Assignment 2 : A 3D World

Due: Sunday 22th Oct , 23:59:59 (The end of week 12)

Late submissions will not be accepted after midday tuesday week 13.

Demo: Week 13

Marks: 25 plus up to 4 Bonus Marks

Assignment 2 Description

For the second assignment you will be building a 3D world .

This is an assignment you should do in pairs. Pairs should be chosen from the same tutorial class if possible. You will be performing a demo of you work in week13 during the tutorial time or the hour after. You may do the assignment individually if you choose to.

The aim of this assignment is to test:

- Your ability to work with **3D transformations**
- Your ability to **generate meshes** from mathematical descriptions of objects.
- Your ability to **render 3D scenes** with a **perspective camera**
- Your ability to use OpenGL **lighting**
- Your ability to use **textures** and **MIP mapping**.

Furthermore, the assignment is open-ended, allowing you to make additional improvements of your own choice.

Task

Your task is to complete the implementation of a 3D world. In this world you control a camera moving around a landscape which includes trees, hills and roads.

Files

Download a set of base classes here. These classes implement the basic data-structures, but are incomplete. The files provided are:

- **Game.java** - this is the main entry point to your game
- **Terrain.java** - this class represents variable height terrain.
- **Tree.java** - this class represents a tree
- **Road.java** - this class represents a road as a bezier curve

- **LevelIO.java** - this class reads and writes game levels to and from JSON files.

There is also a org.json file library for level I/O. The level files are fairly easy to read and change by hand.

You are free to change any of these files as you see fit. We will not be testing individual functions. However you should make sure that the established Level IO format works for your code, because we will be testing your level with standard terrain files.

Game

This is the main class for your game. The main() method in this class will be used to test your game. It expects a single string specifying the name of the level file. If you want to specify any other parameters they should be part of the JSON file.

Terrain

The terrain is represented as a grid. The width and height of the grid are specified in the level file. Each point in the grid has a specified altitude. **Your first task** is to draw the terrain as a mesh of triangles with vertices at each of the grid points with the corresponding altitude.

You can treat X,Z and altitude as OpenGL coordinates. They should all have the same scale. Test maps will be of the order of 10x10 to 20x20. Maximum altitudes will be in a similar range (10-20).

A 2x2 terrain with altitudes:

Note: the bold labels (x0,x1,z0,z1) are just to explain what the values mean and will not actually be part of the data

	x0	**x1**
z0	0	0.5
z1	0	0.3

A 2x2 terrain represents 4 vertices. The altitudes correspond to the Y values for the x,z co-ordinates.

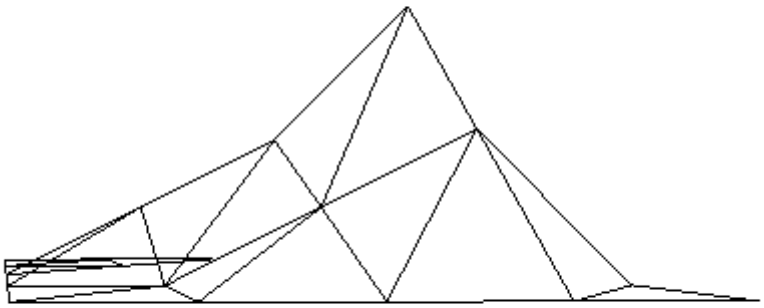
Will create a mesh withe the following co-ordinates

A 5x5 terrain with altitudes:

0	0	0	0	0
0	0	0.5	1	0
0	0.5	1	2	0
0	0	0.5	1	0
0	0	0	0	0

Will create a mesh that looks like this (this may look different depending on the angle/position you view it from and exactly how you set up your camera). This is taken from basically straight ahead at around (0,0.5,9) in world co-ordinates and with a perspective camera:

Note: This screenshot was taken with back face culling on



Note: for the assignment you will shade and texture your terrain and make it look pretty. We are just showing the lines here so you can clearly see the geometry. It is up to you whether you implement your terrain using face nomals or vertex normals.

Trees

The levels include trees at different points on the terrain. **Your second task** is to draw the trees at the specified locations.

For the base version of this project, a tree should be a simple mesh with a cylidrical trunk and and a sphere of leaves. If you want you can make your 'trees' more exotic: lampposts, candy-canes, chimneys, or whatever your imagination dictates. The point is that they are placeable 3d models on the terrain.

Note that the level descriptions only specify the (x,z) location of the tree. You will need to use the terrain data to calculate the altitude of the tree and draw it appropriately. Trees are not

guaranteed to be positioned at grid points, so you will need to interpolate altitude values if a tree is in the middle of a triangle.

Road

The level include roads. Each road is described as a 2D Bezier curve. I have provided a function for you which calculates the (x,z) location of points along the road. **Your third task** is to use this function to extrude a road which follows this curve, with the width specified in the constructor.

You can assume, for the base portion of the assignment, the roads will only run over **flat terrain**, so you will not have to handle going up or down hills.

Camera

You should implement a 3D camera which **moves** around the scene using the arrow keys (ie. the camera's position changes with respect to the world):

- The up arrow moves the camera forward (in the current direction)
- The down arrow moves the camera backward (relative to the current direction)
- The left arrow turns(rotates) left (ie changes the direction that the camera is facing)
- The right arrow turns(rotates) right (ie changes the direction that the camera is facing).

The camera should move up and down **following the terrain**. So if you move it forward up a hill, the camera should move up the hill and not go through it! (Note: it is ok for the camera to go through trees and other objects for this assignment).

The camera should be a **3D perspective** camera with a reasonable field of view. The aspect ratio should **match the aspect ratio** of the viewport.

Lighting

You should render the scene with appropriate **materials and lights**. In the base version you should at least have a single light source representing the sun. The terrain, trees and road should all have suitable materials.

The level files include a "sunlight" field which is a 3D vector specifying a directional light to be included in the scene. The vector represents the direction **to** the source of the light.

Texturing

You should **texture** all the models (terrain, road, trees,avatar,others) in the scene, using **MIP maps** to handle minification. You may use whatever textures you feel suitable. Be creative. Make everything look like Lego or an ice sculpture or origami.

Avatar

Add an avatar and make the camera follow behind the avatar in a 3rd person view. You should being able to switch from 1st person (with no avatar) to 3rd person (with the avatar) by pressing a key of your choice. For the base part the avatar does not need to be a complex model (a glut teapot is ok).

The Other/s

You should extend the scene level language to include the location/s of an enemy/ally/wild beast etc. You must use VBOs instead of immediate mode for these models and use a vertex and fragment shader to render them.

Extensions

The base elements described above are worth 19 of the 25 marks. For the remaining 6 marks you can choose among the following extensions:

- Build a complex model or a model with walking animation or something beautiful or interesting for your avatar or your others! (2..4 marks)
- Write the whole assignment using VBOs and shaders. (3 marks)
- Add a 'night' mode with low ambient lighting. Give the player a torch which shines in the direction they are facing. (2 marks)
- Make the sun move and change colour according to the time of day (2 marks)
- Add rain using particle effects (4 marks)

  For the full marks this would need to include alpha blended billboarded particles, creation and destruction ,some kind of evolution over time (position, size, colour, as is appropriate for your kind of particles).

- Add ponds with animated textures to your world (4 marks)

  Ponds need only lie on flat terrain like roads but should include animated textures showing ripples or waves.

- Add an L-system for fractal tree generation (4 marks)

  To get full marks for this you would need to implement a proper rewite system. You would not need to load the grammar for the L-system from JSON, but it should be possible to alter the grammar just by changing values in the code.

  You should also provide a way to increase/ decrease the number of iterations either interactively or from reading in the number of iterations from a json file. By default you should set it to the number of iterations that looks best/runs best. It does not matter if the tree does not look as good when iterations are increased/decreased. It is also ok if performance drops for high numbers of iterations. This is to be expected.

- Fix road extrusion so roads can go up and down hills (4 marks).
- Add shadows to the trees and terrain (4 marks)
- Add shaders to implement normal mapping on one of your models(4 marks)

- Add reflection mapping (using cube mapping) to one of your models (4 marks)
- Add shaders to implement NPR shading (2..6 marks)
- Add level-of-detail support for rendering distant objects with lower-resolution models (2..6 marks)
- BSP trees for hidden surface removal for terrain rendering (8 marks)
- Add Portal style portals.
    - Portals you can walk through (4 marks)
    - Portals you can walk and see through (8 marks)
- Implement the terrain as Bezier or NURBS surfaces (8 marks)

If you have other ideas for extensions please ask on the forum. If there are any I like, I will add them to the list. I'm looking for extensions which test your use of different rendering techniques rather than just adding more stuff to the world.

Note: The marks above increase roughly logarithmically with the amount of work required. So a task worth 6 marks is about 16 times harder than a task worth 2 marks.

Marking

This assignment is worth 25% of your final mark. Marks are assigned as follows:

Item	Marks
Terrain - mesh generation	2
Terrain - interpolating altitudes	1
Trees - mesh generation	2
Road - mesh generation	3
Camera - perspective projection	1
Camera - movement	1
Lighting	2
Textures	2
Avatar	2

Item	Marks
The others	3
Extensions	10

The extension element includes **4 bonus marks**, so the maximum possible mark is 29/25.

Marks beyond 25 will only be awarded if all the base components work appropriately. You can not make up for marks lost in the core component with marks from the extension component. For example, if you lose 2 marks in the core component, the most you can get is 23/25.

Submission

Submit a **single JAR file** containing all your Java source and any addition files needed to make your project work.

Notes:

- The main entry point to your game should be the **Game.main()** method. This method is expected to take the name of a level file and play that level. Any additional parameters should be incorporated into the level file.
- Your game will be tested on some **standard level files**, so you should support the provided file format.
- You should include **additional level files** which demonstrate any particular extensions you have implemented.
- You should include a **README** text file explaining what extensions you have implemented and where they can be found it the code.
- You should **comment your code** thoroughly so your marker can understand what you have implemented and how it works.

Submit your JAR file using CSE give, either using webcms or the command line:

```
% give cs3421 ass2 Ass2.jar
```

Late submissions will lose 2.5 marks per day from the maximum possible mark.

Submissions will not be accepted after midday tuesday week 13.

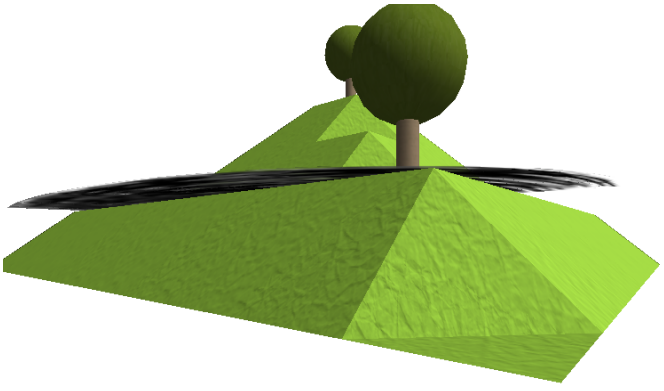
Example world

Sample world file: Test 1

```
{
    "width" : 10,
        "depth" : 10,
```

```
        "sunlight" : [ -1, 1, 0 ],

        "altitude" : [
            0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
            0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
            0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
            0, 1, 1, 2, 2, 2, 2, 1, 1, 0,
            0, 1, 1, 2, 3, 3, 2, 1, 1, 0,
            0, 1, 1, 2, 3, 3, 2, 1, 1, 0,
            0, 1, 1, 2, 2, 2, 2, 1, 1, 0,
            0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
            0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
            0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        ],

        "trees" : [
                {
                        "x" : 1.5,
                        "z" : 1.5
                },
                {
                        "x" : 4.5,
                        "z" : 4.5
                }
        ],

        "roads" : [
                {
                        "width" : 3,
                        "spine": [
                                1.5, 3,
                                1.5, 0.5,
                                7.5, 1.5,
                                7.5, 1.5
                        ]
                }
        ]

}
```

- The screenshot



is the shot of Test 1. The road goes over the edges of the terrain.

You can see more examples here.