COMP3170 Assignment 1 Report

Name	Itai Ktalav
Student ID	43681913

Your development environment

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

Java JDK version used for compilation	13.0.2
Java compiler compliance level used for compilation	13
Java JRE version used for execution	1.8.0_60
Eclipse version	2020-03 (4.15.0)
Your screen dimensions (width x height)	2560 x 1440
Your computer type (Mac/PC)	PC
Your computer make and model	Custom Desktop
Your computer Operating System and version	Windows 10 Education 1803 64bit

Your program features for marking

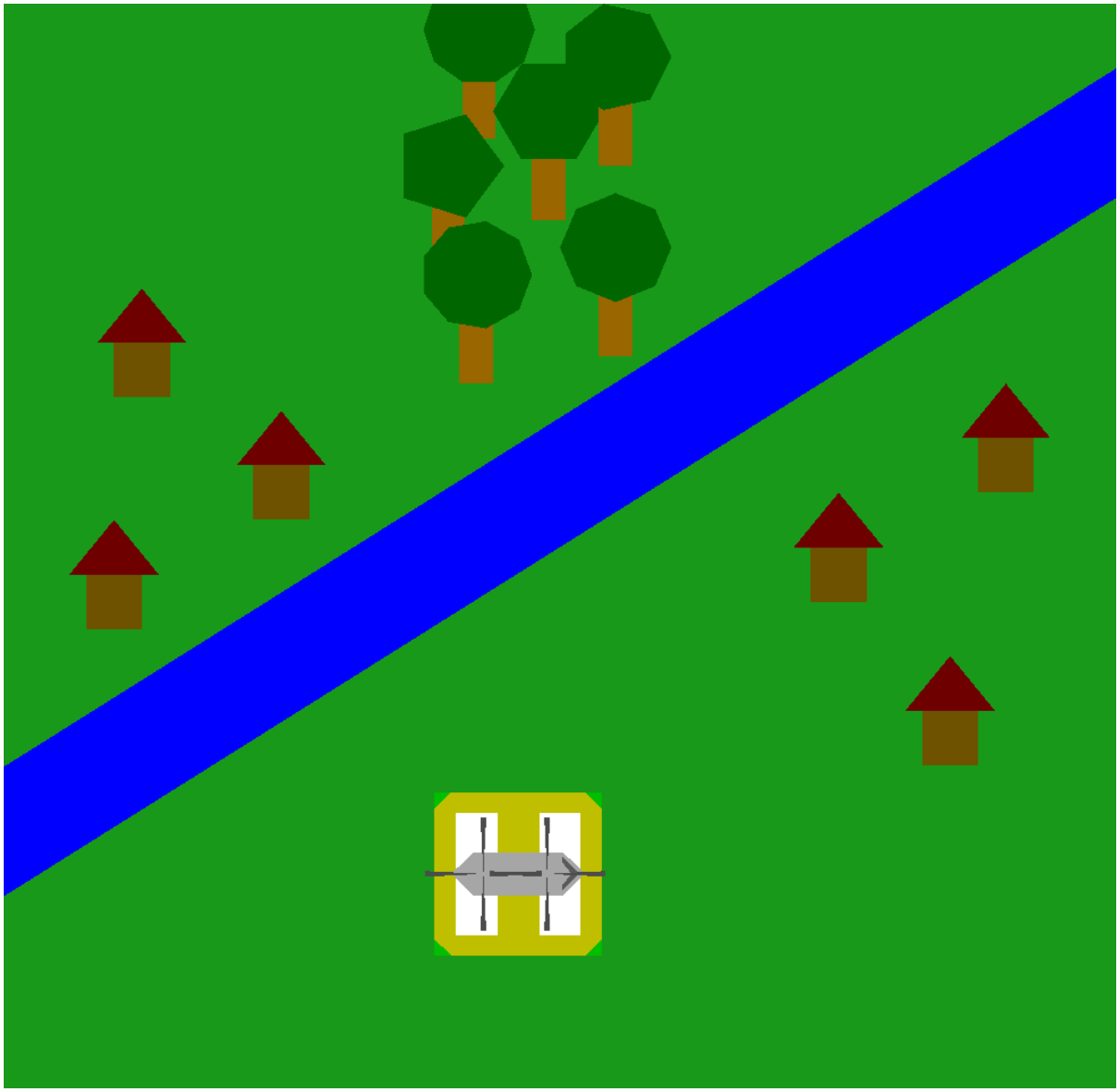
Features to be marked in this assignment. In addition to the required features, select at most three of the optional features for a total mark of 100%.

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%	
Control helicopter with the mouse	10%	Yes
Take-off and landing at the helipad	10%	Yes
Camera mounted on the helicopter	10%	
Minimap	10%	
Curved rivers	10%	
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



Implemented in:

- River.java
- House.java
- Tree.java

Instantiated and managed in Assignment1.java in the init() function
Something noteworthy about this is that as you can see, the top of each tree is a regular polygon with deferring amounts of edges. I programmed it dynamically and you can find the code for this in Tree.java method createLeaves()

Moving helicopter with keyboard

As described in spec doc and key binding section.
Note that the helicopter accelerates to a max while the key is down and then then decelerates when the key is released.

The nose of the helicopter is to signify to the user which way it's going to go when the up arrow key is pressed because otherwise they may forget which point is the front.
Key listening in handled with the KeyListener interface implemented in Assignment1.java in the methods keyPressed() and keyReleased()
Logic related to key events is handled in the update() method of Assignment1.java.
Key related data stored in KeyManager.java
Helicopter movement logic and operations are handled in in the move() parent method and specific logic/operations relating to movement as a response to key presses specifically can be found in the parent moveByButtonPress() parent method both in the Helicopter.java class.

Helicopter with spinning tandem rotors

Completely as described in the spec doc, except implemented with acceleration and deceleration or rotation.
Implemented in all added methods of Rotor.java, namely the rotate() method and initialised/instantiated in the Helicopter.java constructor.
Rotors decelerate while landing and stop spinning completely after landing.
Rotors accelerate before take-off and reach max rotation speed while helicopter is rising.
Call to rotate rotors at Assignment1.java in the update() method.

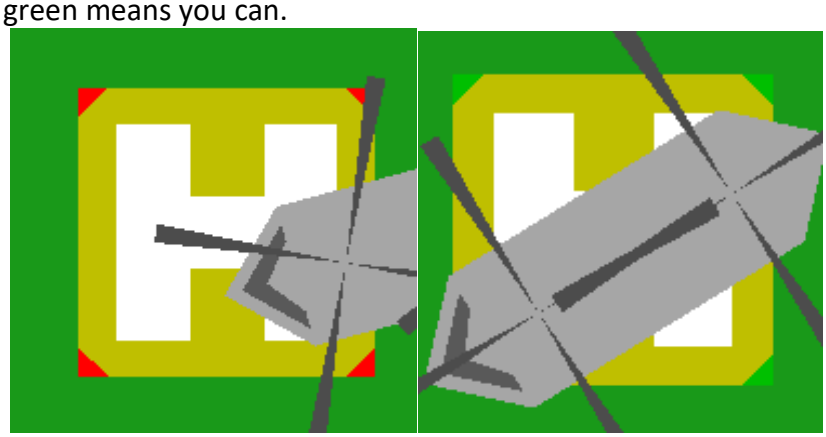
Control helicopter with the mouse

While the helicopter will reach a point where the body is covering the point of click, the final point of click will not be the very centre of the helicopter.
Due to the acceleration system, I used trial and error to detect how far away from the point the helicopter should start decelerating. This number can be found in the second line of the moveTowardsMouseCoordinates() method of the Helicopter.java class.
Clicking sufficiently close to the helicopter will cause it to rotate to face that point.
The helicopter will decide how much to turn before moving forward based on how far the helicopter is away from the click point. This number can be found in the first line of the moveTowardsMouseCoordinates() method of the Helicopter.java class.
Occasionally a click isn't registered. The best thing to do then is click again.
Also, occasionally when clicked at a particular angle (that I couldn't work out) the helicopter will do perpetual loops around the point instead of reaching it.
Mouse click listening handled in Assignment1.java by implementing the MouseListener interface.
Mouse movement logic in the parent method moveByMousePoint() in the Helicopter.java

Take-off and landing at the helipad

First thing to note that the helicopter won't start taking off immediately after the 't' button is pressed. Instead, the rotors start rotating and the helicopter will start rising when the rotos are at 50% max speed.
On the other hand, the rotos slow down immediately when landing and do so at rate so they they're still spinning slowly for about a second after the helicopter has landed.
The spec document says that the helicopter should only land when it's over the helipad and so in order to implement that, I let landing be possible when the centre of the helicopter is a certain distance away from the centre of the helipad. The distance was found through trial

and error, can be found at update() method of Assignment1.java at the top of the last conditional block. It was set to make sure that when the helicopter lands on the pad, the full body of the helicopter is over the pad and none is sticking off the side.
Note that the helicopter can be moved while taking off but not while landing in order to avoid being able to start landing when in correct distance but then moving out of the distance.
Finally, it felt necessary to me to give the user feedback as to when the program will let the user land or not. In order to make that happen I put wedges on the corners of the helipad to kind of simulate lights that change colour to indicate this. Red means you can't land and green means you can.



The part of the code that handles letting the user land the helicopter is in the same conditional block of code described above.
Code for handling taking off and landing are in the handleTakingOff() and handelLanding() methods in the Helicopter.java class.

Variations to specification

The one variation I made from the spec document was to do with sizes. It feels appropriate to me to have the helicopter larger than the suggested size. The reason for this is because I really wanted to emphasize that the helicopter was close to the camera when in flight and far away when landed. Furthermore, I also wanted to emphasize the process of the helicopter getting closer to the camera when taking off and getting further away from the camera when