The Kig Handbook

Dominique Devriese

Contents

Abstract

Kig is an application for Interactive Geometry by KDE.

Chapter 1

Introduction

Kig is an application for Interactive Geometry. It's intended to serve two purposes:

- Allow students to interactively explore mathematical figures and concepts using the computer.
- Serve as a WYSIWYG tool for drawing mathematical figures and including them in other documents.

You can report problems in Kig using the internal bug reporting tool (**Help** → **Report Bug...**).

Since Kig supports macros and the construction of loci, it allows for some rather advanced macros to be defined. If you have created an interesting macro, which you think might be useful for other people, please open a review request for the change, so that it can be evaluated for the inclusion in the distribution (if you do this, it will be licensed under the terms of Kig's license, the GPL, so that other people can freely use and adapt it).

Chapter 2

Basic usage

2.1 Constructing Objects

2.1.1 Constructing points

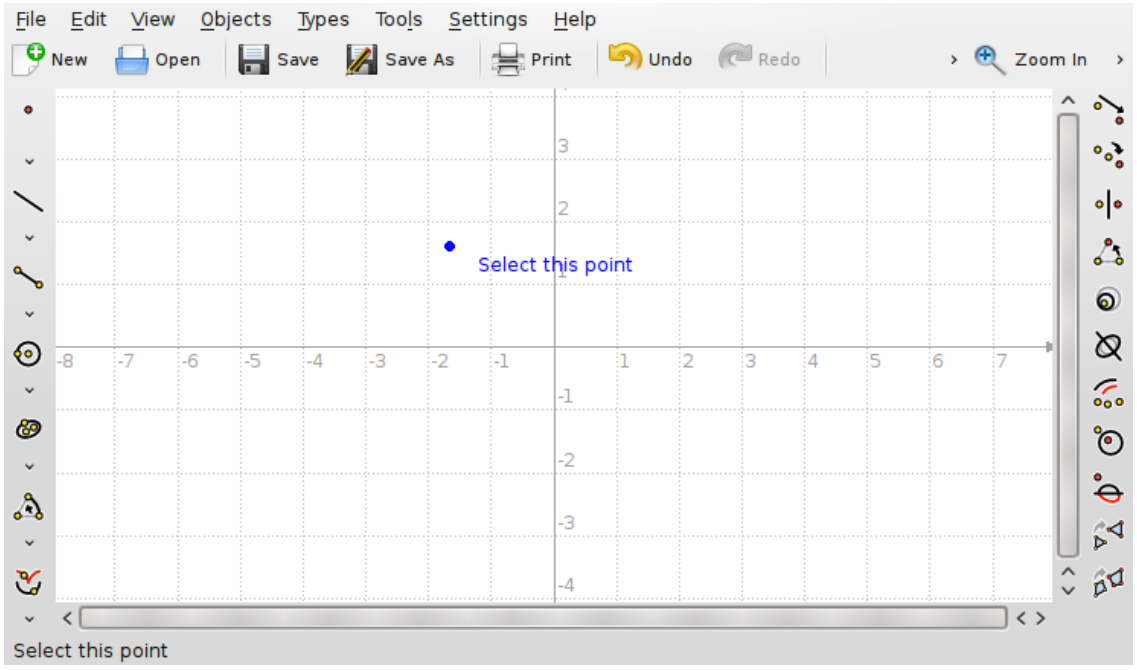
You can construct points in several ways:

- Select **Objects** → **Points** → **Point** from the menubar or press the appropriate button in the toolbar. You can then construct a point by clicking at the desired position in the window.

> NOTE
> Press the **Shift** key to activate a snap raster mode which allows you to set any point in a construction exactly to points of the grid.

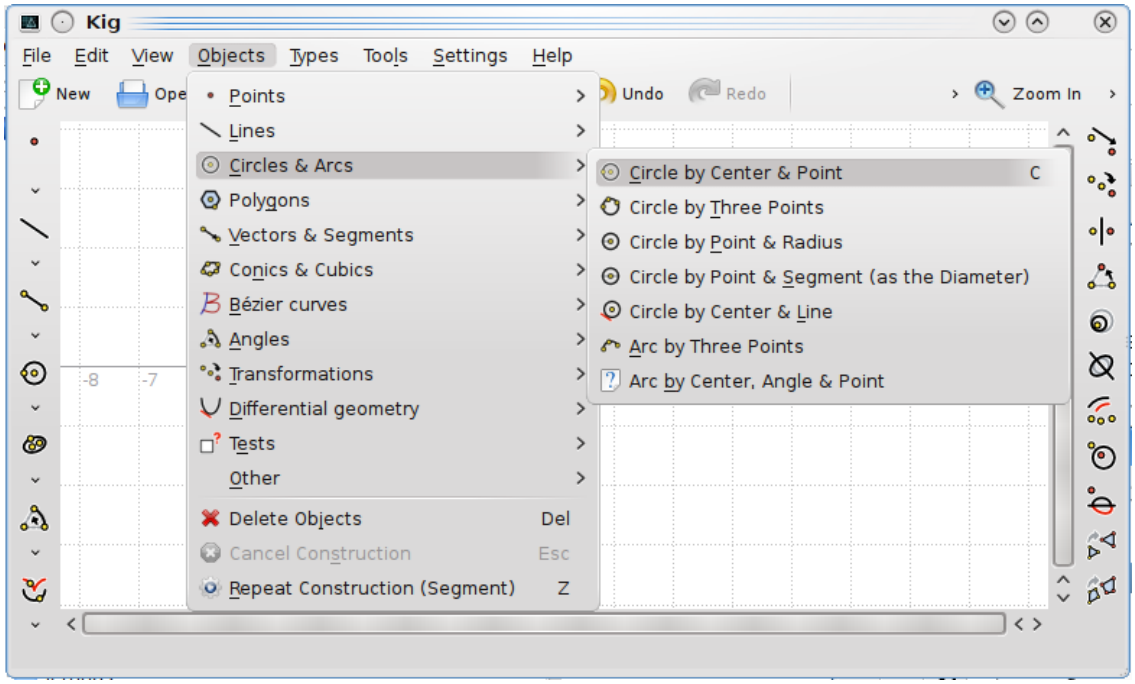
> NOTE
> Actually, this works the same way for constructing other objects as well: click on the desired menubar entry or toolbar button and select the necessary items to construct the object.

- Since you often need to construct points, simply clicking somewhere in the screen with the middle mouse button will construct a point for you, without going to a menu or button.
- You can construct points while you are building other objects in the background, optionally selecting them for the object you are building. For more on this, see Section 2.1.2.



2.1.2 Constructing Other Objects

Constructing objects other than points is usually done by selecting the appropriate entry in the **Objects** menu, or by clicking on one of the toolbar buttons.

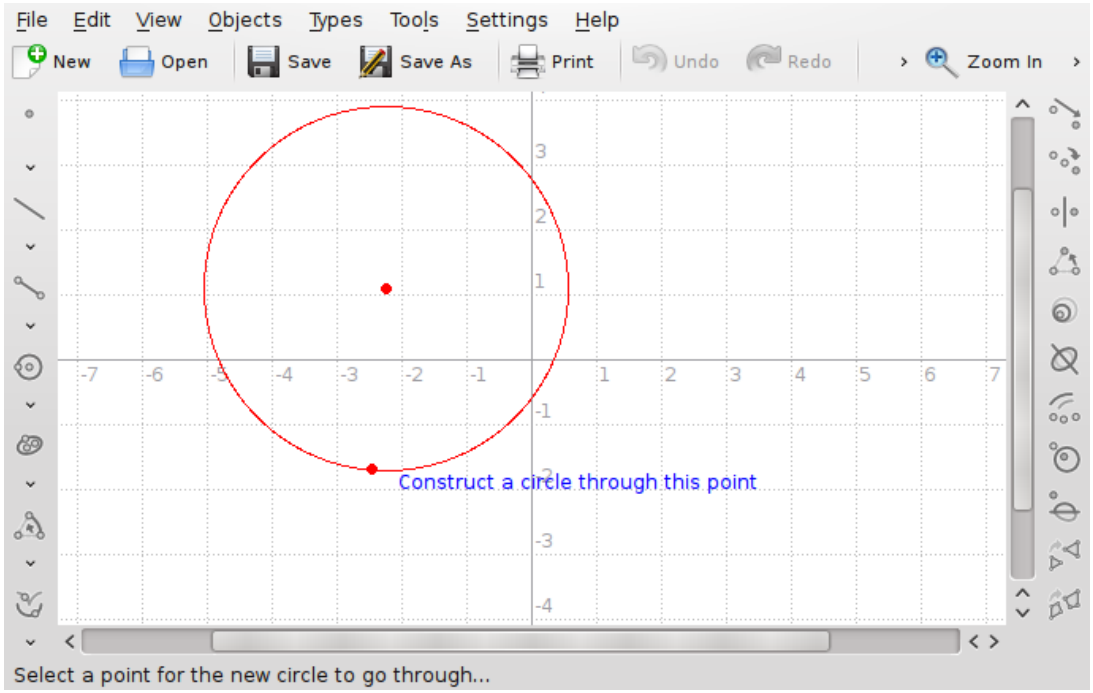


This will start the construction of the chosen object type. All of these types require arguments. For example, if you selected to construct a circle by center and point, you will need to give two points: one for the center, and one for the point on the circle.

These arguments are objects too, which can also be selected, simply by clicking on them. When you move the cursor over an argument you want to use to construct an object, a preliminary

image will be shown of the object, so you will know what it will look like. For objects that require points as arguments, you can place a new point at the current cursor position and select it by clicking the left mouse button.

You can always cancel the construction of the new object by pressing the **Esc** button or by clicking on the **Cancel Construction** button (red octagon with an 'X') on the toolbar.

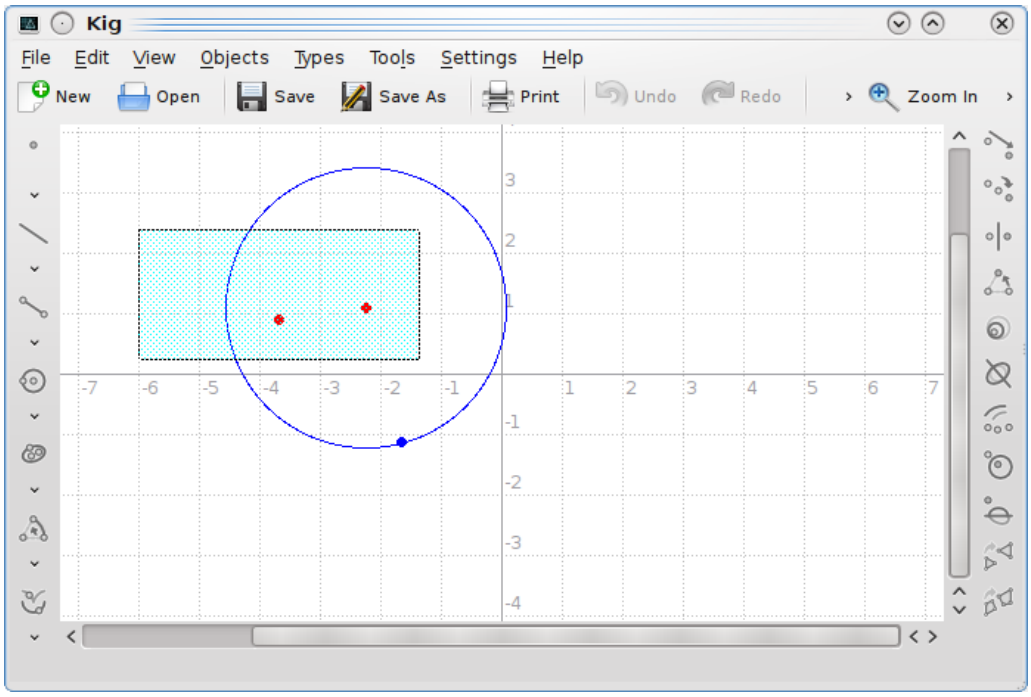


2.2 Selecting Objects

Selecting objects can be done in two ways:

- Simply clicking on an object causes that object to be selected, clearing the current selection. If you want to select multiple objects simultaneously, hold down the **Ctrl** key while clicking on an object.
- By clicking and dragging on an empty spot on the screen, you can select all objects within the rectangle that is created. This action will clear the current selection. As in the previous case, holding down the **Ctrl** key allows you to keep the current selection.

When you have more than one object under the mouse (indicated by **Which object?** in the statusbar and at the mouse cursor), you can easily choose which object select or add to the current selection. Click with the left mouse button, while holding the **Shift** key or click the right mouse button, to have a popup with all the objects under the mouse. Then, you can select the object you need. As said before, the behaviour of the **Ctrl** key will affect the selection.



2.3 Moving Objects

To move objects, you must first select them.

When the objects you want to move are selected, you can start moving them by left mouse button clicking and dragging any one of them. When you are done, simply release the left mouse button.

> NOTE
> For some types of objects (especially when defined by complicated loci), moving them can be slow on old hardware. This is unfortunate, but inevitable, given the calculations involved.

> NOTE
> If you right mouse button click one of the selected objects, and choose **Move**, moving the mouse will move the object. When the object is moved to the desired position, another left mouse button click will stop the moving of the object.

2.4 Deleting objects

Deleting objects is done by first selecting them, and next doing either of these:

- Press the **Del** key.
- Press the **Delete Objects** button on the toolbar.
- Right-click on one of the objects, and select **Delete** in the context menu that appears.

2.5 Showing and hiding objects

In Kig, objects can be hidden. This is done by selecting the objects, right mouse button clicking one of them, and selecting **Hide** in the context menu that appears.

To unhide the objects, use the **Edit** → **Unhide All**. This will unhide all currently hidden objects.

2.5.1 Night Vision

Night Vision is a particular way of working with hidden objects. When you have to move or change something in one or more object but without unhiding all the hidden objects you have, then the night vision mode will be of benefit to you.

Basically, it allows you to see the hidden objects as if they were visible, so that you can manipulate them as you would normally. In Night Vision mode, the hidden objects will be visible with a grey colour.

To toggle the night vision mode, use **Settings** → **Wear Infrared Glasses**.

2.6 Undo/Redo

In Kig, you can undo almost any change you make in the document. Just use the **Undo**/**Redo** buttons on the toolbar, the **Edit** menu or the appropriate shortcuts.

2.7 Full Screen Mode

Kig also has a Full Screen mode. To use it, click the appropriate button on the toolbar, or select **Settings** → **Full Screen Mode**.

To leave Full Screen mode, right mouse button click the screen at a place where there is no object present, and select **Exit Full Screen Mode**, or press the shortcut **Ctrl**+**Shift**+**F**.

Chapter 3

Kig Object Types

Kig supports a rather large number of object types. Please note that not all of the available object types are shown in the toolbars: there are some objects that you can only construct via the **Objects** menu in the menu bar. Of course, as with all KDE applications, the contents of the toolbars are configurable. Try out the **Settings** → **Configure Toolbars...** option if you want to do this.

Chapter 4

Advanced Usage

4.1 Context Menus

Kig has context menus for its objects. Try right mouse button clicking on an object in order to see a context menu appear. There are many options: for constructing other objects, setting colours, pen width, style, and even hiding, moving or deleting objects. Some objects have options of their own (e.g. you can redefine certain points to be constrained to a line if they previously weren't, etc.). These options should be very straightforward to understand.

Some of the actions in the context menu of an object need another defined object as reference, for example:

Rotate a shape a specific number of degrees around a point

1. Click on **Objects** → **Angles** → **Angle by Three Points** and make your desired angle of rotation somewhere in the Kig window, generally in an unobtrusive place. If desired, right click on the angle and click on **Add Text Label** → **Angle in Degrees**.
2. Click with the right mouse button on the angle and select **Set Size** and insert the new size for this angle in the dialog.
3. Click on **Objects** and construct your desired object.
4. Right click on your shape, and select on **Transform** → **Rotate**.
5. Set your desired point of rotation.
6. Click on your angle.
7. Your shape is rotated!

Translate an object

1. Click on **Objects** → **Vectors & Segments** → **Vector** and construct the object by selecting the start and end point somewhere in the Kig window.
2. To adjust length and direction of the vector you have to edit its start and end point. Select them and choose in the context menu **Set Coordinate...**
3. Click on **Objects** and construct your desired object.
4. Right click on your object, and select on **Transform** → **Translate**.
5. Select the vector to translate by.

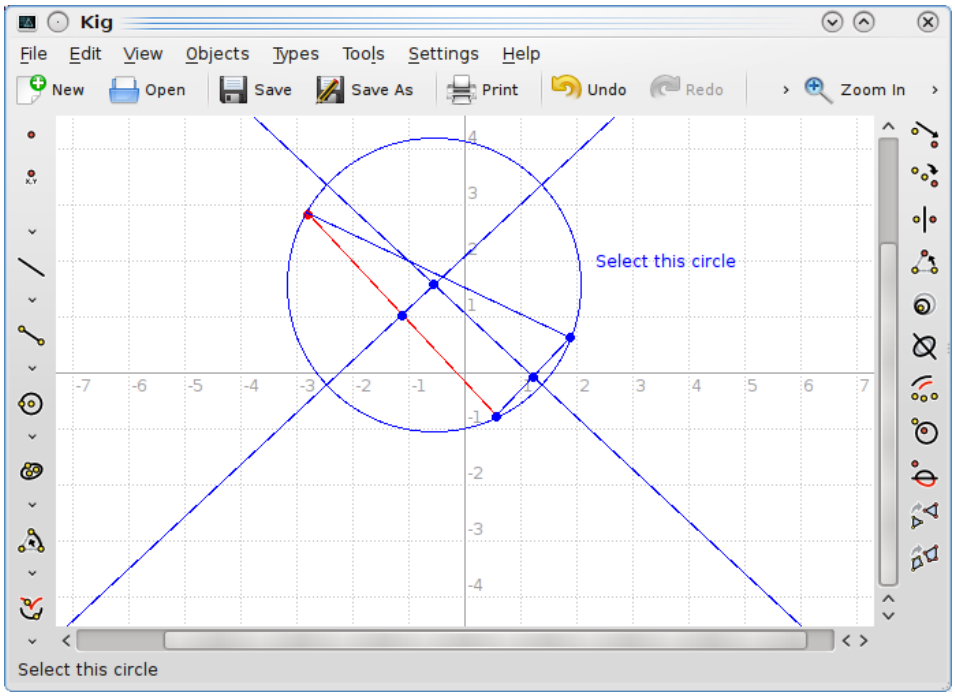
4.2 Document context menus

Right-clicking on the document (i.e. not on an object) will present a popup that you can use to start constructing a new object, change the coordinate system used, show hidden objects, zoom in and zoom out of the document and switch to the full screen mode.

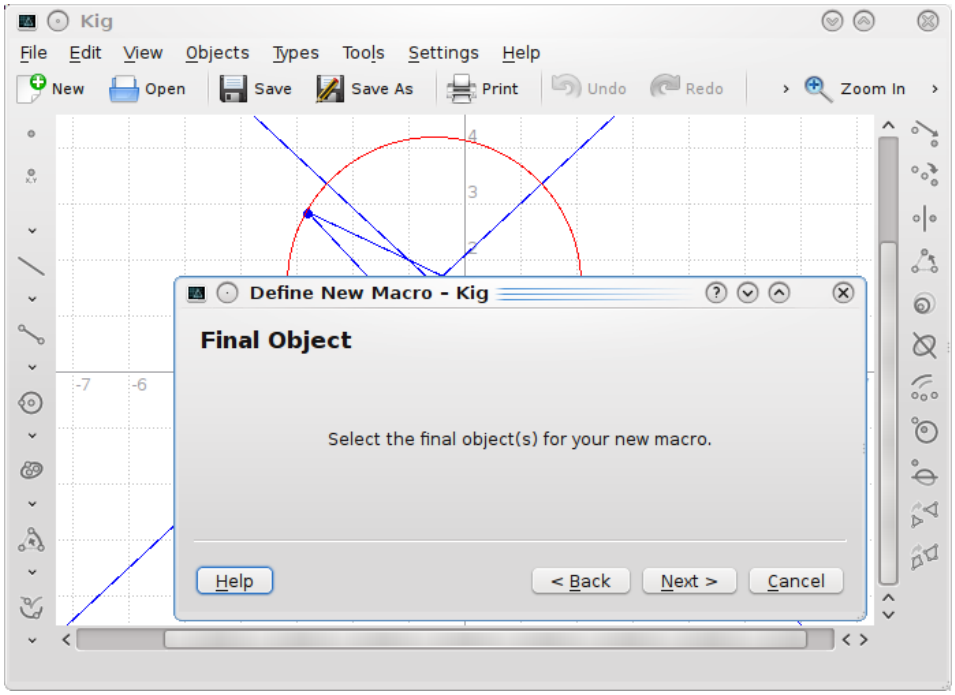
4.3 Defining Macros

One of the more advanced features in Kig is its support for macros. This allows you to define new types of objects from other ones which are defined already.

For example: Suppose you want to make a macro for constructing a circle from three points on it. You would input three points, then construct some perpendiculars and midpoints until you find the center. Now you can use the existing **Circle by Center & Point** command (using one of the three points as the point for this command). The following image should make this a bit more clear:

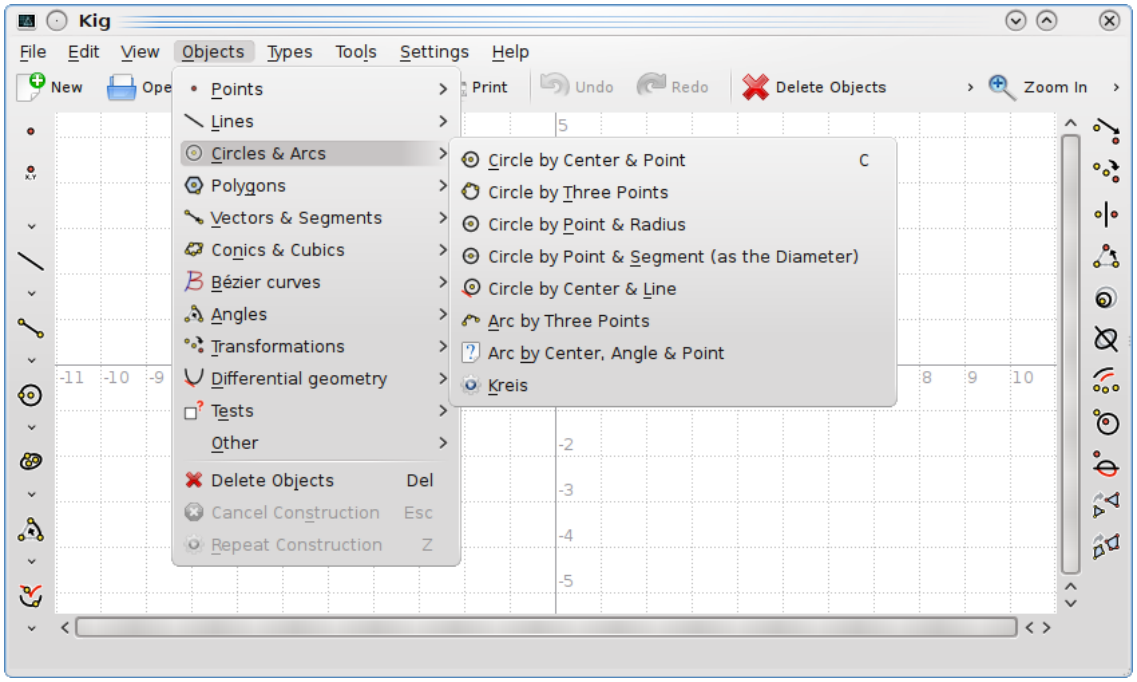


Next comes defining the macro. Select **New Macro...** from the **Types** menu, or click on the button on the toolbar. A wizard will appear and ask you to select the given objects. In our example, these are the three points. Select the three points (click on them to select, click again to unselect) and click the **Next** button to continue. Finally, select the last objects (only the circle in our example).



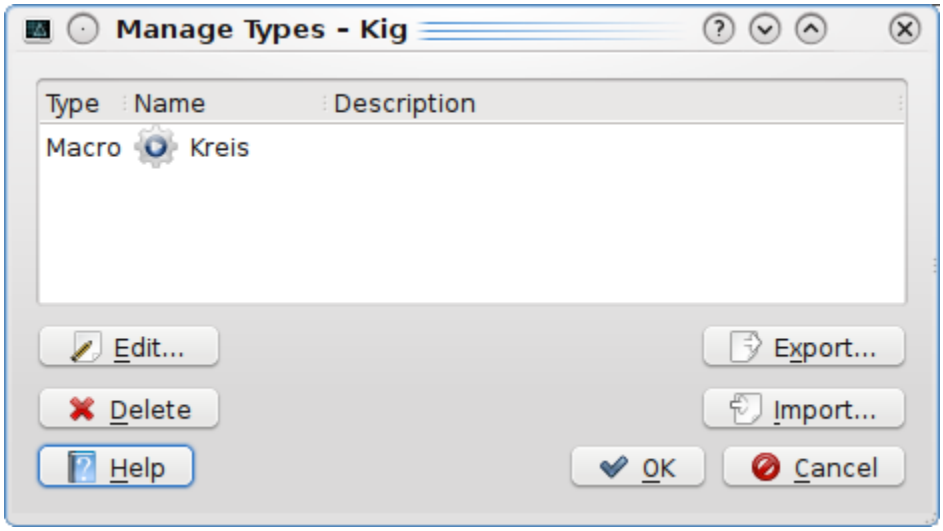
After the previous steps are completed, click the **Next** button to continue. Enter a name and optionally a description for your new type, and click the **Finish** button. Your macro type is now finished.

To use the new macro type, click its button on the toolbar or use the **Objects** menu. Constructing a macro object is just like constructing any other object.



4.4 Working with types

As you saw in the previous chapter, Kig allows you to create your own objects. Kig also makes sure that once you have created an object, it is saved on exit and loaded on startup. You do not have to manually save or load macro definitions. However, Kig does allow you to do more with the macros. If you select **Types** → **Manage Types...** from the menu, you will see a dialog where you can edit your types. It allows you to modify the existant types, delete types that are no longer used, export them to a file, or even load them from another file.

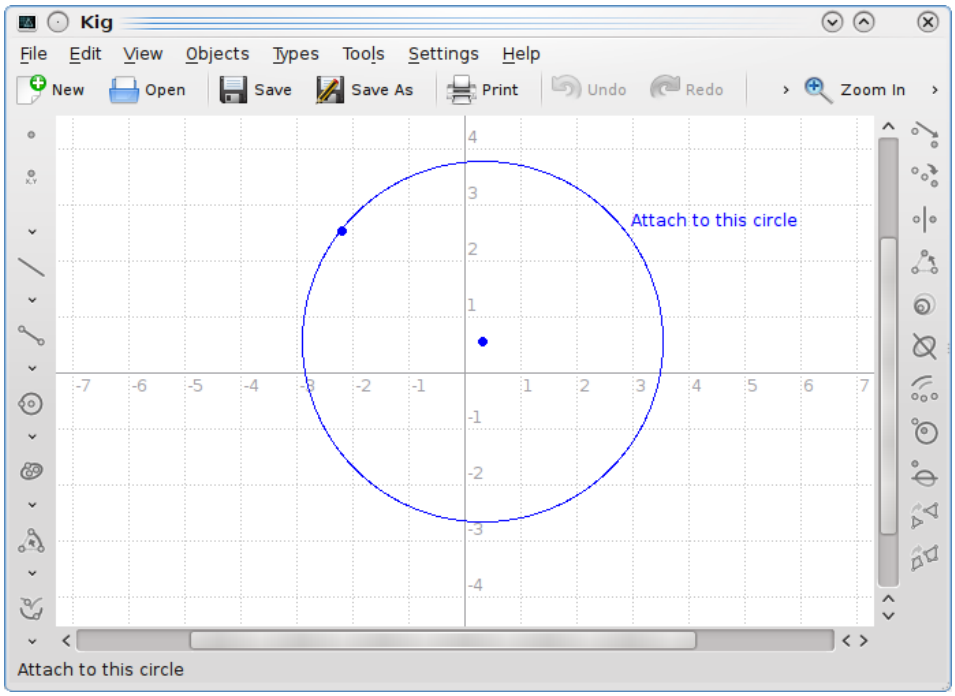


4.5 Text labels

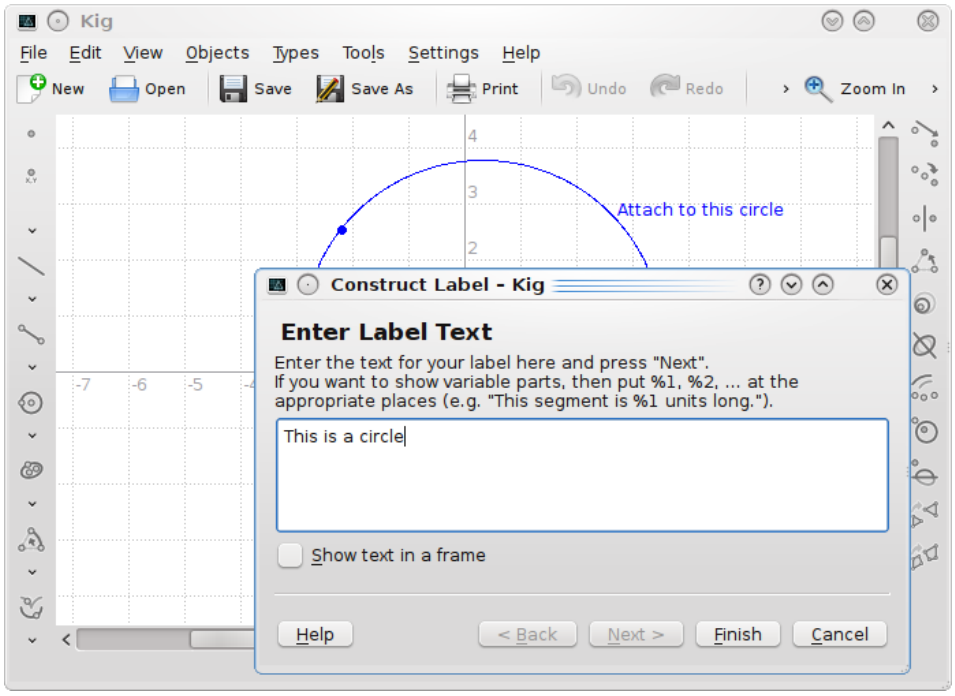
Kig allows you to add text labels to a construction. This is very useful for adding names, explanations or other text to constructions. Kig can also display variable information about objects (also known as 'properties').

To start constructing a text label, simply press the **Text Label** button in the Kig toolbar or select **Objects** → **Other** → **Text Label** in the menubar.

Next, you have to choose a location for the text label. You can either just select a random location on the screen, or choose to 'attach' the label to an object.



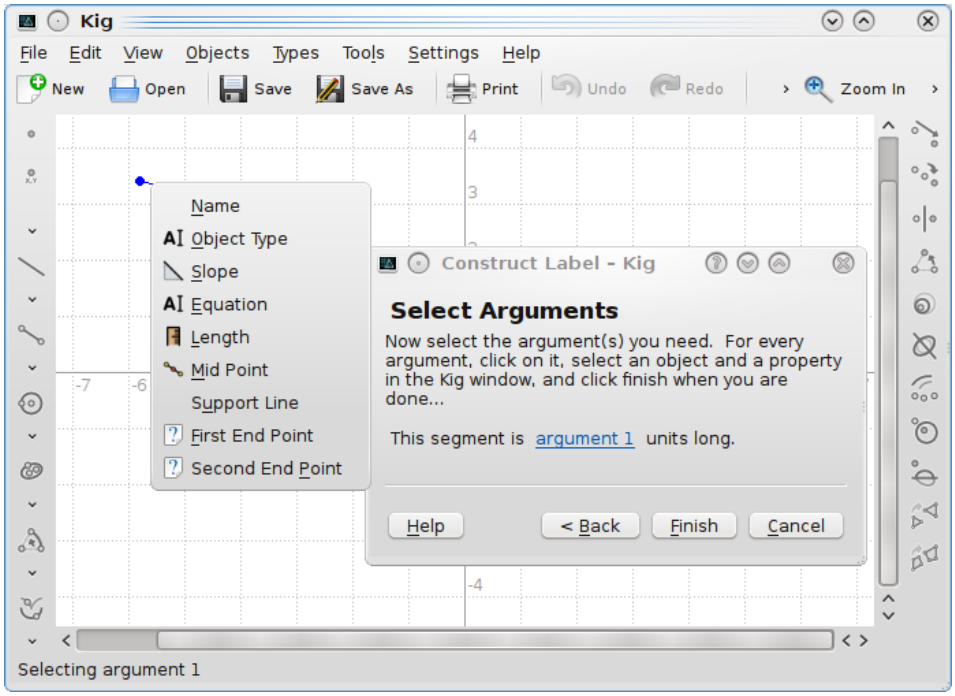
After you have selected where to put the label, the text label dialog appears. Here, you can type in the text that you want in the new label, and click **Finish**. You should now see the label in your document.



The previous example was a simple one, and limited to just text. However, there is also support for showing variable information about objects in a label (e.g. you can construct a label with the text 'This segment is %1 units long.' where %1 would be dynamically replaced with the length of a specific segment).

To do this, enter a text with a number of placeholders (%1, %2 etc.) in it. Then, press the **Next** button to continue. If you want to change the text or variables later, you can go back using the **Back** button.

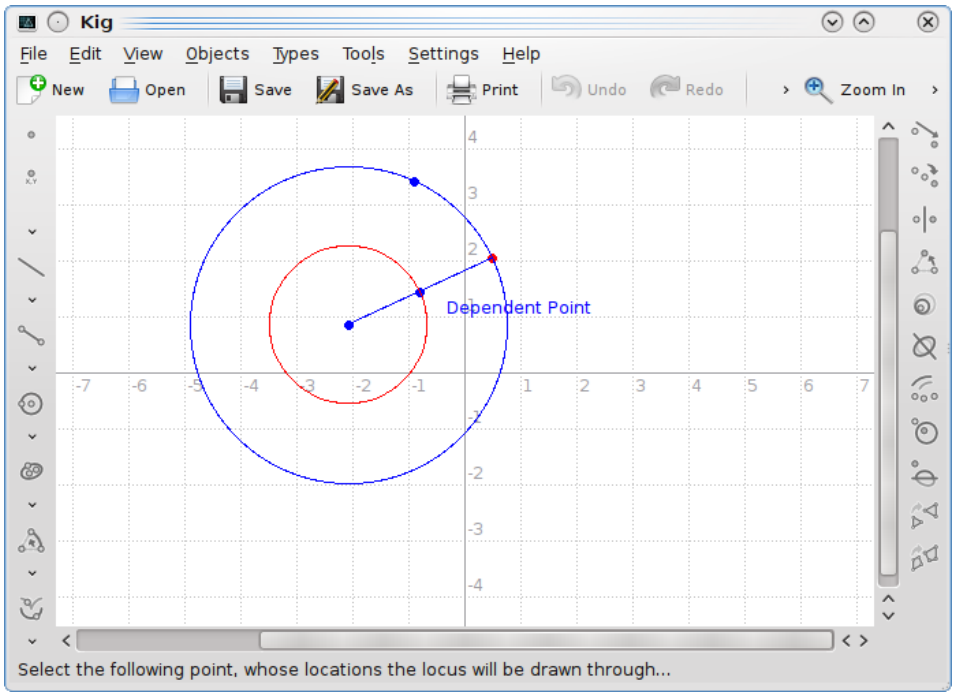
The wizard now shows the text you entered with all placeholders replaced by something like **argument 1**. Selecting the property connected to a certain argument is done by first clicking on the argument in question. Then click on the object that you need and that has this property, and then select the property itself in the popup menu that appears. For instance, in the example above, you would click **argument 1**, click on the correct segment in the main Kig window, and select the property **Length**. Afterwards, you can fill in the rest of the variable parts, or select another property for one of the variable parts if you wish to change it. When you are ready, click the **Finish** button to complete the construction of the text label.



4.6 Loci

Kig supports the use of loci. A locus is mathematically defined as the set of all points or lines that satisfy or are determined by specific conditions; as in 'the locus of points equidistant from a given point is a circle'. Let's look at an example of how to use loci in Kig:

Consider the following geometrical construction: We draw a circle, and a point that can move only along its circumference (construct this point by positioning the cursor on a circle, and clicking the middle mouse button. If you then try to move the resulting point, you'll see that you cannot move it off the circle). Then, we draw a segment from that point to the center of the circle, and the midpoint of that segment.



Now if you move the point that is constrained to the circle, you'll see that the second point moves along with it. If you were to hold a pen at the second point, and move the first point around the entire circle, a new circle, half the size of the other would be drawn. The path that the second point travels while the first one moves around the circle is its locus.

Actually constructing the locus of a point is very easy. Click the **locus** button in the toolbar, or select **Objects** → **Other** → **Locus** from the menubar. Then select the constrained point as the moving point (the text **Moving Point** will appear as you move the mouse over it), and the other as the dependent point. The locus of the dependent point will then be drawn.

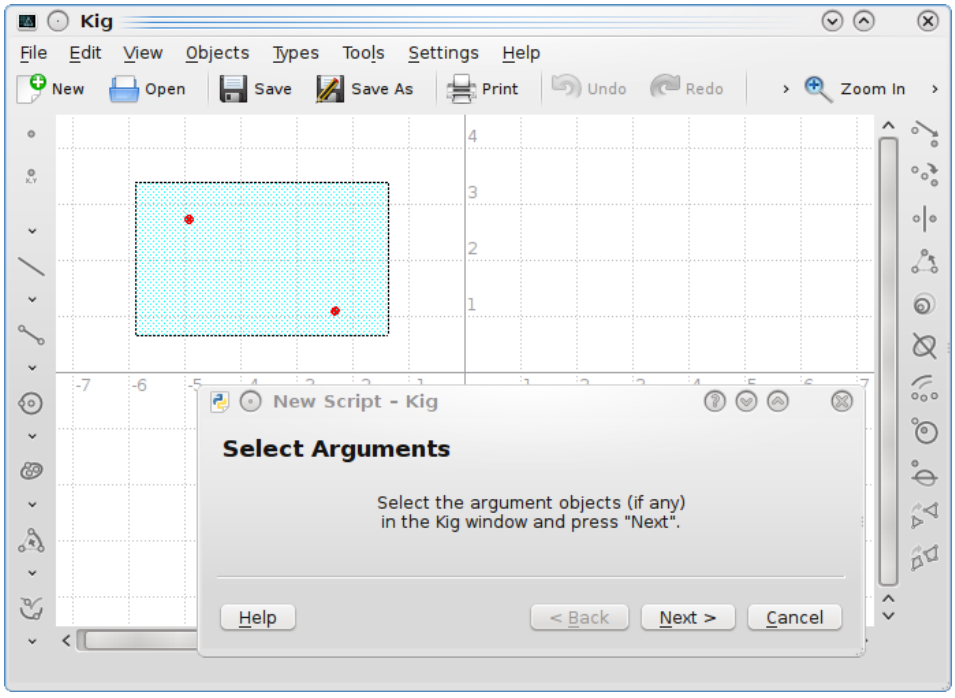
Chapter 5

Scripting

Kig allows you to create custom types in the Python scripting language. This is a very advanced feature, and I know of only one other Interactive Geometry program that has a similar functionality (the GNOME program Dr.Geo).

Python Scripting in Kig basically allows you to create your own objects from certain parent objects. For example, if you are a math teacher, and you have some fancy way of calculating an interesting point on a conic, then instead of messing with complex constructions and macros, you could just write down in Python code how the point is to be calculated and then Kig will show it for you.

Suppose you were not aware of the Kig built-in type 'Mid Point', and you wanted to show the midpoint of two given points. You would then click on the **Python Script** button in the toolbar, or select **Objects** → **Other** → **Python Script** from the menubar. You are then presented with a wizard that allows you to proceed.



The first thing you have to do is select the arguments for the script object. In our example, this means the two points of which we want to show the midpoint. Select them in the Kig main window, and click **Next** to proceed.

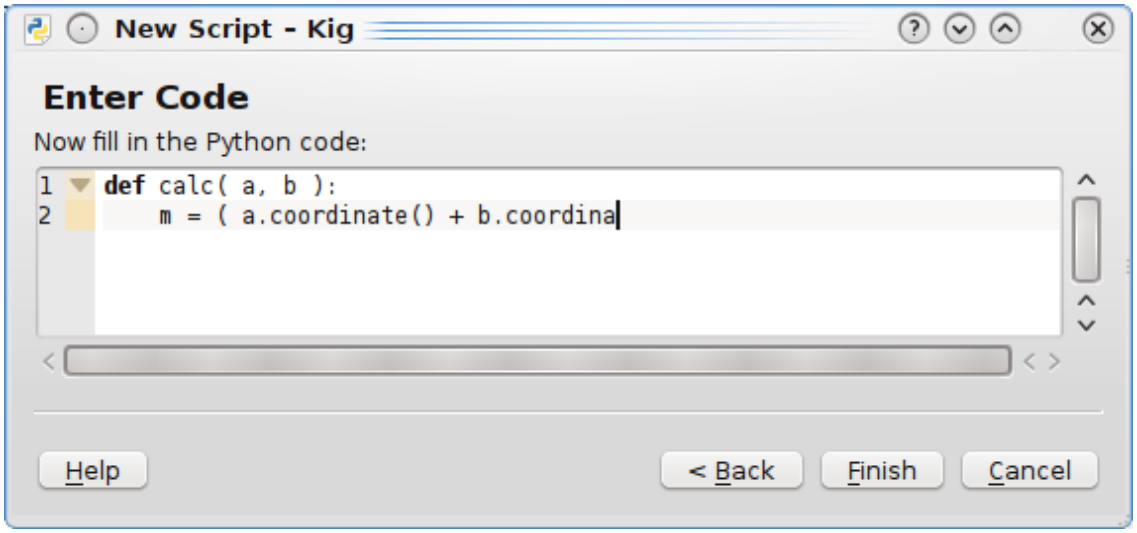
Now you are presented with a text edit box where you can enter the code for you script object. Template code and some comments are already in place. It is important to make sure that your code is valid Python code. People familiar with Python will notice that we are actually defining a Python function called `calc`. It is therefore necessary to adhere to the Python rules for defining functions. For example, every line of the function should start with a **Tab**. The first line not starting with a **Tab** ends the definition of the function.

The Python function that we want to define is called `calc`, and in our case it accepts two arguments. These are the objects you have selected as arguments in the previous screen. You need as many arguments as you have selected there. They are called *`arg1`* and *`arg2`*, but you can change their names to something more meaningful if you want.

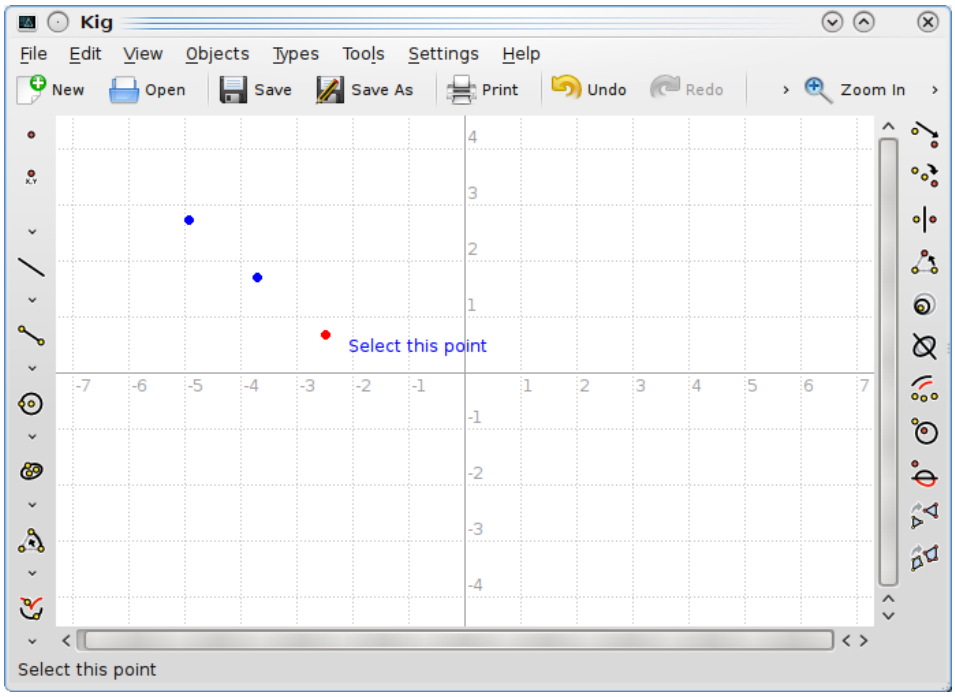
In the function, you can do all sorts of calculations that you deem necessary, using the two arguments if needed. You should return the object you want to define. In our case, this is a `Point` object. The two arguments are also `Point` objects, and we can use the `Point.coordinate()` function to define the coordinates of the two given points.

The calculation necessary in our example is very simple, we simply add the two sets of coordinates, and divide the new set by two. We then construct a new point using the result. The Python code needed is:

```
def calc( a, b ):
    m = ( a.coordinate() + b.coordinate() ) / 2;
    return Point( m )
```



If you now click the **Finish** button, the new object will appear in the Kig document. If you move one of the points, the newly created point will move along with it. Much more powerful objects can be built in this way: you are encouraged to try it out.



All objects in Kig can be used in the Python code. As we have seen above, points are of the `Point` class, and you can use e.g. the `Point.coordinate()` method. You can also return all kinds of objects, not just a `Point`. Many more classes and methods are available in the Kig Python code, and a more complete reference is provided on the Kig website.

Chapter 6

Kig Features

- Kig is an open source application. This means that you are free to use and modify it any way you like it. Distributing Kig is subject to some restrictions, basically that everyone should have the same rights to use Kig, including your modifications, as you and me.

  Free software programs are developed in a very open spirit, and its developers are usually very responsive to user feedback. Therefore, if you have any questions, complaints, or whatever about Kig, please contact the kde-edu mailing list at kde-edu@kde.org.

- Kig is a KPart application, which means that you can embed it into other KDE software. If you open a `.kig` file in Konqueror, it can be opened directly in the Konqueror screen without the need to start an external application.

- Working with Kig should be very straightforward. Constructing objects is easy and interactive, with preliminary results being shown, etc. Moving, selecting and building all work as one would expect them to. Undo support should also be very intuitive.

- Kig supports macros to be defined in a straightforward manner. These objects are presented to the user like normal objects. They are saved on exit and loaded on startup, so that they aren't lost on exit. You can manage these objects in the **Manage Types** dialog (see Section 4.4). You can export them to files, import them from files, edit and delete them.

- Kig saves its data in a clear XML format.

- Kig supports the construction of loci.

- Kig allows you to export a Kig file to some interesting formats, like images, XFig and LaTeX files, and SVG vectorial images. This is rather useful, because not all programs support the Kig file format yet.

- Kig has a very flexible transformation system.

- Kig aims to be compatible with its competitors. This is why it supports the KGeo file format, the KSeg file format and partially the Dr.Geo and Cabri formats; moreover, support for other formats is planned.

Chapter 7

Credits and License

Main authors:
Kig copyright 2011-2015 David E. Narvaez david.narvaez@computer.org
Kig copyright 2002-2013 Maurizio Paolini paolini@dmf.unicatt.it
Kig copyright 2004-2010 Pino Toscano pino@kde.org
Kig copyright 2002-2005 Dominique Devriese devriese@kde.org
Documentation copyright 2002-2004 Dominique Devriese devriese@kde.org.
Documentation copyright 2004-2005 Pino Toscano toscano.pino@tiscali.it.
Reviewed by Philip Rodrigues phil@kde.org.

Appendix A

Contribute

A.1 Free Software

Kig is Free Software. This means that its source code is freely available on the Internet, and everyone can use it, read it, modify it, and distribute it.

A.2 Contribute

In this chapter I want to point out to you (the user) the rights that Kig's license gives you. As with all free software, you are allowed (and encouraged) to fix problems you encounter while using it, to add features you miss, to distribute your modified program, and to send these modifications back for inclusion through the review tools available for the KDE community.

If you are uncertain of your rights to use this software, or other people's right to use any modifications you make to this program etc., please read the license. You can find it in the `COPYING` file in the Kig source tree or the license link in the **About Kig** dialog.

A.3 How to contribute ?

Any contributions are welcome. If you don't like the icons, or think that the manual needs updating, or if you have this really cool macro that you want to share with the world, do not hesitate to send it to me. Please note that your contributions will be distributed under the terms of the GNU GPL; you can find the terms of this license in the `COPYING` file in the Kig source tree, and in the Credits and License chapter in this manual.