

Project Title: Adding support for Geogebra files in the KDE Interactive Geometry application KIG.

Name: Aniket Anvit

College: National Institute of Technology, Durgapur, India

Degree: Bachelor of Technology (3rd year)

Department: Computer Science and Engineering

E-mail: seeanvit@gmail.com

freenode IRC nick: packo (on #kde, #kde-devel, #kde-edu)

Phone : +918942031270

Location (City, Country and Time Zone):

Durgapur

India

UTC + 5:30

Problem Description:

Both KIG and Geogebra are popular applications for Geometry learning. They are widely used by the Geometry learning/teaching community. A lot of teachers use them to demonstrate Geometrical constructions (or graphs of various algebraic functions) to their students. Despite the fact, that both have been designed to serve the same purpose and are very similar in features/functionality, currently Geogebra worksheets cannot be opened in KIG.

My project aims at providing support for Geogebra native format in KIG enabling users to open Geogebra worksheets in KIG. This will open up KIG to a large number of people who have only been using Geogebra so far and let them taste some wonderful KIG-exclusive features.

Motivation:

One of the nicest things about any application is its ability to support file-format of its competing softwares. KIG already has a lot of filters and supports file-formats of DrGeo(The GNOME tool for Geometry), KSeg, KGeo. Geogebra is a very popular software among the Geometry and Algebra learning/teaching community. KIG and Geogebra - both have been created for the same purpose but there are a few features in KIG (like there are a variety of tests like parallel test, perpendicular test, collinear test, various transformations like harmonic homology, projective shadow, similitude etc.) which are not in Geogebra . If the Geogebra worksheets could be opened in Kig, then users would be able to use these Kig features to manipulate those Geogebra worksheets. Also, there are some teachers who have a large number of quizzes, test or demonstrations pre-made in the form of Geogebra worksheets. Re-creating them as Kig-files (.kig) would be a lot of redundant effort. If Kig gets the ability to support Geogebra worksheets then they can open and manipulate those worksheets in KIG as well.

Project Goals:

Geogebra saves its files in two forms -

- 1) Geogebra Worksheets (.ggb)
- 2) Geogebra Tool (.ggt)

The goal of my project is to enable KIG to open both these types of files. KIG is only designed for Geometry but Geogebra supports some Algebra features in addition to Geometry (as the name implies Geometry + Algebra). So, some Geogebra worksheets may contain function-defined plots in addition to the Geometric constructions. If a user tries to open such worksheet in KIG, then she will be presented with a dialog stating that the worksheets contains some Algebraic plots too. If the user tells, then KIG will open up those plots in **KAlgebra**.

If Tool Files (.ggt) files are opened, a new menu entry (or sub-menu entry in the *objects* menu) will be made (similar to when they are opened in Geogebra, they get placed on the default toolbar) using which the user can create constructions.

Implementation:

Geogebra saves its worksheets in the form of an XML (Extensible Markup Language) file. So, the basic idea behind implementing geogebra-worksheet support, is to read the information imbedded between the XML tags and create a KigDocument according to the information so obtained. Detailed specification about the XML tags used by Geogebra can be found at the below mentioned web-pages -

http://wiki.geogebra.org/en/Reference:XML_tags_in_geogebra.xml

http://wiki.geogebra.org/en/Reference:XML_tags_in_geogebra_macro.xml

A start in the direction of this project has already been made. KIG already had existing filters for DrGeo and KSeg. The approach taken to implement the Geogebra filter is slightly different from the other existing filters. Here we are using the *XSLT* approach (eXtensible Stylesheet Language Transformations) to read and convert the data from a Geogebra worksheet into a KigDocument. A *KigFilterGeogebra* class has been created by subclassing the *KigFilter* and *QAbstractXmlReceiver* classes. First we are applying XSLT on the XML file of Geogebra. This is being done to extract the information that we require from the pile of XML data . This way we get a simplified XML format containing information about what types of objects the worksheet contains and their relationship with other objects (i.e. which object has been constructed/ depends on which other objects). From here the objects and their arguments can be obtained easily and homogeneously. Then we use *QXmlQuery* to query the simplified XML data and use the callback interface provided by the *QAbstractXmlReceiver* class to read the objects and their arguments/ dependencies. The *endElement()* and *startElement()* virtual methods (inherited from the *QAbstractXmlReceiver* class) are suitably overloaded to recognise object-types and create objectHolders of required types. Finally, when all the

objectHolders have been allocated, the generated KigDocument is returned.

Given below is a table showing all the construction-options that Geogebra provides, and their expected difficulty level in adding support for these construction-containing-worksheets in KIG.

<u>S.No</u>	<u>OBJECT</u>	<u>IMPLEMENTATION</u>
1	Point	Done
2	Point on Object	X
3	Mid-Point	Done
4	Intersection	Done
5	Line	Done
6	Segment	Done
7	Segment (given length)	Easy
8	Ray	Done
9	Polyline	Easy
10	Vector	Easy
11	Vector (from Point)	Easy
12	Parallel Line	Easy
13	Perpendicular Line	Done
14	Perpendicular Bisector	Easy
15	Angle Bisector	Easy
16	Tangents	Medium
17	Polar and Diameter	Medium
18	Locus	Difficult
19	Polygon (from Points)	Easy
20	Regular Polygon	Medium
21	Rigid Polygon	Medium
22	Vector Polygon	Medium

23	Circle (centre and point)	Done
24	Circle (centre and radius)	Medium
25	Compass Circle	Medium
26	Circle (3 points)	Done
27	Semicircle (3 points)	Medium
28	Circular Arc (centre and 2 points)	Medium
29	Circumcircular Arcs	Easy
30	Circular Sector	Difficult
31	Circumcircular Sector	Difficult
32	Ellipse	Easy
33	Parabola	Easy
34	Hyperbola	Easy
35	Conic (5 Points)	Easy
36	Angle	Easy
38	Slope	Easy
39	Reflect about a line	Easy
40	Reflect about a point	Easy
41	Reflect about a circle	Easy
42	Dilate from Point	Difficult
43	Tools	Difficult

* Those constructions which are exactly same in both KIG and Geogebra will be easily supportable. However , those which are not exactly same , there will be a little manipulation done with the inputs (arguments) to create them in KIG (the medium and difficult ones).

Adding support for all the above mentioned constructions would require some smart XSL Transformations and suitable overloading of the *startElement()* and *endElement()* methods of

the *KigFilterGeogebra* class.

Here is some XML code fragment from a Geogebra worksheet containing a Line :

```
<command name="Line">
  <input a0="A" a1="B"/>
  <output a0="a"/>
</command>
```

The attributes of input element represent the parent points from which the Line is constructed. (**A** and **B** are the parent points which determine the slope and intercept of the line).

Here is what the XSLT applied form of the above XML code looks like :

```
<Kig>
  <FixedPoint label="A">
    <Double value="-0.52"/>
    <Double value="5.04"/>
  </FixedPoint>
  <FixedPoint label="B">
    <Double value="2.98"/>
    <Double value="4.28"/>
  </FixedPoint>
  <LineAB label="a">
    <Parent label="A"/>
    <Parent label="B"/>
  </LineAB>
</Kig>
```

The Line here is of type AB(i.e. constructed from 2 points A and B as parents). The parent tags contain the parent objects of the object.

* The result of applying XSLT stylesheet on a XML file can be obtained by using the [xmlpatterns](#) command line utility.

```
xmlpatterns myStylesheet.xsl myInput.xml
```

The above command will produce the output of application of the stylesheet file myStylesheet.xsl on the xml file myInput.xml. This is particularly useful for debugging purpose.

Locus Implementation:

To draw a Locus there needs to be a dependent point and one dependent point. Locus feature of both KIG and Geogebra are similar.

Here is some XML code fragment from a Geogebra worksheet containing a locus :

```
<command name="LocusEquation">
  <input a0="Q" a1="P"/>
  <output a0="a"/>
```

</command>

As can be seen , Locus objects are saved just as normal objects i.e with the command tag. The input attributes *a0* and *a1* represent the dependent and the independent point respectively.

Geogebra tools :

Geogebra tools are equivalent to KIG macros.

If a user wants to create a series of dependent objects from some given independent ones repeatedly then he can save it in the form of tool. Tools allow construction of derived objects (meaning constructed with the help of other primary objects) repeatedly with ease.

Tools are saved very much like regular geogebra worksheets files (i.e. with the command tags).

Plots :

Geogebra worksheets may contain some plots created with algebraic-functions . Unlike Geogebra, KIG currently has no input-box facility (no CAS ability) and can only draw predefined objects . Therefore if the Geogebra worksheet contains any such plots , then KIG will tell the user that the worksheet he is trying to open contains some algebraic-plots which KIG can't open. It will then ask the user if she still wants to open the plot. If the user says yes, then KIG will open those plots in KAlgebra. Rest of the constructions contained in the worksheet will be drawn in KIG.

Here is some XML code from a Geogebra worksheet containing one such plot (actually a plot of $f(x) = x^2$) :

```
<expression label = "f" exp="f(x) = x^(2)"/>
<element type="function" label="f">
  <show object="true" label="true"/>
  <objColor r="0" g="0" b="0" alpha="0.0"/>
  <layer val="0"/>
  <labelMode val="0"/>
  <animation step="1" speed="1" type="0" playing="false"/>
  <lineStyle thickness="2" type="0" typeHidden="1"/>
</element>
```

The label and type attributes of the element tag describe the function that the plot represents.

* *KToolInvocation::startServiceByDesktopPath* method will be used to start KAlgebra If the user decides to open the plots.

Tentative Timeline:

Before Coding Period Begins: Learning more advanced uses and implementations of XSLT.

Studying the KIG and KAlgebra code in deeper details to make my work as clean and bug-free as possible.

19th May - 24th May : Adding support for Geogebra Ray, PolyLine, Vector in KIG

25th May - 31st May : Adding support for Geogebra Parallel Line, Perpendicular Line, Perpendicular Bisector, Angle Bisector in KIG

1st June - 10th June : Adding support for various Geogebra Polygons, circles, arcs and conics in KIG.

11th June - 20th June : Adding support for tangents, Polar and Diameter.

21st June - 30th June : Adding support for Geogebra locus in KIG.

Mid-Term Evaluations : By Mid-Term KIG will be able to support most of the constructions of a regular Geogebra worksheet (.ggb) file.

1st July - 5th July : Adding support for various transformations (Reflection from line, point) and
'Dilate from Point' feature of Geogebra.

5th July - 20th July : Adding support for Geogebra tools in KIG. .ggt files when opened in KIG will
behave as macro.

20th July - 31st July : Adding support for Geogebra algebraic-plots in KAlgebra.

1August - 7th August : Buffer week (for some unforeseen difficulties).

7th August - 10th August : Writing Code Documentation

10th August - 18th August : Testing , Debugging and further polishing of my work.

After GSoC : Monitoring of my implementations and Bug-fixing in KIG and KAlgebra. Implementing any other ideas KIG or KAlgebra which I gather during the GSoC period.

About Me:

My name is Aniket Anvit and I am in 3rd year of my Bachelor's degree in Computer Science from National Institute of Technology – Durgapur, India. My areas of interest are Physics, Maths and

Multimedia.

I have a thorough knowledge of C++ (which is my primary programming language) and well acquainted with Qt-Framework and KDE libraries.

I have been using KDE distribution for 3 years now and love the edu-application very much.

Github : <https://github.com/aniketanvit>

Blog : <http://aniketanvit.wordpress.com/>

Obligations During GSOC Period:

The official GSoC coding period this year ranges from (19th May) to (18th August). My college vacations are from the start of May to mid of July. So, a large part of the coding period lies in my vacation time . I am also not taking up any internships. I have no other obligations during most of the coding period which means I will have ample time and focus for my project. I will work 40-45 hours per week (or as necessary) to ensure the completion of my project in time. I shall keep all the progress regarding my project posted on my blog <http://aniketanvit.wordpress.com/>

.

Previous Contributions :

Bug fix in Dolphin (the file manager) -

<https://projects.kde.org/projects/kde/applications/kde-baseapps/repository/revisions/17eeee6b03e7beb7ba6d216b63641d63d9c894af>

Google Summer of Code 2013 on STEP (The Physics Simulator)