to sharing of content in much the same manner as Free/Open Source Software, providing licenses that guarantee that a work can be copied freely (CC), and if desired translated and improved (Sharealike), as long as the results are published under the same license. It provides other licensing options as well.

**curvilinear abscissa** In Dr.Geo, the parametric coordinate of a free point on a curve in the range  $[0 ; 1]$ . For circles, arcs, and polygons, this is proportional to arc length. For straight lines, and for loci based on straight lines, it is based on the arctangent of the distance from a base point, mapping  $[-\infty ; \infty]$  to  $[0 ; 1]$ . One of the points defining the line is taken to be its origin, with curvilinear abscissa 0.5, while a point on the line with curvilinear abscissa 0 or 1 is at infinity.

**data pane** The area at the left in a Dr.Geo sketch window, listing the items in the sketch in the order of their creation, and giving some essential properties of the items.

**debugger** A Smalltalk tool for investigating programs, which can be used to find and correct errors or to explore the working of Smalltalk code.

**degree** A measure of angles, with 360 degrees in a circle, 180 degrees in a straight angle, and 90 degrees in a right angle.

**development environment** A set of tools for creating, inspecting, and maintaining software.

**domain** The set of values on which a function is defined.

**dynamically typed** Of a programming language, allowing the type of a variable to be changed during program execution, or allowing a data structure to be expanded or shrunk. In Smalltalk, objects of different classes can be assigned to the same variable at different times.

**free point** A point that the user can drag freely in the workspace or on a curve. Contrast with constrained point, defined by its relations to other sketch objects, such as the intersection of two lines.

**Free Software** Software provided under a license that protects the rights of the user, especially the four Software Freedoms defined at <http://www.gnu.org/philosophy/free-sw.html>:

1. 0 The freedom to run the program, for any purpose.
2. 1 The freedom to study how the program works, and change it so it does your computing as you wish. Access to the source code is a precondition for this.

3. 2 The freedom to redistribute copies so you can help your neighbor.
4. 3 The freedom to distribute copies of your modified versions to others. By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

**functional** Of interactive software, requiring commands to be explicitly invoked at each use, as opposed to modal, where a command can be applied multiple times after being selected once. Dr.Geo has elements of both, and can be toggled between them with the single/multiple creation button.

**global variable** A variable accessible anywhere. Contrast with local variable and instance variable.

**homothety** Scale to larger or smaller size. For example, make similar triangles.

**instance variable** A variable declared in the header of a method as a local variable, accessible only within that method.

**local variable** A variable whose value can be accessed or changed only in some restricted context, such as the methods of a single class, or within a single method. Contrast with global variable.

**lock** Fix a Dr.Geo point in place in a sketch so that it cannot be moved by clicking and dragging. It can still be relocated under program control or by editing its coordinates in the property dialog.

**hide** Make a Dr. Geo object invisible and unselectable in the sketch. Hidden objects appear in light gray and can be selected when editing object styles.

**infinity** In geometry, a point at infinity is defined as the intersection of parallel lines. In algebra, an infinity can be defined as the limit of a divergent sequence, as  $1/0$ , as  $\tan(\pi/2)$  and in other ways. Dr.Geo allows for points and even lines at infinity, but does not handle algebraic infinities well.

**inspector** In Smalltalk, a tool for viewing properties of an object.

**label** The name of an object, displayed alongside it. The user can create labels in the object style editor.

**locus** A curve or set of points defined by constraints such as distance from given points or lines. For example, the locus of a point equidistant from two rays is the bisector of the angle between the rays. Plural loci.

**macro** In many programming systems, macro processing means making text substitutions in source code. In Dr.Geo a macro is an encapsulated construction specifying a set of objects as its starting point, and another set of objects as results. The construction must uniquely determine the outputs from the given inputs. Where this is the case, Dr.Geo determines how to carry out the construction and records the procedure for reuse. Macros are built using tools accessed in the macro construction tab.

**math item** Object in a class from the hierarchy `DrGMathItem`, including points and curves of various kinds, numeric values, and others.

**method** In object-oriented programming, a program associated with a class specifying some behavior of the object type of that class (instance method) or of the class itself (class method). The name of the method is called its selector.

**method finder** A Smalltalk tool for searching for methods based on their names, their associated classes, their program text, examples of use, or pragmas.

**method source** The code defining a Smalltalk method.

**method string** Literal string within method code.

**modal** Of software, having distinct states or modes in which particular functions can be applied repeatedly, as opposed to stateless or functional, where the function has to be

specified for each use. Dr.Geo has elements of both, and can be toggled between them with the single/multiple creation button.

**name** In the object style properties dialog for Dr. Geo objects, a label to be displayed with the object.

**norm** Length of a vector.

**object** In Smalltalk, a member of a class whose behavior is specified by methods defined in that class.

**object-oriented** In programming languages, using object definitions rather than data types, where objects are defined in classes by the methods that act on them, and classes can inherit methods from other classes, usually in a tree-structured hierarchy. The class structure of Pharo Smalltalk can be examined using the hierarchy function in the class browser in Dr.Geo.

**Open Source** Software published under the Debian Open Source guidelines or any other license that requires making source code available to all users without charge or other restrictions. Similar to Free Software, but there are differences in the terms of the various licenses.

**ordinate** In coordinate geometry, the y coordinate.

**pane** An area in a Smalltalk window reserved for a particular purpose.

**parameter** Another name for a function argument. See also parametric curve.

**parametric curve** A curve defined by functions of a variable other than the coordinates. For example,  $x = \cos(t)$ ,  $y = \sin(t)$ , where  $x$  and  $y$  are the coordinates of a point, defines a circle with  $t$  as the parameter.

**Pharo Smalltalk** The version of Smalltalk used to implement Dr.Geo.

**pin** Click in a sketch window to place an object at the point selected.

**plane** In Dr.Geo, an extended Euclidean plane, where parallel lines are equidistant, and in principle intersect in the same point at infinity in both directions. Dr.Geo does not handle points and lines at infinity consistently, where a line at infinity joins two points at infinity. This is in contrast with the standard Euclidean plane, which does not have points at infinity; with non-Euclidean planes where there are no equidistant straight lines and there may be points at or beyond infinity; and with the projective plane, where all pairs of lines intersect, but there is no concept of distance or length. It would be possible to use homogeneous coordinates in a projective plane to make Dr.Geo handle infinities consistently, but explaining them is beyond the scope of this glossary. Note that it is possible to model and explore all of these geometries within Dr.Geo, and that an Omar (Our Most Assiduous Reader, in the lore of Conway game theory) could reconstruct Dr.Geo to support any or all of them (and more) directly.

**pointBlockItem** a DrGPointBlockItem, defined as a "Point item defined by a block closure". It is a subclass of DrGPointCoordinateItem, which is a subclass of DrGPointItem, so that it responds to the normal point methods.

**pragma** A system for tagging methods, used by the method finder but not relevant in Dr.Geo.

**protocol** The set of categories for the methods in a class, displayed in the class browser.

**radian** A measure of angles based on the lengths of arcs in a unit circle with circumference  $2\pi$ . 1 in radian measure is the angle subtending an arc of length 1 in this circle, or more generally the radius in any circle. Commonly used in symbolic form in mathematics, where a right angle is represented as  $\pi/2$ . For purposes of arithmetic, angle measure in degrees is usually more convenient.



**reference** Within Smalltalk, any occurrence of a given class name in any class definition or method code. A search for "references to it" using Extended Search on the right-click menu in a workspace opens a Users tool. See Smalltalk documentation for details.

**reflective** In a programming language, providing tools for examining objects by methods other than reading source code files.

**script** A Smalltalk method created in the DrGeoUserScripts class that can be called using the Use a script button in the Numerics tab or the Use a script command in the Numerics menu. The result of a script can be pinned in a sketch, so that the script is saved and reloaded with the sketch.

**selector** In Smalltalk, the name of a method. It may contain more than one part, in which case each part will end with a colon. For example, distance: to:.

**sender** For a given selector, a class that has a method of that name.

**sketch** A geometric figure defined using Dr. Geo commands, including constraints, formatting, and other properties of points, lines, curves, and compound elements. Macro constructions, equations, and pinned scripts are also parts of a sketch.

**sketch pane** The area in a Dr.Geo sketch window below the toolbars where geometric, numeric, algebraic, and script items are placed to form the sketch.

**Smalltalk** An object-oriented, graphical programming language and environment designed by the Learning Resources Group at Xerox PARC (Palo Alto Research Center), the model for all modern Graphical User Interfaces (GUI) after the Apple Lisa and Macintosh were based on it.

**Smalltalk-80** The first publicly-released version of Smalltalk.

**stateless** Of interactive software that is not modal, that is, where commands do not change the state of the application so that only certain commands are active until the user changes to another mode.

**style** Properties of a sketch object, such as name (also called label), colour, shape, size, and the states hide/display and lock/unlock.

**system browser** The class browser.

**type** In programming languages generally, the type of a variable defines the number of bits used to represent it plus any internal structure, and the functions that apply to it. In Smalltalk, object types are defined as classes, including the methods that apply to objects in each class, and do not imply any particular representation in memory or in storage.

**value** Of a variable, the object it refers to at the time, if any. Of a function, the object it returns when called with particular arguments, if any.

**variable** A name that functions like a pronoun in human languages, being able to refer to different objects in different classes at different times. Variables are implemented as objects, just like everything else in Smalltalk. Making a variable refer to a different object is called assigning that object to the variable as its value.

**visibility** The property of displaying in the sketch or being hidden.

**workspace** A Smalltalk window for entering, executing, and debugging Smalltalk code.

# Appendix B GNU Free Documentation License

GNU Free Documentation License  
Version 1.2, November 2002

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc.  
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA  
Everyone is permitted to copy and distribute verbatim copies  
of this license document, but changing it is not allowed.

## 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

## 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means

the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

## 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit

reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

#### 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add

to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

## 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

## 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

## 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of



following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation;

with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. ■

A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

# Conceptual index

## A

angle, geometric .....	13
angle, oriented .....	13
arc, centre .....	10
arc, length .....	12
arc, three points .....	10

## C

circle .....	10
circle, perimeter .....	12
curve, clone .....	18

## D

debugger .....	77
----------------	----

## E

edit, property .....	16
edit, style .....	14
eraser .....	14

## F

free point, on curve, property .....	16
free point, property .....	16
free value, property .....	16

## G

glossary .....	79
grid, display .....	19
grid, magnet .....	19

## I

inspector .....	78
-----------------	----

## L

line .....	9
line, angle bisector .....	9
line, distance .....	12
line, parallel .....	9
line, perpendicular .....	9
line, perpendicular bisector .....	9
line, slope .....	12
locus .....	11
locus, script .....	39

## M

macro construction .....	13
macro construction, create .....	26
macro construction, execute .....	28
macro construction, execute, dialog box .....	28
macro construction, execute, menu .....	29
macro construction, introduction .....	26

macro construction, open .....	21
macro construction, save .....	20, 21
misc tools .....	14

## N

network, share, sketch .....	20
number .....	12
number, free value .....	12

## O

object, delete .....	14
object, hide .....	14
object, move .....	18
object, property .....	16
object, rename .....	14
object, style .....	14

## P

point, coordinates .....	13
point, defined by coordinates .....	8
point, free .....	8
point, intersection .....	8
point, lock .....	14
point, merge .....	18
point, middle .....	8
point, mutate .....	18
point, rename .....	14
polygon .....	10
polygon, regular .....	10

## R

ray .....	9
rename, object .....	14

## S

script .....	25
script, coordinates, point .....	38
script, curve, tangent .....	37
script, edit .....	13
script, examples .....	33
script, examples, complex .....	37
script, examples, Ptolemy's theorem .....	64
script, examples, simple .....	34
script, examples, tangent to a curve .....	39
script, function .....	38
script, function, image .....	38
script, introduction .....	31
script, parameter(s), 0 .....	33
script, parameter(s), 1 .....	36
script, parameter(s), 2 .....	37
script, property .....	16
script, <b>random value</b> .....	35
script, use .....	13
segment .....	10
segment, length .....	12

segment, mark	15
session, open	21
session, save	20
share, network, sketch	20
sketch, export	20
sketch, move	18
sketch, new	2
sketch, open	21
sketch, rename	20
sketch, save	20
sketch, save, network	20
sketch, save, session	20
sketch, scale	18
Smalltalk sketch	44
Smalltalk sketch, angle	54
Smalltalk sketch, arc	50
Smalltalk sketch, circle	50
Smalltalk sketch, complementary methods	57
Smalltalk sketch, equation	55
Smalltalk sketch, examples	44, 58
Smalltalk sketch, examples, animation	58
Smalltalk sketch, examples, Baravelle spiral	68
Smalltalk sketch, examples, Pappus chain	70
Smalltalk sketch, examples, Sierpinski triangle	59
Smalltalk sketch, examples, Teodoro spiral	66
Smalltalk sketch, execute	44
Smalltalk sketch, geometric transformation	51
Smalltalk sketch, line (straight)	48
Smalltalk sketch, locus	52
Smalltalk sketch, number, distance, coordinates	53
Smalltalk sketch, object attributes	56
Smalltalk sketch, point	46
Smalltalk sketch, polygon	51
Smalltalk sketch, ray	49
Smalltalk sketch, run	44
Smalltalk sketch, segment	49
Smalltalk sketch, text	55
Smalltalk sketch, various messages	45
Smalltalk sketch, vector	53
style	14
style, line	15
style, point	14
style, polygon	15
style, script	15
style, segment, mark	15
style, value	15
<b>T</b>	
text	13
text, property	17
tools, debugger	77
tools, debugger, breakpoint	77
tools, inspector	78
tools, profiling	76
tools, workspace	74
transformation	11
transformation, homothety	12
transformation, reflection	11
transformation, rotation	12
transformation, scale	12
transformation, Smalltalk sketch	51
transformation, symmetry	11
transformation, translation	11
<b>V</b>	
value, lock	15
vector	10
vector, coordinates	13
vector, norm	12
<b>W</b>	
workspace	74
workspace, compiling code	75
workspace, executing step by step	77
workspace, pasting code	74
workspace, profiling code	76

# Method index

## A

abscissa on DrGPointOnCurveItem .....	41
abscissa: on DrGPointOnCurveItem .....	41
abscissaOf: on DrGCurveItem .....	42
abscissaOf: on DrGeoCanvas .....	53
altIntersectionOf:and: on DrGeoCanvas ....	47
angle:to: on DrGeoCanvas .....	55
angle:to:to: on DrGeoCanvas .....	55
angleBisector: on DrGeoCanvas .....	49
angleBisectorSummit:side1:side2: on DrGeoCanvas .....	49
arc:to:to: on DrGeoCanvas .....	50
arcCenter:from:to: on DrGeoCanvas .....	50

## B

backgroundColor: on DrGWrappedText .....	56
--	----

## C

center on DrGCircleItem DrGArcItem .....	43
centerTo: on DrGeoCanvas .....	46
circleCenter:radius: on DrGeoCanvas .....	50
circleCenter:to: on DrGeoCanvas .....	50
closestPointTo: on DrGMathItem .....	41
color: on DrGWrappedItem .....	56
contains: on DrGCurveItem .....	42
coordinates on DrGWrappedPoint .....	53
cross on DrGWrappedPoint .....	57

## D

dashed on DrGWrappedCurve .....	57
degreeAngle on DrGAngleItem .....	43
direction on DrGDirectionItem .....	42
distance:to: on DrGeoCanvas .....	54
do: on DrGeoCanvas .....	46
dotted on DrGWrappedCurve .....	57

## E

equationOf: on DrGeoCanvas .....	55
exist on DrGMathItem .....	40
extremity1 on DrGSegmentItem .....	42
extremity2 on DrGSegmentItem .....	42

## F

float:at:from:to:name: on DrGeoCanvas ....	58
freeValue: on DrGeoCanvas .....	54
fullscreen on DrGeoCanvas .....	46

## G

gridOn on DrGeoCanvas .....	46
-----------------------------	----

## H

hide on DrGWrappedItem .....	56
------------------------------	----

## I

intersectionOf:and: on DrGeoCanvas .....	47
item on DrGSegmentItem .....	42

## L

large on DrGWrappedCurve .....	56
large on DrGWrappedPoint .....	57
length on DrGCircleItem DrGArcItem DrGPolygonItem .....	43
lengthOf: on DrGeoCanvas .....	54
line:to: on DrGeoCanvas .....	48
locusOf:when: on DrGeoCanvas .....	53

## M

medium on DrGWrappedPoint .....	57
middle on DrGSegmentItem .....	42
middleOf: on DrGeoCanvas .....	47
middleOf:and: on DrGeoCanvas .....	47
move: on DrGMathItem .....	41
moveAt: on DrGPointItem .....	41
moveTo: on DrGWrappedItem .....	57

## N

name: on DrGWrappedItem .....	56
new on DrGeoCanvas .....	45
normal on DrGDirectionItem .....	42
normal on DrGWrappedCurve .....	56

## O

o on DrGeoCanvas .....	53
ordinateOf: on DrGeoCanvas .....	54
origin on DrGDirectionItem .....	42

## P

parallel:at: on DrGeoCanvas .....	48
parents on DrGMathItem .....	41
perpendicular:at: on DrGeoCanvas .....	49
perpendicularBisector: on DrGeoCanvas ....	49
perpendicularBisector:to: on DrGeoCanvas .....	49
plain on DrGWrappedCurve .....	56
plot:from:to: on DrGeoCanvas .....	57
point on DrGPointItem .....	41
point: on DrGeoCanvas .....	46
point: on DrGPointItem .....	41
point:parents on DrGeoCanvas .....	47, 48
pointAt: on DrGCurveItem .....	42
pointOnCurve:at: on DrGeoCanvas .....	47
pointX:Y: on DrGeoCanvas .....	47
polygon: on DrGeoCanvas .....	51
position: on DrGValueItem .....	43

**R**

radianAngle on DrGAngleItem .....	43
radius on DrGCircleItem DrGArcItem .....	43
ray:to: on DrGeoCanvas .....	49
reflect:axe: on DrGeoCanvas .....	52
regularPolygonCenter:vertex:sides: on DrGeoCanvas .....	51
rotate:center:angle: on DrGeoCanvas .....	51
round on DrGWrappedPoint .....	57

**S**

safeName on DrGMathItem .....	40
scale: on DrGeoCanvas .....	46
scale:center:factor: on DrGeoCanvas .....	52
segment:to: on DrGeoCanvas .....	50
show on DrGWrappedItem .....	56
slopeOf: on DrGeoCanvas .....	54
small on DrGWrappedCurve .....	56
small on DrGWrappedPoint .....	57

square on DrGWrappedPoint .....	57
symmetry:center: on DrGeoCanvas .....	52

**T**

text: on DrGeoCanvas .....	55
text: on item .....	56
text:at: on DrGeoCanvas .....	55
translate:vector: on DrGeoCanvas .....	52

**U**

update on DrGeoCanvas .....	46
-----------------------------	----

**V**

value: on DrGWrappedValue .....	54
valueItem on DrGValueItem .....	43
valueItem: on DrGValueItem .....	43
vector: on DrGeoCanvas .....	53