Geometric Modeling
Assigment sheet #0 (practice)
"Introduction to GeoX"

Silke Jansen, Ruxandra Lasowski,
Avinash Sharma, **Art Tevs**, Michael Wand

The solution of this assignment sheet is *voluntary*.
You **cannot** gain any points and the solution will **not** need to be presented.
The purpose is to ensure that the software is running correctly on your systems and that you get an impression of how the framework can and will be extended during the lecture.

Installation guide

The file geox.zip can be downloaded from the course web page:

http://www.mpi-inf.mpg.de/departments/d4/teaching/ss2010/geomod/index.html

It contains the necessary classes for the basic framework which is used for the practical assignments.
It is designed to run on Unix platforms as well as on Windows systems.
In order to run the main program you need to have QT 4.2.x or higher installed on your system.
If you use the Unix machines in the CS-CIP-POOL it is already installed.
For installation of QT on your private system we refer to:

http://qt.nokia.com/downloads

We provide a .proj file in the *windows* folder for users of MS Visual Studio and a .kdevelop file in the *unix* folder for KDevelop which is available on the student machines.

For the use on the CIP-machines all necessary paths should already be declared.
It requires to run *qmake* recursively on the top most folder followed by *build*.
Due to a linker issue it might be necessary to run *build* again.
To execute the program you have to open the *project options/run options* then you can change the executable path to the file in the geox/unix folder.

For use with MS Visual Studio you need to update the include folders.
Therefore open: *Tools/Options/Projects and Solutions/VC++ Directories*
Here you have to change the paths to your QT installation in
Executable files, Include Files and *Library Files.*
For execution under windows it might also be required to directly copy the requested QT-dll files to the folder which contains the executable file. (e.g.:/geox/windows/Debug)

Now the software should be ready for use on both platforms.

In the first exercise class on Wednesday 2nd and Friday 4th of May, we will be around with several teaching assistants and will help if there are remaining problems running the code.
A sample solution for the exercises listed below, will be provided before that date so that it can also be discussed.

In case of any problems, feel free to drop a mail to one of the teaching assistants.

Exercise 1:

Run Geox!

Explore the example experiments. After that, inspect the corresponding files in the experiments folder of the project. Try to become familiar with how the code and the result on the GUI are related. In the 2D example you can move the camera by using the left mouse button and zoom with the right one. You can manually add points and manipulate their positions, by using the different modes on the left.

Exercise 2:

Display a number of points on random positions between [-5,-5] and [5,5] which are of random color.
Therefore create a new class *Experiment02* derived from *Experiment*.
The class should provide a field *NumberOfPoints* that allows to determine how many new points should be added.
(like `ADD_INT32_PROP` in the example exercise)
Furthermore it should contain a command button to *addPoints* that actually triggers the action.
(like `ADD_NOARGS_METHOD` in the example exercise)
Include *Random.h* and use the function *rnd01()* in this experiment (it returns pseudo-random numbers between 0 and 1)
You should use a *GLGeometryViewer2D* to add the points and to display your results.
Do not forget to include and register your new experiment-class in the file:
InitGeoX.cpp

Exercise 3:

The two squares which are drawn in the ExampleExperiment2DGraphics (Button in the lower right) look similar but are of different nature.
The left one consist of 4 vertices which are connected. The right one consists of 4 independent lines. You can easily see the effect of this difference by picking vertices.
Create a new experiment. Draw a regular octagon that contains **exactly** 8 points which are connected by **exactly** 8 lines which is centered at the origin. If the connectivity is preserved when you drag and drop vertices then your solution is correct.
Add another method-button that allows you to reset the position of the 8 vertices to their initial value without introducing new points.