Assignment 2: A 3D Game

(https://www.openlearning.com/unsw/courses/COMP3421-2013S2/Activities/Assignment2A3DGame)

IMPORTANT NOTE:

Students trying to submit today have faced an error message:

```
classrun: cannot create submission directory "/import/adams/1/cs3421/13s2.work/ass2/wed17a/3373157"
 login: aeli748
 stuID: 3373157
 class: cs3421
NO submission made.
```

This is happening because the class account has run out of disk quota. SS are in the process of fixing this. Please try to submit again in a little while.

Update 21/10: I didn't make the interaction between bonus marks and late marking very clear. It wasn't my original intention but for the sake of fairness I will say that late marks are take off the 24 mark cap. So if you submit today (Monday) you can still get up to 22 marks.

Update 16/10: I've written an FAQ (https://www.openlearning.com/unsw/courses/COMP3421-2013S2/Activities/Assignment2A3DGame/Faq) for a lot of the common questions I'm being asked.

Update 11/10: I have created some example maps and screenshots (https://www.openlearning.com/unsw/courses/COMP3421-2013S2/Activities/Assignment2A3DGame/Examples).

Update 10/10: Note that the sunlight vector in the file below is the vector **FROM the sun.** However in specifying a directional light in OpenGL you want to specify the vector **TO the sun**. So you should reverse the sign of all three components. My apologies for causing this confusion.

Update 27/9: Fixed a bug in LevelIO - the width of roads was not being properly specified. This also meant some changes for Terrain.addRoad() and the Road constructor. The width of a road should be a double, not an int.

Update 27/9: Changed LevelIO.java and Terrain.java to include a "sunlight" field. This field specifies the direction of the sunlight in your world as a 3D vector. You should include a directional light which points along this vector.

Update 25/9: Changed LevelIO.java to use "depth" and "z" instead of "height" and "y", to reduce ambiguity. Also fixed a bug in the save() method (trees were output with their y position (altitude), not their z position).

For the second assignment you will be implementing a 3D game.

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 game. In this game you control a camera moving around a landscape which include trees, hills and roads.

Files

Download a set of base classes (http://www.cse.unsw.edu.au/~cs3421/assignments/ass2.zip) 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 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. I am in the process of writing a small level editor to assist with creating levels.

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 on 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.

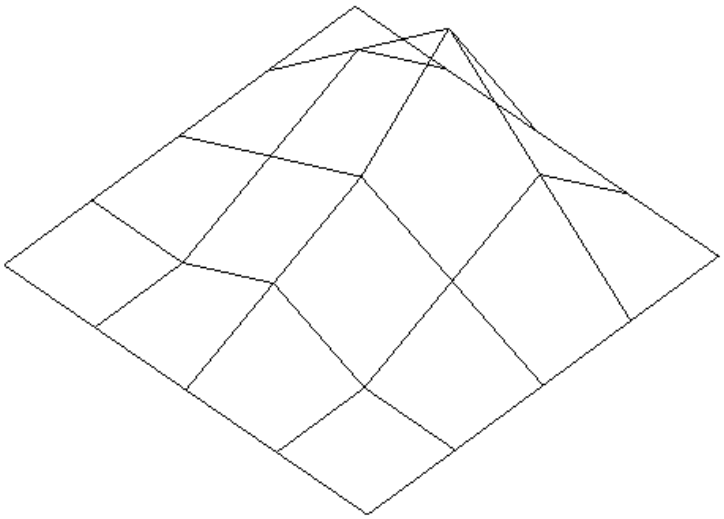
Terrain

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. **You first task** is to draw the terrain as a mesh of quads with vertices at each of the grid points with the corresponding altitude.

So 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 like this:



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 quad.

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:

- The up arrow moves forward
- The down arrow moves backward
- The left arrow turns left

- The right arrow turns right.

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.

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.

Texturing

You should **texture** all the models (terrain, road, trees) 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 ice sculpture or orgami.

Extensions

The base elements described above are worth 14 of the 20 marks. For the remaining 6 marks you can choose among the following extensions:

- Add an avatar and make the camera follow behind the avatar in a 3rd person view (2 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 shaders to implement Phong shading (with textures) (2 marks)
- Add rain using particle effects (4 marks)
- 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 NPR shading (4-6 marks)
- Add an L-system for fractal tree generation (6 marks)
- BSP trees for terrain rendering (6 marks)
- Add level-of-detail support for rendering distant objects with lower-resolution models (6 marks)
- Implement the terrain as Bezier or NURBS surfaces (8 marks)

If you have other ideas for extensions please post them in the comments below. 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 20% 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
Extensions	10

The extension element includes **4 bonus marks**, so the maximum possible mark is 24/20.

Marks beyond 20 will only be awarded if all the base components work appropriately.

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 the web interface (https://cgi.cse.unsw.edu.au/~give/code/login.php?app=/~give/Student/give.php) or the command line:

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

Submissions are due by **11:59 Sunday October 20** (the end of week 11).

Late submissions will lose 2 marks per day from the maximum possible mark.

Example world

Sample world file:

```
{
    "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
                        ]
                }
        ]

}
```

You can more more example here (https://www.openlearning.com/unsw/courses/COMP3421-2013S2/Activities/Assignment2A3DGame/Examples).