MTAT.03.015 Computer Graphics (Fall 2013)
Exercise session XIV: OGRE

Konstantin Tretyakov, Ilya Kuzovkin

December 9, 2013

In this exercise session we will have a look at high-level graphics engine called OGRE[1]. We will see how the concepts we know about are included into the OGRE engine making our life easier: lighting, materials, shadows, environmental mapping and other techniques are made accessible by adding few lines of code, without the need to implement all annoying details on our own.

The solutions will have to be submitted as a zipped project directory. Please keep Windows libraries even if you work on Linux or Mac.

You can always seek for additional information and help in official tutorials `http://www.ogre3d.org/tikiwiki/tiki-index.php?page=Tutorials`: Basic Tutorials section.

1 Structure of the application

We start by comparing the structure of the application to the familiar structure we've been using so far. Please open `1_OgreTriangle` project and read through the code in the `triangle.cpp` file. Compare it to the GLUT-based applications we have seen before.

Exercise 1 (0.5pt). Add a small square which will fly around the triangle and rotate around it's own center. For that you will need to

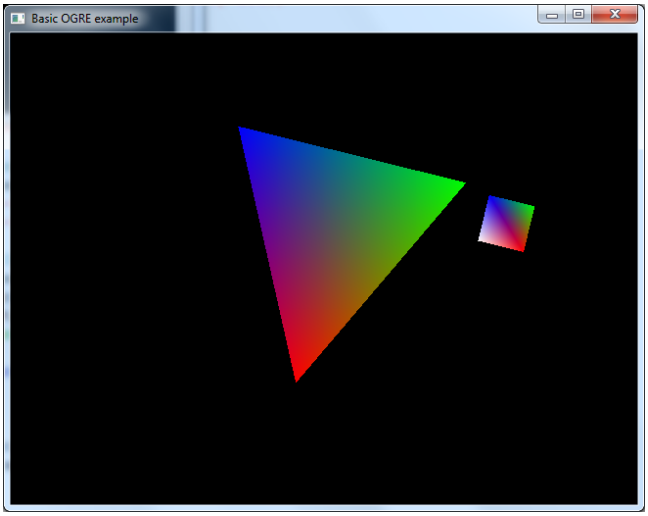
1. Create new `Ogre::ManualObject` object using `createManualObject()` method of the scene manager.
2. Describe vertices of your square. Look up in the documentation of the `Ogre::RenderOperation` class[2] which operation type you should use. Note that in OGRE we first create the vertex itself and then describe it's attributes.
3. Create `Ogre::SceneNode` and attach the new object to it.

[1] `http://www.ogre3d.org/`

[2] `http://www.ogre3d.org/docs/api/html/classOgre_1_1RenderOperation.html`

4. Use this `SceneNode` to animate our object (update it's potision) in the `frameRenderingQueued()` method, which is an analog of `idleFunc()` in GLUT.

The existing code for the triangle will serve you as example. The result should look something like this:



2 High-levelness

Open project `2_OgreLight`. The structure is same as before. Have a look at `createLitSphereScene()`, here we create a sphere and add materials and lighting to it. See how it is done. Now pay attention to these lines in the middle of `run()` function:

```
Ogre::ResourceGroupManager::getSingleton().
                    addResourceLocation("../data", "FileSystem");
Ogre::ResourceGroupManager::getSingleton().
                    initialiseAllResourceGroups();
```

First line tells OGRE where to look for resources: data files and *scripts*. Second line instructs it to read in resource descriptions and initialise *resource groups*. Have a look at the file `Examples.material` in the `data` folder and try to apply some of materials described there to our sphere:

```
sphere->setMaterialName("Examples/WaterStream");
```

If you would like to try other examples you should download Ogre SDK[3] and add the resources needed for each particular example to our `data` folder.

[3] `http://www.ogre3d.org/download/sdk`

Exercise 2 (0.5pt). Create a new file `data/Sphere.material`. Use `data/Examples.material` and `http://www.ogre3d.org/docs/manual/manual_16.html` to create a material script, which reproduces exactly same material settings as we have in our code. The result should look exactly the same: red sphere with white specular spot on it.

Exercise 3* (0.5pt). Texture

Exercise 4 (0.5pt). Add a plane and a stencil shadow

Open project `3_OgreMesh`

Exercise 5 (1pt). Skybox with sky texture. Or any other texture (are there many of them in OGRE?)

Exercise 6* (0.5pt). Make mesh reflective. So that it will reflect this skybox

3 Plugins

Open project `4_OgrePlugins`. You can switch scenes, look how it's done.

Exercise 6 (0.5pt). Do ??? with particle system

Exercise 6* (1pt). Add some other plugin (or any other display of creativity?)