COMP3170 Assignment 1 Report

Name	Joshua Brookes
Student ID	4360 3467

Your development environment

Please record your eclipse settings and your software & hardware configuration below.

Java JDK version used for compilation	jdk-13.0.2
Java compiler compliance level used for compilation	13
Java JRE version used for execution	13
Eclipse version	Oxygen.3a Release (4.7.3a)
Your screen dimensions (width x height)	1000 x 1000
Your computer type (Mac/PC)	Desktop PC
Your computer make and model	CPU: Intel i5 - 6500 GPU: GTX 1070 Motherboard: Gigabyte B150M-D3H RAM: Kingston 16gb DDR4
Your computer Operating System and version	Windows 10 64-bit (Build 18362)

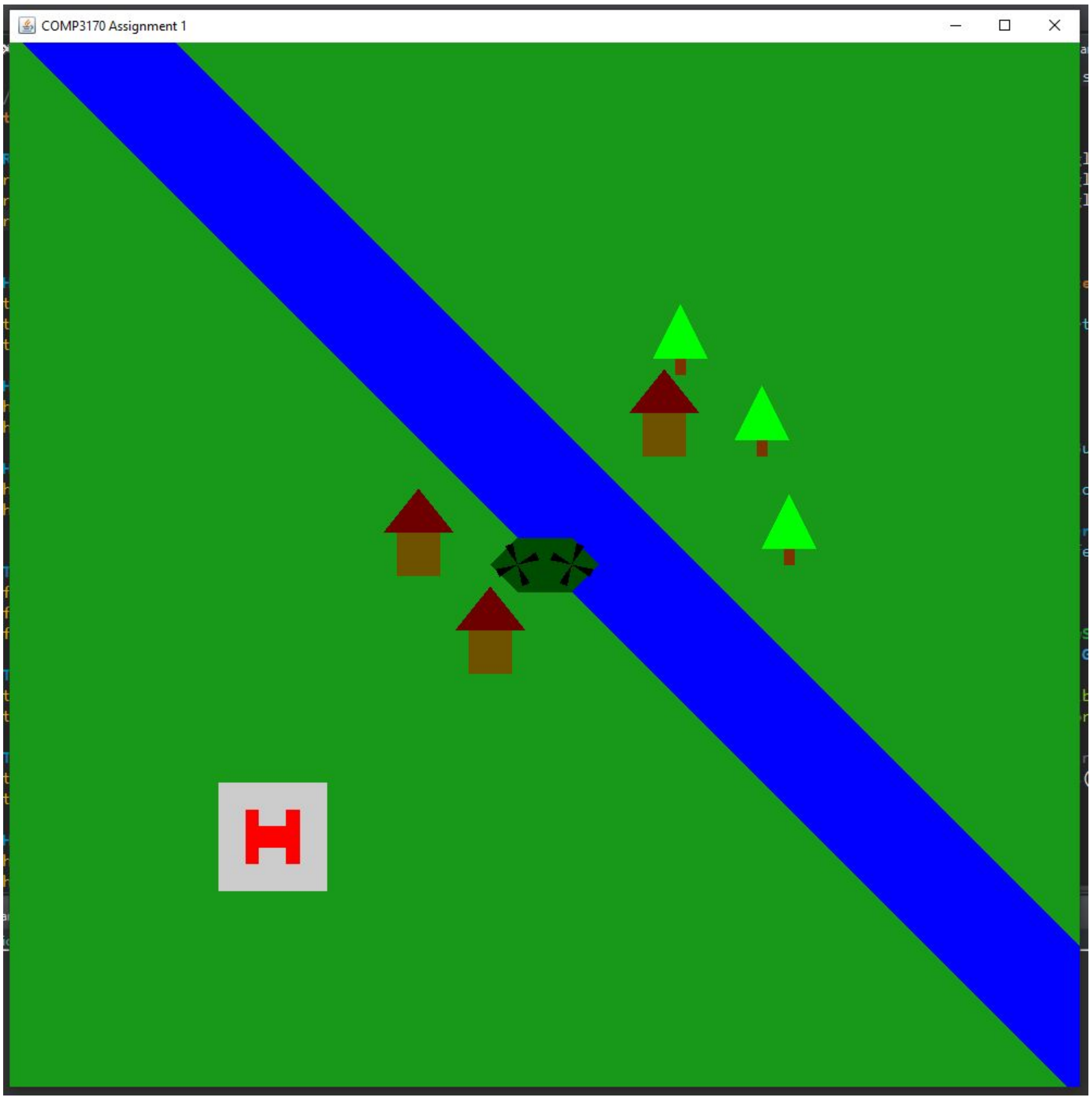
Your program features for marking

Features to be marked in this assignment. In addition to the required features, select at most three of the optional features for a total mark of 100%.

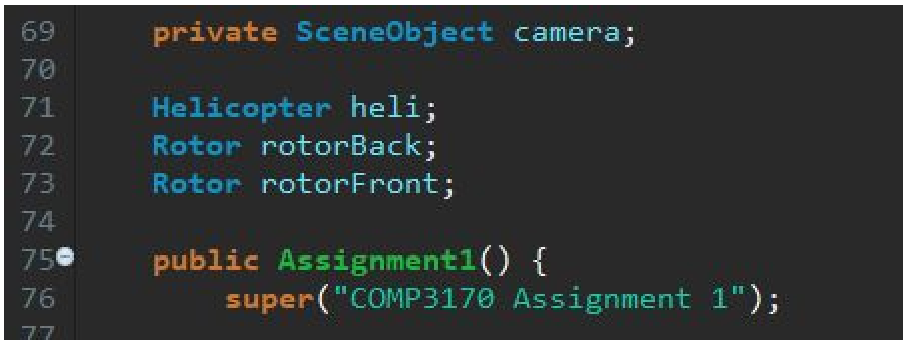
Feature	**Mark**	**Indicate "Yes" if feature is to be marked**
Static 2D terrain: Town, trees, river, helipad	40%	Required - Yes
Moving helicopter with keyboard control	30%	Required - Yes
Helicopter with spinning tandem rotors	10%	Yes
Resizing the canvas, maintaining resolution	10%	Yes
Control helicopter with the mouse	10%	
Take-off and landing at the helipad	10%	
Camera mounted on the helicopter	10%	
Minimap	10%	
Curved rivers	10%	Yes
Heads up display	10%	
Forest using instancing	10%	
TOTAL (max 100%)		

On the following pages you should indicate where each of the above features appear in your program, using screenshots and filenames/line-numbers to indicate where it occurs in your project. Include relevant Java source and shader source file names.
You will not get marks for a feature if your marker cannot easily locate it within your world.

Static Terrain

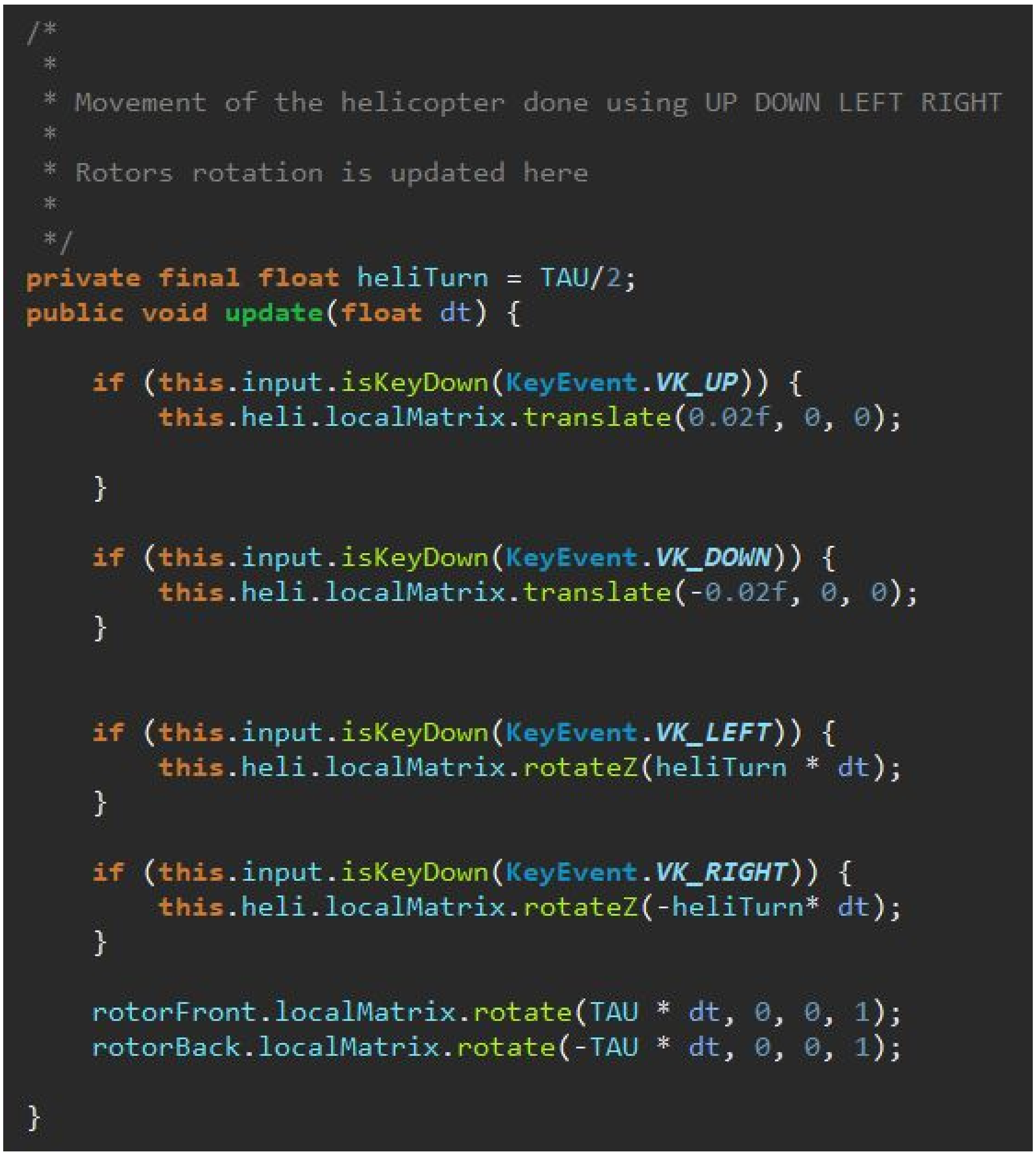


- Assignment1.java:71-73 - Helicopter and rotors first created
    - this was created here so that the update function can access them



```
69	private SceneObject camera;
70
71	Helicopter heli;
72	Rotor rotorBack;
73	Rotor rotorFront;
74
75	public Assignment1() {
76		super("COMP3170 Assignment 1");
```

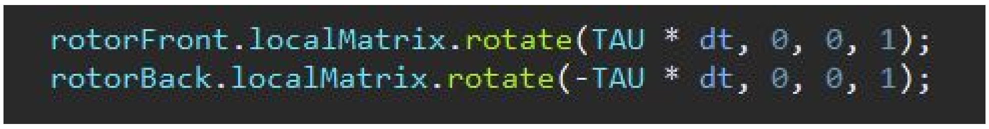
- Assignment1.java:135-185 - Scene created in world space. This includes houses, trees, helipad and river
    - House.java
    - Helicopter.java
    - HeliPad.java
    - River.java
    - Rotor.java
    - Tree.java
- Assignment1.java:199-223 - Helicopter movement updates



```java
/*
 *
 * Movement of the helicopter done using UP DOWN LEFT RIGHT
 *
 * Rotors rotation is updated here
 *
 */
private final float heliTurn = TAU/2;
public void update(float dt) {

    if (this.input.isKeyDown(KeyEvent.VK_UP)) {
        this.heli.localMatrix.translate(0.02f, 0, 0);

    }

    if (this.input.isKeyDown(KeyEvent.VK_DOWN)) {
        this.heli.localMatrix.translate(-0.02f, 0, 0);
    }


    if (this.input.isKeyDown(KeyEvent.VK_LEFT)) {
        this.heli.localMatrix.rotateZ(heliTurn * dt);
    }

    if (this.input.isKeyDown(KeyEvent.VK_RIGHT)) {
        this.heli.localMatrix.rotateZ(-heliTurn* dt);
    }

    rotorFront.localMatrix.rotate(TAU * dt, 0, 0, 1);
    rotorBack.localMatrix.rotate(-TAU * dt, 0, 0, 1);

}
```

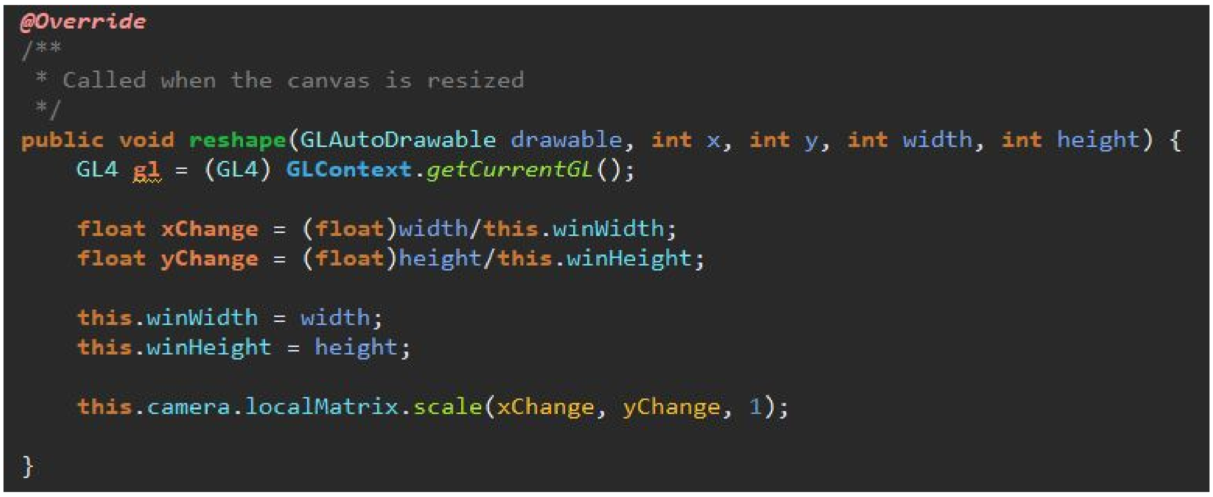
- Assignment1.java:220-221 - Helicopter with spinning tandem rotors

  - rotate in opposite directions. Just like real life!



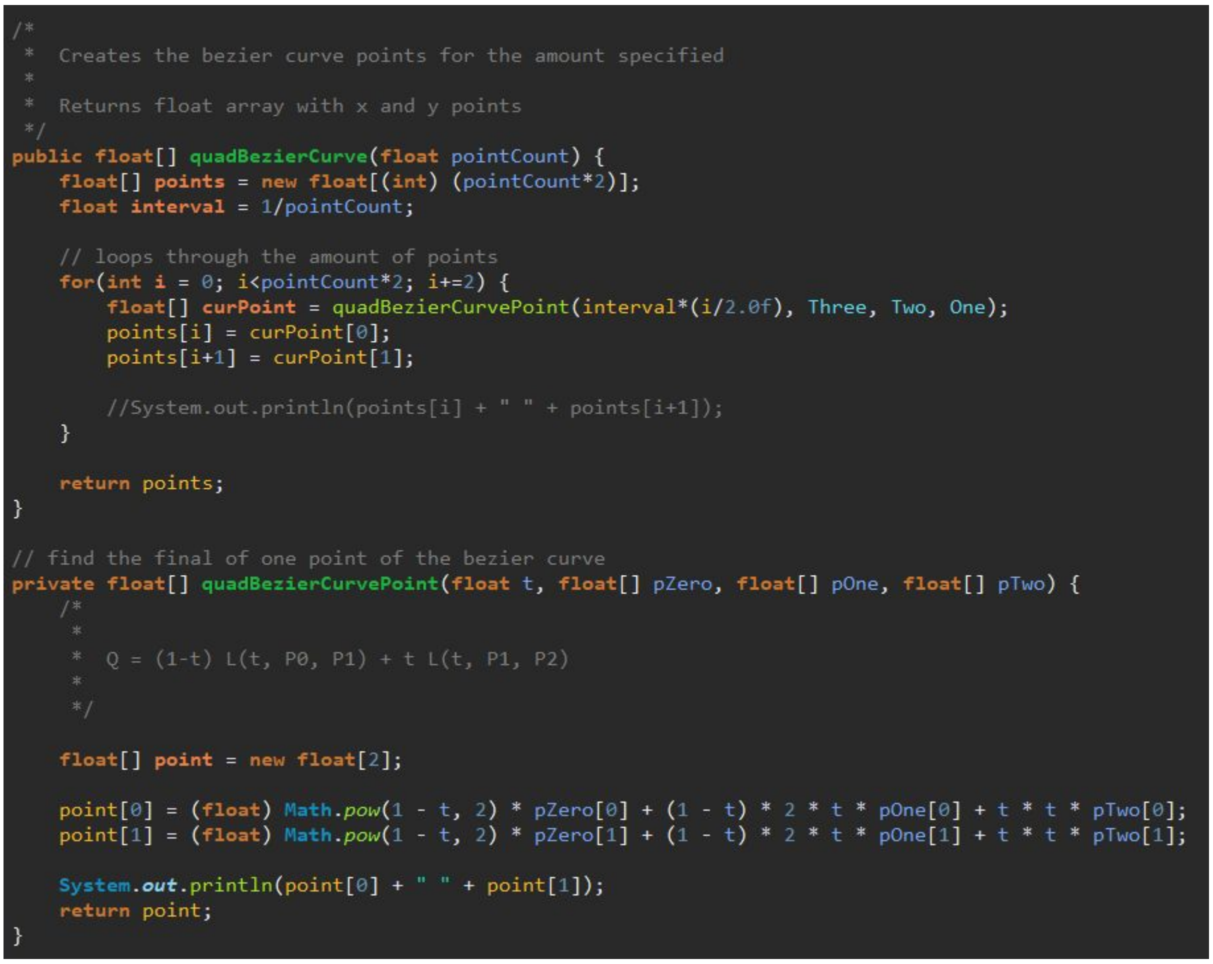
```
rotorFront.localMatrix.rotate(TAU * dt, 0, 0, 1);
rotorBack.localMatrix.rotate(-TAU * dt, 0, 0, 1);
```

- Assignment1.java:257-268 - Resizing the canvas, maintaining resolution
  - finds the ratio of change when window reshaped then scales the camera matrix by that amount



```
@Override
/**
 * Called when the canvas is resized
 */
public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height) {
    GL4 gl = (GL4) GLContext.getCurrentGL();

    float xChange = (float)width/this.winWidth;
    float yChange = (float)height/this.winHeight;

    this.winWidth = width;
    this.winHeight = height;

    this.camera.localMatrix.scale(xChange, yChange, 1);

}
```

- River.java:42-73 - Bezier Curve
    - to test uncomment/comment
        - River.java:32 - comment
        - River.java:32 - uncomment
        - River.java:84 - comment
        - River.java:85 - uncomment
        - Assignment1.java:138-139 - comment



```
/*
 *  Creates the bezier curve points for the amount specified
 *
 *  Returns float array with x and y points
 */
public float[] quadBezierCurve(float pointCount) {
    float[] points = new float[(int) (pointCount*2)];
    float interval = 1/pointCount;

    // loops through the amount of points
    for(int i = 0; i<pointCount*2; i+=2) {
        float[] curPoint = quadBezierCurvePoint(interval*(i/2.0f), Three, Two, One);
        points[i] = curPoint[0];
        points[i+1] = curPoint[1];

        //System.out.println(points[i] + " " + points[i+1]);
    }

    return points;
}

// find the final of one point of the bezier curve
private float[] quadBezierCurvePoint(float t, float[] pZero, float[] pOne, float[] pTwo) {
    /*
     *
     *  Q = (1-t) L(t, P0, P1) + t L(t, P1, P2)
     *
     */

    float[] point = new float[2];

    point[0] = (float) Math.pow(1 - t, 2) * pZero[0] + (1 - t) * 2 * t * pOne[0] + t * t * pTwo[0];
    point[1] = (float) Math.pow(1 - t, 2) * pZero[1] + (1 - t) * 2 * t * pOne[1] + t * t * pTwo[1];

    System.out.println(point[0] + " " + point[1]);
    return point;
}
```

- The bezier curve didn’t get working the way I wanted due to some strange value which I posted about in the forums, but it's some progress.
- You can also adjust the amount of points found by changing value on line 16 in River.java