Intersect Improvements

Google Summer of Code Program 2015 Project Proposal

Name:	Oana-Georgiana Niculaescu
E-mail:	oana.niculaescu@gmail.com
University:	University Politehnica of Bucharest, Master's Degree (starting Fall2015) - University of Massachusetts Boston, PhD
Current city, country:	Bucharest, Romania, until September, 2015
Phone:	+ 40 0727 840 100
Skype:	oana.niculaescu
Website	*https://github.com/elf11*

1. Technical aspects of the proposal

GeoGebra supports the intersect command but it needs some improvements to allow for more objects to intersect. What I propose for this project is a redesign of the command to allow new intersect options and implementations of those options.

About

GeoGebra already supports the Intersect command for a variety of objects, the goal of this project is to improve on this command and add implementation for the missing cases. The project development can be broken down in 2 parts:

- adding and improving the algorithms for intersecting 2D objects
- adding and improving the algorithms for intersecting 3D objects

I've decided to work on this project because it is one important improvement that the application needs (if I would be a GeoGebra mentor/core developer I'd like to have this implemented), and because I consider it will give me real leverage for contributing later to the project too.

Project Deliverables

- April 27 (community bonding period) - I intend to continue solving bugs and add improvements for the opened issues that can be found on Trac. During this time I will also try to establish with the mentors what parts of my proposed schedule needs adjustments (if any). Start looking at the redesign options for the command and discuss them with the mentors.
- May 25 (coding period begins officially) - Redesign the existing commands if need be and start implementing the Intersect additional features for 2D.
- June 26 (Midterm Evaluation) - Have most of the important Intersect 2D commands implemented, do the midterm report. Start working on the Intersect 3D feature. Have woking test cases for the 2D implemented commands.
- August 22 (Final Evaluation) - Implemented all the agreed upon Intersect 3D functions, have working test cases for the implemented code.
- August 28 (Final Results Announced) I would have polished code, fixed the additional details, assisted with integration.

- Beyond the program - Stick around for maintenance or any other minor tasks (NOTE: As I mentioned, I'll vanish a week when I'll move abroad(around 1st of september)), keep being involved in community as much as I can.

Technical description

After looking over the GeoGebra forum and what the users on there are asking for when it comes to the Intersect command (I've also looked at what versions of the command already exists) I've concluded that some of the desired feature for this improvement would be the addition of the following commands:

- 2D wise
    - Intersect[locus, line]
    - Intersect[locus, polyline]
    - Intersect[locus, conic]
    - Intersect[curve, conic]
    - Intersect[polyLine, polyLine]
    - Intersect[Circle, PolyLine]
    - Intersect[Ellipse, PolyLine]
- 3D wise
    - Intersect[Ellipsoid, plane]
    - Intersect[conic, plane]
    - Intersect[cone, sphere] - not sure if this is not already implemented (found the Intersect[quadric, quadric] command but couldn't make it work for cylinders, it might be just me and not being fully able to understand how it works

What I propose is for me to get comfortable first with the Intersect command code. I've seen that there are at the moment 3 types of intersect command:

- Intersect
- IntersectPath
- IntersectConic

After looking through the Intersect command code, I think starting work on the Intersect[polyLine, polyLine] command would be most indicated. This is similar with the Intersect[line, polyLine] command, so I estimate that understanding what is going on there won't be a problem.

Besides the code there would be .ggb files that would add test cases for each and every one of those new features.

I am not sure if the proposed project and the timeline are underestimated/overestimated, but if they are then I am willing to adjust them with the mentors. In case I overestimated how much time it would take to implement the proposed features then I'd continue to develop any new features commonly established with the mentors until the end of the coding period.

My estimation is that for each command I would need roughly 1 ½ weeks to implement and test properly (for the 2D commands). I think this is pretty realistic considering that my Normalize[] command took roughly the same amount. Probably the commands in the beginning will take a little bit longer than the ones at the end. Anyway, as I stated before, if I overestimated the time I should spend on every command, then I can adjust the timeline.

Proposed timeline

<u>April 27 - May 24 (Community Bonding Period)</u>

<u>April 27 - May 11</u>

- during this first 2 weeks I plan to continue solving bugs and polish the already submitted patches (if need be); I'm also planning to start with one of the Intersect commands, most probably the Intersect[polyLine, polyLine] one.

<u>May 11 - May 26</u>

- in the second part of the month I hope to have the Intersect[polyLine, polyLine] command implemented and tested, also I hope for more discussions with the mentors during which we will establish more clearly the goals of the project and what parts of the Intersect command they consider most "we must have"

<u>May 25 - June 26 (First Half)</u>
<u>May 25 - June 10</u>

- during those weeks I'll develop the Intersect[locus, line] command and test it, I'll probably start on the Intersect[locus, polyline] one too.

<u>June 10 - June 26</u>

- finish implementing and testing Intersect[locus, polyline] command too; and implement the Intersect[locus, conic] command too

Deliverable: by the midterm evaluation I hope to have at least 4-5 command implemented and tested; also the midterm evaluation report will be prepared in time

<u>June 27 - August 21 (Second Half)</u>
<u>June 27 - July 11</u>

- implement the other 2D Intersect command features, in my proposal those would be Intersect[curve, conic], Intersect[Circle, PolyLine], Intersect[Ellipse, PolyLine] (the last two should be very similar and should not take that much time. Also add tests for them and make sure they work properly.

<u>July 11 - July 25</u>
- the 3D commands that I want to implement are the ones above, I think those are nice to have features; the 3 of them would probably take up like 3 weeks (Intersect[Ellipsoid, plane], Intersect[conic, plane], Intersect[cone, sphere]), so I'd start developing them during this time and finish sometime in around the 1st of August

<u>July 25 - August 8</u>

- continue implementing the 3D commands and add tests for them; still have to find any new ones

August 8 - August 22

- those weeks I'll spend adding and testing other intersect commands, have to think some more about it. I'll also do the final evaluation report, polish the code and test it.

2. Human aspects of the proposal

I am Oana, a 24 year old Masters' student at the University Politehnica of Bucharest, Romania, studying Computer Networks and Security, but this fall I will start my PhD at the University of Massachusetts, Boston, where I will study data security and data consistency problems in computer networks. I've also got my Bachelor degree from the University Politehnica of Bucharest in Computer Engineering.
I am passionate about technology and science in general, and computers and software in special. As an undergraduate I've tried to explore as much as I could of the Computer Science field. I've worked on diverse projects, some of them low level - I've done development for routers' operating systems (mostly C), application level - development for Android applications and internet banking applications (mostly Java), but also web development using HTML/CSS/JS.
Regarding the main languages I am familiar with I will list below a few projects I've done using the ones I am most comfortable with:

- Java (since it is the main requirement of GeoGebra project I am interested in):
  - I've done development for various applications using Java in two of my internships; one of the project was a bank back-end system. The application was a REST app, that provided the user with the possibility to interrogate a DB that had information about clients accounts (the account number, how much money does the client has in that account, the latest transfers from and to that account, reports for the last month/6 months etc.). For this project I've developed the dynamic report generation part using the iText library.
  - I've also implemented a peer-to-peer file sharing system; the details of this system were as following: the main design pattern used for developing was the Mediator Design pattern, the GUI was done using Java building blocks (SWING), the clients transfer the files directly peer-to-peer, but they connect before hand to a web server and share with this one the list of files they have put to sharing. The other clients get from the web server the list of available files to download. The network module is done using Java NIO. (project source here https://github.com/elf11/idp)
  - different Android applications, one of them can be found here https://github.com/PDSD-2014/niculaescu-patrascoiu it is a Shared Whiteboard application, clients that use the same AP can connect to each other and draw on a common whiteboard.
  - other smaller projects that can be found on my GitHub account (https://github.com/elf11/KNN_haos_prediction , https://github.com/elf11/JavaAssignments)
- C projects (I've mostly used this during my internships - so the code is not open source, but there are some repositories with programs that I've coded on my Github account):
  - another client server application, the server intermediates the communication between the clients (https://github.com/elf11/client_server_application)

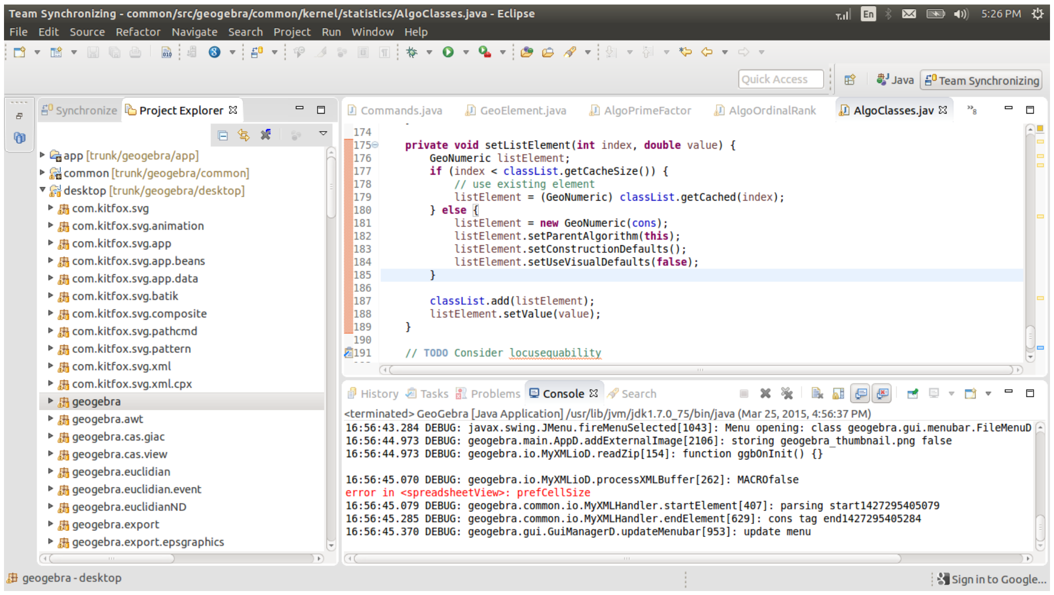
- implementing and optimizing a Blas library function and comparing my results with the Blas library implementation (comparing the performance of my function against the library function) (https://github.com/elf11/BlasFunction)
- and other assignments that could be found on my Github account

I am new user to GeoGebra, meaning that I did not use the program before thinking of applying with this organization for GSoC. But, I think I am a fast learner and I got the gist of it pretty fast.

About contributing to the GeoGebra project, I know that there is a 40h/week requirement and I am willing to spend this time on developing the project each week. Also, I agree to GeoGebra's Copyright Assignment (http://geogebra.org/static/contributions/GeoGebra-Copyright-Assignment-Agreement.pdf) for my contributions. After the end of GSoC I will continue to contribute to the development of the software, I am not sure how much time I will be able to commit each week after the end of the program, so a commitment of X hours a week is unrealistically. I will move countries so the first two weeks or so after the end of the program I probably won't be able to be in contact with the organization (this is after 1st of September), but after that I'd spend some time on bug fixing and offering support on the Intersect command development and future enhancements.

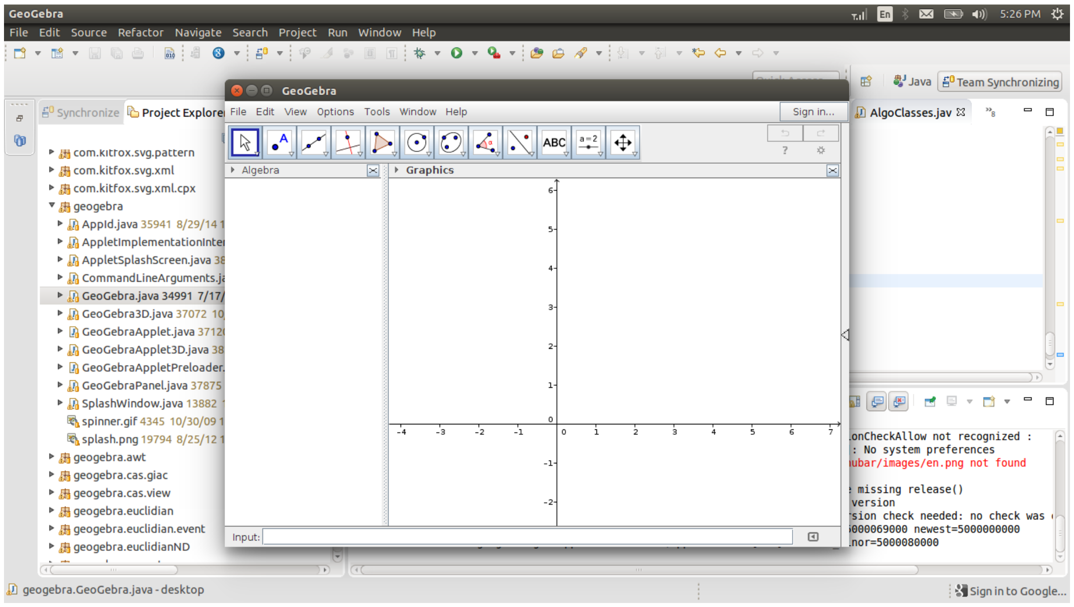
3. Pre-requisites

1. Download and compile GeoGebra (desktop + common) and GeoGebraWeb (web + common) [provide screenshots]

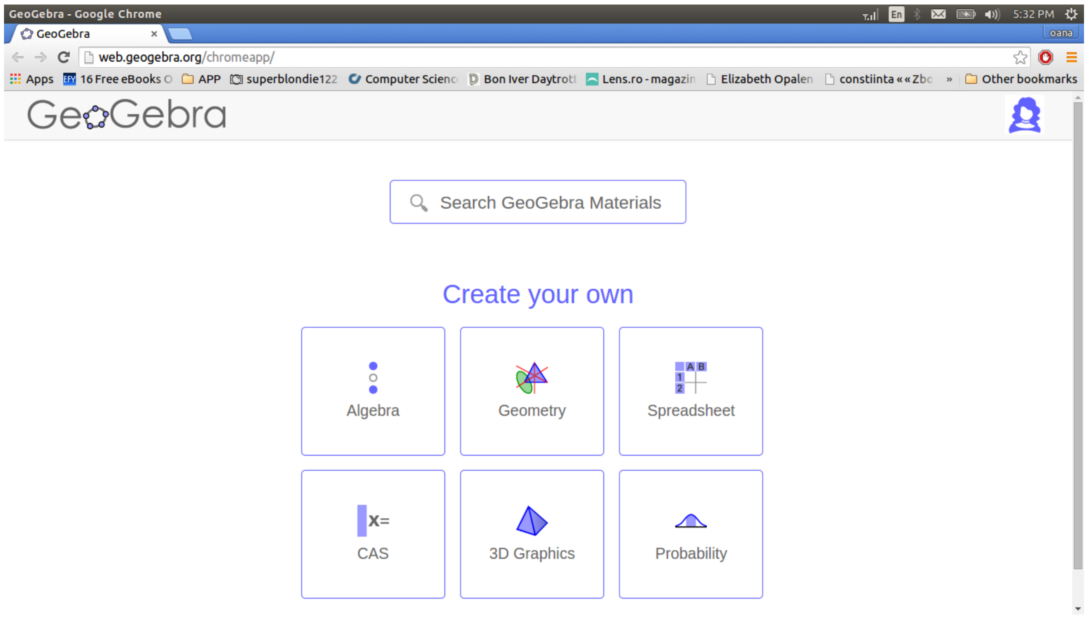


```java

    private void setListElement(int index, double value) {
        GeoNumeric listElement;
        if (index < classList.getCacheSize()) {
            // use existing element
            listElement = (GeoNumeric) classList.getCached(index);
        } else {
            listElement = new GeoNumeric(cons);
            listElement.setParentAlgorithm(this);
            listElement.setConstructionDefaults();
            listElement.setUseVisualDefaults(false);
        }

        classList.add(listElement);
        listElement.setValue(value);
    }

    // TODO Consider locusequability
```

```
<terminated> GeoGebra [Java Application] /usr/lib/jvm/jdk1.7.0_75/bin/java (Mar 25, 2015, 4:56:37 PM)
16:56:43.284 DEBUG: javax.swing.JMenu.fireMenuSelected[1043]: Menu opening: class geogebra.gui.menubar.FileMenuD
16:56:44.973 DEBUG: geogebra.main.AppD.addExternalImage[2106]: storing geogebra_thumbnail.png false
16:56:44.973 DEBUG: geogebra.io.MyXMLioD.readZip[154]: function ggbOnInit() {}

16:56:45.070 DEBUG: geogebra.io.MyXMLioD.processXMLBuffer[262]: MACROfalse
error in <spreadsheetView>: prefCellSize
16:56:45.079 DEBUG: geogebra.common.io.MyXMLHandler.startElement[407]: parsing start1427295405079
16:56:45.285 DEBUG: geogebra.common.io.MyXMLHandler.endElement[629]: cons tag end1427295405284
16:56:45.370 DEBUG: geogebra.gui.GuiManagerD.updateMenubar[953]: update menu
```



```
onCheckAllow not recognized :
: No system preferences
ubar/images/en.png not found

 missing release()
 version
rsion check needed: no check was
5000069000 newest=5000000000
nor=5000080000
```



2. Skype account

   Provided in the introduction at the beginning, but here it is again: oana.niculaescu

3. Good knowledge of Java (best way to show this is to make some patches, fix bugs etc)

   I'll attach the links to my patches to the proposal.

4. Show some understanding of what GWT is (we don't expect you to have used this before)

   I did not use GWT before, but I've looked over the pages provided on the development Wiki and I am trying to get up to speed with it.

5. Familiarity with the development wiki http://dev.geogebra.org/trac/wiki (I am working my way through it, read through it in order to be able to be consistent with GeoGebra development style, but I keep it close for future references.)