HÁSKÓLINN Í REYKJAVÍK
REYKJAVIK UNIVERSITY

Haustönn 2014

Tölvugrafík

T-511-TGRA

Final Report

Ari Freyr Ásgeirsson
Sigursteinn Bjarni Húbertsson

6. nóvember 2014

Kennari: K-machine

Introduction

Step in side the Tardis and prepare yourself for an epic adventure, but don't worry.... it's bigger on the inside. We take you on a journey through our solar system, visiting all the beautiful places you always wanted to go to! Our first destination is the smoldering hot Mercury and from there we thread the system going from planet to planet.
You will get a good look at all the planets and some of their moons and if all goes according to the plan we end our journey at the cold outer rim of our system where see Neptune in all his glory, if you expected to see Pluto we are sorry to disappoint you. We hope you enjoy the this epic space trip and thank you for choosing the Tardis.

Problems and solutions

The first problem we bumped into was the background for our project. We first just tried to put a texture as a background for the whole project and as we later found out did not work for us. The second attempt was to create a sky box but after some time of fine tuning we saw that it was horrible and flushed the idea. Finally we just created a giant sphere that we called space and put some nice texture map on it and voila, we created the universe.
The second problem was importing objects in our project as libgdx is very unclear in its documentation but it was solved through trial and error and long hours.
The thrid problem was clipping. Because our space is so huge the far plane clipping starts to cut our view after certain distance. We solved this by finding the ideal size for the space Sphere and found out the space can't be

scaled more than 25.000, any bigger than that and we run into problems, so space it self moves as our ships moves to prevent the clipping from happening.

Program implementation

Since the we have been using LibGDX 0.9.8 throughout the course we decided to stick to it. Here is a list of all the functions we used from the library.

- com.badlogic.gdx.graphics.Texture
- com.badlogic.gdx.graphics.g3d.model.Model
- com.badlogic.gdx.graphics.g3d.loaders.wavefront.ObjLoader
- com.badlogic.gdx.graphics.GL11
- com.badlogic.gdx.utils.BufferUtils
- com.badlogic.gdx.audio.Music
- com.badlogic.gdx.Input
- com.badlogic.gdx.backends.lwjgl.LwjglApplicationConfiguration
- com.badlogic.gdx.graphics.g3d.ModelLoaderHints

Here is a list of all the functions we used from OpenGL

glEnable, glDisable	**Enables or disables OpenGL capabilities**
glShadeModel	**Selects flat or smooth shading.**
glClearColor	**Specifies clear values for the color buffers**
glMatrixMode	**Specifies which matrix is the current matrix**
glLoadIdentity	**Replaces the current matrix with the identity matrix**
gluPerspective	**Sets up a perspective projection matrix.**
glEnableClientState	**Enables and disables arrays, respectively**
glClear	**Clears buffers to preset values**
glMaterial	**Specify material parameters for the lighting model**
glLight	**Set light-source parameters**
glPushMatrix, glPopMatrix	**Push and pop the current matrix stack, respectively**
glScale	**Multiply the current matrix by a general scaling matrix**
glRotate	**Multiply the current matrix by a rotation matrix**
glTranslate	**Multiply the current matrix by a translation matrix**
glVertexPointer	**Defines an array of vertex data**
glNormalPointer	**Defines an array of normals**
glTexCoordPointer	**Defines an array of texture coordinates**

Since we were using OpenGL almost exclusively we had to implement a lot of basic functionality like bezier curves and camera motion. We made a custom implementation of the bezier curve which you can see below:

Listing 1: Bezier Curve implementation

```
private void bezierCurve(Point3D start, Point3D c1,
Point3D c2, Point3D end, float t) {
   float x = (float) (Math.pow((1 - t), 3) * start.x +
         3 * Math.pow((1 - t), 2) * t * c1.x +
         3 * (1 - t) * Math.pow(t ,2) * c2.x +
         Math.pow(t, 3) * end.x);

   float y = (float) (Math.pow((1 - t), 3) * start.y +
         3 * Math.pow((1 - t), 2) * t * c1.y +
         3 * (1 - t) * Math.pow(t ,2) * c2.y +
         Math.pow(t, 3) * end.y);

   float z = (float) (Math.pow((1 - t), 3) * start.z +
         3 * Math.pow((1 - t), 2) * t * c1.z +
         3 * (1 - t) * Math.pow(t ,2) * c2.z +
         Math.pow(t, 3) * end.z);

   this.position.x = x;
   this.position.y = y;
   this.position.z = z;
}
}
```