# COMP330 2019 Assignment 1
# SES Helicopter
# Due: 7:00pm 20 April 2019 (mid break)
# Value: 20%

The State Emergency Service assists people affected by natural disasters. In this assignment, you will develop a 2D computer graphics simulation of the operation of an SES Helicopter in a flood setting.

Australia often experiences natural disasters, particularly flood. In this assignment scenario, a flood is rising near a town, threatening the town and its residents. The SES helicopter (in this assignment) can rescue people who are trapped by the flood waters, and carry them to safety. The simulation will allow the end user to develop their skill in deciding what the helicopter should do as the situation develops in order to most effectively protect the town and its residents.

## Task Description

This assignment is about 2D graphics in WebGL.

Write a WebGL program that provides an overhead view of an emergency services helicopter. Show the helicopter flying over the flood scene and performing actions as described below. The program is controlled by the mouse and keyboard.

Also write a short report that discusses the important features of your program, with screen shots, and provides a very brief user documentation explaining how important features work. Use the **report template** provided with this assignment on iLearn. On the **front page** of the template there is a table where you must select the features that you have implemented.

Your program must meet the following specifications.

1. Your program is to run in the WebGL environment embedded in Firefox developer edition as installed in the lab. In particular, your program will be tested on a computer running the same environment as in the lab. If you develop your program in your own environment, you should test it in the labs to ensure it runs in the standard environment.
2. The program will display the helicopter as viewed from above, along with other features (see below).
3. The helicopter may be in flight or hovering. An added feature allows the helicopter to land at an SES base.
4. Short-cut keyboard actions may support additional features as defined below.
5. The program must be properly commented, with authorship information including your name and student ID. Each procedure or function must have a brief comment describing the behaviour of that function. Data structures / classes must be similarly described.
6. The program must follow WebGL coding conventions as presented in this unit.

7. The program's display should fill the entire canvas. Optionally, the program should adapt to changes in the canvas size.
8. The program code must be your own original work. Where you use ideas from other people's programs, you must acknowledge the author in a comment and include the URL of the source if it is on the Internet. You may use code examples from the COMP330 web site; however, it is also appropriate to acknowledge them.
9. In order to receive marks for the operational features of your program, they must be obvious to the marker. This means that the marker will not be reading your source code to look for program documentation. Instead, they will check out the things that they can see on the screen, and in the first page of your report.
10. Use exactly the command keys as listed in this assignment specification, so that the marker can easily operate your program.

## Features

Note that the features total more than 100%. This is to allow you opportunity to choose which features you implement. On the front page of your report, you will select which features you want to be marked on. Note that you should not select features with a total of more than 100% - if you do, your final mark will be based on the first selection of features that total no more than 100%, not the sum of marks for every feature that you implement. In other words, do a good job of implementing a set of features that total to at most 100%, rather than trying to implement more features but doing a poor job of each of them.

The first two features are essential for a pass mark. All other features are optional.

See the supplied sample screen shots for an example of what the various features may look like. It is not intended that you copy the example screen shots, but you are encouraged to use them to aid your understanding of the key ideas.

### Static world with buildings, a (flooded) river, and a helipad. 20%

Essential feature. This is the basic background feature required of every program. Start with this.

Show a town from the air, with a flooded river nearby. For simplicity, the river can be straight and the town can consist of a collection of at least 3 simple buildings. See the sample screen shots for examples of what this could look like. Include a helipad some distance from the town.

For best marks for this section, use appropriate colours for the ground, buildings and flooded river.

### Moving the helicopter with keyboard control: turning, moving forward. 20%.

Essential feature. This feature provides a basic helicopter. The helicopter can be a simple polygon that roughly resembles a helicopter viewed from above. The helicopter should appear to be about 50-150 pixels long – detail needs to be visible but the helicopter should be able to move on the screen a significant distance relative to its length. The helicopter does not need to be in scale relative to the features on the ground.

Keyboard commands are used to control the helicopter – see the keyboard command table below. The speed of rotation of the helicopter, and the speed of movement, should be chosen so that it is

possible to move the helicopter across the screen in about 4-8 seconds, and it should be possible to rotate the helicopter by 180 degrees in about 2-4 seconds.

## Helicopter with spinning tandem rotors. 20%

Enhance the helicopter by showing spinning rotors on top of it. Google "helicopter with tandem rotors" or see https://en.wikipedia.org/wiki/Tandem_rotors. You will find that such helicopters have one rotor at the front and another at the rear. The rear rotor spins in the opposite direction to the front rotor.

Whenever your helicopter is in flight, or hovering, the rotors should spin relative to the body of the helicopter. Choose a rotation speed that allows the user to clearly see the rotation happening - say ½ to 1 rotation per second.

## Rescuing people. 5%

People sometimes need to be rescued. At random times, a person may appear near one of the buildings desperately needing to be rescued. This is more likely to happen if a building is close to the floodwaters. Since rescuing people is the main simulation activity, ensure that people needing to be rescued appear at a suitable rate to make the simulation interesting, and perhaps challenging at times, but not impossible.

Rescue consists of picking people up and taking them to the SES base. The base is the helipad together with a building next to it.

The helicopter can hover over the people and pick them up, carry them back to the base and safety. When the helicopter is over the person, press the command key listed below to pick up the person. Fly to the base and land there, using the command key listed below, to take them to safety. After landing, the people leave the helicopter.

To make it simple to determine whether the helicopter is over a person, you can just compare the centre point of the helicopter with the centre point of the person. If the distance between them is less than about 1.5 times the width of the helicopter, then the helicopter can pick up the person. Use a similar rule to decide whether the helicopter is over the helipad so that it can be commanded to land. You can adjust the radius to make the game play work better.

When the helicopter picks up a person, show the person on the helicopter until the helicopter lands. After landing, they leave the helicopter and enter the SES base building, so they are no longer visible on the screen. You do not need to animate people moving from the helicopter to the building, although you can if you wish.

To instruct the helicopter to take off again from the base, press the command key listed below or instruct it to fly to a destination location.

## Resizing the canvas, maintaining aspect ratio. 5%

If the user resizes the drawing canvas, your program will ensure that the resulting display maintains the correct world aspect ratio. This means that you may need to display more or less of the world. Ensure that your solution maintains a sensible display even if the canvas is resized many times.

## Control helicopter with the mouse. 10%

Your program responds to a mouse click on a screen location by sending the helicopter to that location. The helicopter must turn towards the target position, and fly towards it. When it arrives, it must hover at the location, awaiting further commands.

Animate the helicopter to respond to the mouse click by turning towards the destination position and then flying towards it. To provide the user with reasonable control, it should take somewhere between 4 and 8 seconds for the helicopter to move from one side of the screen to the other.

Simple animation of the helicopter will appear that the helicopter can turn on the spot when it is stationary but the helicopter should not spin too quickly– it should take the helicopter somewhere between 2 and 4 seconds to complete a 180 degree rotation.

To instruct the helicopter to move, the operator clicks on the destination position for the helicopter. The helicopter then starts to execute the instruction, by turning to the direction of travel and then flying forward at a steady pace until it reaches the marked point where it hovers.

For added realism, the helicopter may be observed to accelerate and decelerate rather than instantaneously changing its speed of motion. It may also be observed to turn and fly forward at the same time, traveling on an initially curved path towards the destination.

## Camera mounted on the helicopter. 10%

The program normally shows the view from above the entire scene. Provide a short-cut command as listed below to switch to show the view from the helicopter. The same command can switch back to show the entire scene – it switches between the two modes.

The helicopter itself will not appear in the view from the helicopter. To make the view from the helicopter more interesting, show only the area immediately surrounding the helicopter, with all features on the ground enlarged by a factor of between 2 and 3, as though the camera on the helicopter had a telephoto lens. Also, when the helicopter lands, zoom in on the base and zoom out again when it takes off.

Note that, when the helicopter turns, the ground scene will appear to rotate in the opposite direction, since the camera turns with the helicopter. Similarly, when the helicopter flies forward, the ground will appear to move in the opposite direction because the camera is moving with the helicopter.

You should assume that the camera on board the helicopter is aligned so that “up” in the image corresponds to the longitudinal axis of the helicopter – i.e. to the direction “forward” on the helicopter.

## Minimap. 10%

Use a viewport to draw a minimap of the entire world and the helicopter location in a corner of the screen. Use a separate (larger) viewport for the main camera view.

## Curved rivers. 10%

Draw the river and a branch, or two rivers, as extruded curved 2D shapes.

### Rain. 10%

Use a particle system of points to draw rain falling across the scene. Raindrop locations should be randomised and the points should get smaller as they drop to the ground and then disappear.

### Heads up display. 10%.

Provide a status display overlaid on the camera view that shows the ground speed of the helicopter. When the helicopter is hovering or landed, it should show 0 speed, but when it is flying forward it should show the speed of motion. Draw a standard "needle" type of speedo, and you don't have to show any numbers, just the speedo arcs, speed tick marks and the needle.

## Keyboard Commands

To avoid any confusion, here is a table of all keyboard commands that you may use in this assignment. You must use the exact keyboard commands listed here, to simplify marking.

Where more than one key is listed, your program should respond to every alternative. For example, to pick up a person, the user may press lower case 'p' or upper case 'P' and both should work.

| Keyboard command | Action |
|---|---|
| Left arrow | Helicopter rotates anticlockwise in the world as viewed from above. Hold for continuous rotation. |
| Right arrrow | Helicopter rotates clockwise. Hold for continuous rotation. |
| Up arrow | Helicopter moves forward a fixed distance. Hold to fly forward at a fixed speed. |
| 'p' or 'P' | Pick up a person |
| 'l' or 'L' | Land the helicopter at the base |
| 't' or 'T' | Take off from the base. The helicopter rises and hovers. |
| 'c' or 'C' | Switch camera views – normal or helicopter mounted camera |

## Marking

The features that you identify in your report will be marked against the requirements listed above. Some marks will reflect creativity and user experience, but the majority of marks will be for implementation showing that you have understood and applied the relevant WebGL features.

If your feature set exceeds 100%, the marker will not mark excess features. Select the features on the front page of your report that you want to be marked. Do not select features totalling more than 100%, as the marker will ignore excess features.

## Submission

Submit your program as a WebGL project that can be run in the lab's Firefox developer edition browser environment. Be sure that your program does not crash in testing – markers cannot correct programming bugs that you may introduce at the last minute so be sure to test your final submission thoroughly.

- Markers cannot test your program in any other environment. You must ensure that it works in the lab environment.
- You are encouraged to use a source code revision management system (such as Git) to manage your project.
- Make small changes and test them thoroughly before moving on to other features.
- Test the interactions between features.
- Make it clear to the marker what your program can do, and how to use it. In particular, the summary table in your report will be used by the marker to explore your program's capabilities.

Details of the submission method will be provided as an update to this assignment specification – watch this space.

## Marking Rubric

Total mark = 40% for components implemented, 30% for code correctness, 20% for clarity of code and 10% for creativity.

Each component that you implement will be marked for completeness of the implementation against the specification. The overall program will be marked for correctness, clarity and creativity according to the following rubric.

| Grade | Correctness | Clarity | Creativity |
|---|---|---|---|
| HD (100) | Code is free from any apparent errors. | Good consistent style. Well structured & commented code. | Original use of graphics to achieve effects not included in lectures or pracs. |
| HD (90) | Code has minor errors which do not significantly affect performance | Minor inconsistencies in style, or patches of undocumented code. | |
| D(80) | Code has one or two minor errors that affect performance | Code is mostly readable but could be better structured | Creative use of graphics effects from lectures/pracs to add distinctiveness to submission |
| Cr(70) | Code has multiple significant errors | Code is inconsistently structured, but readable | |
| P(60) | Code is functional but contains major flaws | Code is poorly structured and patchily documented | Basic effort put into colour palette and shapes to give the simulations a consistent style |
| F (< 50) | Code is not functional | Code is undocumented and largely unreadable | Implements required features without effort put into style |