# PG612 Mandatory Assignment 1: Model Viewer

In this assignment, you will learn how to use quaternions to perform arbitrary rotations, load models from file, and render using interleaved arrays.

## Important information:

* **All subtasks are possible to implement independently. Do not get stuck on a single subtask. It is better to get 90% right of all than 100% right on one.**
* **You have to write all of the code yourself. Base your solution on the lecture notes and books in the course.**

### Subtask 1: Create a “virtual trackball”-viewer.

* Base your solution on the skeleton code.
* Implement your code as part of the VirtualTrackball class.
* Implement zoom in your viewer so that pgup and pgdown zoom in and out on the loaded model.

**Hints:**

* See lecture 2 for details on the trackball viewer.
* Use the projection matrix to zoom by changing the field of view of the viewing volume.

### Subtask 2: Load models using Assimp

* Implement loading of a model with positions and normals.
* Use an interleaved VBO.
* Load models from file using a filename given on the commandline: “GL32SDL.exe <modelname>”.
* Find a simple bounding box of the model by looping over the vertices.
* Find translation and scaling coefficients for the model so that its bounding box fits within the unit sphere.

**Hints:**

* Interleaved VBOs will be covered in lecture 3.
* Use argv[1] as the filename, and set as commandline argument in the visual studio debug options.
* Use std::numeric\_limits<float>::max() and - std::numeric\_limits<float>::max() (NOT min) as initial minimum and maximums when finding the bounding box.
* Use the model matrix to represent the translation and scaling

## Requirements

* Code must extend the skeleton from itsl
* Code must compile and run out-of-the-box using
  + Visual Studio 2010, SDL, GLM, Assimp, OpenGL 3.3+
* Short (5-25 lines, 80 columns) text (.txt) README-file
* Doxygen compliant source code comments (javadoc)
* No temporary files (Visual Studio, svn, etc.).

## Grading

The following criteria are used for grading this assignment:

* 40% Doing the assignment. For example:
  + Have you completed all subtasks?
  + Have you written every line of code yourself?
* 30% Code quality. For example:
  + Is your solution correct, simple, and elegant?
  + OpenGL efficiency,
  + Use of deprecated OpenGL functionality,
  + Useful comments,
  + Compilation errors, warnings, etc.
* 20% Visual quality and natural feel. For example:
  + Does your solution “feel” natural, does it look “right”, etc.
* 10% Overall rating. For example:
  + Anything I feel is not covered by the above points which deserves extra credit