# UD1414 — Group Project
# Real-time custom viewer for Maya

| | |
|---|---|
| Deadline | 30th of October 23:55 |
| Submission | Zip file uploaded to Its Learning |
| Grading | U/G |

## 1 Introduction

In this group project you will implement a real-time communication mechanism between Maya and an external application using message passing through shared memory. The external application will be a rendering engine with basic support for rendering such as transforms, meshes, lights, materials and camera movements.

## 2 Details

You have to implement a plugin in Maya that will track (using callbacks) the user's actions that affect the Maya scene and send some of these changes to another application using messages over shared memory. On the other end, a custom application will read the messages and extract the necessary information to create a similar representation of the Maya scene using its own rendering techniques. The interaction has to be in real-time, meaning that actions performed in Maya should be seen almost immediately in the custom renderer. If you already have a rendering application you can reuse the source code, otherwise you can also use existing rendering Engines such as [6][7][8][9].

You also have to reuse most of the work done in assignments 1 and 2.

### 2.1 Implementation details

For the messages, you have to think in how to construct them in the Maya plugin so that when the client (renderer) gets access to the messages they are ready to be used. For example, if a vertex buffer is used in the renderer, there will be a particular format that this vertex buffer uses (in the Input Assembly description), therefore it is advisable to arrange the information in the same format in the plugin and send it using the shared memory. This will allow for a rapid update or creation of resources on the rendering side.

The shared memory size will be fixed in compile time, so it will be a known value beforehand. You can pick a reasonable value and change it if necessary (100-200 megabytes)

The name for the filemap and the mutexes that you use will also be hard-coded and known by both the plugin and the viewer.

### 2.2 Minimum requirements

The following information is required to be sent and processed in the viewer

**Meshes** Vertex positions, normals and texture coordinates. Also for each mesh one transform matrix will be sent, and it should represent the accumulation of transformations applied to the mesh node.

**Materials** Materials applied to a mesh should be sent to the viewer. These includes Colours and Textures. You will have to extract this information from the shading groups attached to the node. You have to support at least one material per Mesh.

**Camera** The system should allow for camera attributes updates. The active camera in Maya (when presented in a single Viewport) should be the one used in the viewer at all times. If the user in Maya changes the camera view for example from Perspective to Top, the viewer should also change (including the type of projection – ortographic). Also if the user in Maya moves the camera around, the viewer should update the camera in real-time.

**Transformations** The Maya DAG can be quite complicated and contain many Transform nodes. You are only required to send one transform per mesh Node, which should represent the concatenation of transformations from that node up to the root of the DAG in Maya.

**Lights** If your viewer provides some form of lighting, you are not required to send any lighting information from Maya to the Viewer. Otherwise, you have to send at least one Point light from Maya to the real-time viewer.

## 3 Submission and grading

The assignment will be done in groups of 2 students and assessed through a short demonstration and explanation of the code by each group. Both participants have to divide equally the amount of work, and each participant will explain in a short presentation in which parts of the solution he was involved. Both participants have to fully understand all the code submited.

## References


[1] Assignments 1 and 2.

[2] Windows File Mapping. `http://goo.gl/w1xJgW`

[3] Using Mutex Objects. `http://goo.gl/fZlxA0`

[4] Useful notes on C pointers. `http://goo.gl/17VTZ0`

[5] Maya API documentation

[6] GamePlay3D – http://www.gameplay3d.io

[7] Panda3D – https://www.panda3d.org

[8] Ogre – http://www.ogre3d.org

[9] OpenSceneGraph – http://www.openscenegraph.org