COMP330 2017 Assignment 1
Ocean Fighter
Due: 11:45pm 15 April 2016 (week 7)
Value: 20%

Seeshoot, an Indie film company, is preparing for a movie that features futuristic sea fight scenes. To reduce production costs, computer graphics will be used to portray the vessels and the sea battles. In order to plan the scenes, they first require a graphics program that can show the behaviour of a vessel from overhead. Your task, as their OpenGL expert programmer, is to develop this graphics program. Because this software will be used for concept development, you only need 2D graphics to produce suitable animations.

Task Description

Write an OpenGL program that provides an overhead view of a futuristic sea fighting boat called "Ocean Fighter". Show the boat roaming the ocean and performing actions as described below. The program is controlled by the mouse and keyboard. Your programs must meet the following specifications.

1. Your program is to be compiled and run under the Eclipse OpenGL environment provided in the labs and on iLearn. In particular, your program will be tested on an x64 machine with Eclipse 4.4.2 and the environment specified on iLearn.
2. The program will display the Ocean Fighter as viewed from above, along with other features (see below).
3. The mouse position on the screen specifies the point that the Ocean Fighter should move towards. If the Ocean Fighter reaches that point, or is as close as it could get, then the Ocean Fighter will stop. This means that you can stop the Form Fighter by positioning the mouse over it, and start it again by moving the mouse away. This happens even when the mouse button is not pressed.
4. Clicking the left mouse button fires a torpedo straight ahead from the Ocean Fighter. The torpedoes will be visible on the screen as they move away from the Ocean Fighter at high speed. Holding the left button down causes additional torpedoes to be fired behind the first – you should see more than one torpedo on the screen at the same time.
5. Clicking the right mouse button pauses the simulation and provides a menu of additional features. Minimally, there is a menu item to exit the program. In normal circumstances (other than exiting the program), animation will resume as soon as the menu request has been processed.
6. Short-cut keyboard actions may support additional features. The short-cut keyboard actions should also appear in the menu so that a user can right click to find out what keyboard shortcuts are supported by your program.

7. 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. See the Code Style guide that is available in the resources section.
8. The program must follow C++/ C coding conventions as specified in the Code Style guide.
9. 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; again, it is appropriate to acknowledge them.
10. 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 your program menu.

Grades and Features

Each grade level corresponds to a list of features and expected code quality parameters.

Pass

A pass-level program will satisfy have the following features and capabilities.

1. Program code will satisfy commenting requirements with perhaps some minor lapses. In particular, most procedures will be described by comments.
2. Program code will satisfy C or C++ coding conventions with perhaps some minor lapses.
3. Program displays Ocean Fighter from above. Ocean Fighter may be represented as a polygon that should give the appearance of the shape of a boat. Higher marks will be awarded for more creative representations of the Ocean Fighter.
4. Program animates Ocean Fighter to turn towards the mouse and drive towards it. To provide the user with reasonable control, it should take somewhere between 3 and 6 seconds for the Ocean Fighter to move from one side of the screen to the other.
5. The animation of the Ocean Fighter may appear as though the Ocean Fighter can turn on the spot when it is stationary but the Ocean Fighter should not spin too quickly if the mouse is moved rapidly across the screen – it should take the Ocean Fighter somewhere between 2 and 5 seconds to complete a 360 degree rotation.
6. The screen window must be approximately 1000 pixels wide and 750 pixels high.
7. The Ocean Fighter should appear to be about 40-80 pixels long – detail needs to be visible but the Ocean Fighter should be able to move on the screen a significant distance relative to its length.
8. The animation rate must be a fixed rate of at least 60 frames per second.
9. The Ocean Fighter is initially located somewhere near the centre of the screen window.
10. The ocean is blue.

Additional Features

The following additional features may earn higher grades. Larger programs should be broken into separate source code files to enhance modularity.

Large features

One large feature properly implemented is sufficient to move from pass to credit grade.

1. There are a few fuel drums floating on the ocean. The Ocean Fighter can torpedo the drums and they will explode. The flames burn on the water surface for about two seconds. If the Ocean Fighter runs over the drums, the drums catch fire but the Ocean Fighter is not damaged. For best marks, have the fuel drums slowly drift around randomly in the ocean as a result of the action of waves.
2. There is an island with a gun placement that fires on the Ocean Fighter when it is within range. The Ocean Fighter is able to withstand the attack of the gun and displays no damage, although hits on the Ocean Fighter appear as momentary explosions. Missiles fired by the gun should be visible on the screen as they travel towards the Ocean Fighter. The gun should be near a corner or side of the screen and the range should be something like 1/3 of the screen width so that it is not difficult to move the Ocean Fighter into range of the gun, but also there is plenty of screen space where the gun can fire on the Ocean Fighter. Use an appropriate colour for the island. A torpedo cannot travel through the island.
3. Mines. The ocean fighter can drop mines from the rear of the boat. Mines float on the water and explode if they are touched by the ocean fighter, or by a fuel drum. If a fuel drum hits a mine, the drum explodes and catches fire. Mines are visible in the image. Mines drift slowly around the screen due to effect of ocean currents. If the ocean fighter hits one of its own mines, it is damaged and sinks after a few seconds. Creativity can be shown in the rendering of damage and sinking of the ocean fighter. The program should provide an option to resume the simulation with a 'new' ocean fighter after the ocean fighter has sunk.

Small features

Each small feature is worth half of a large feature. Two small features properly implemented are sufficient to move from pass to credit grade.

4. A fuel fire on the ocean surface produces a cloud of dense black smoke that slowly drifts across the screen in the sea breeze, until it disappears off the edge of the window. The smoke should take 5-10 seconds to cross the screen. The smoke makes it impossible to see anything under it – island, ocean, Ocean Fighter, etc are all hidden from view when the smoke passes over them.
5. The Ocean Fighter leaves a white wake behind it as it travels across the ocean. The wake disappears after 2-3 seconds. For extra realism, the wake fades gradually over 2-3 seconds rather than suddenly disappearing.
6. Radar dish and rudder. The ocean fighter has a radar dish on top of the boat that rotates (relative to the boat) about once every 3-5 seconds. A rudder on the rear of the boat shows how the boat is being turned. Make the rudder position appear realistic for the motion of the ocean fighter.

Credit

For a credit grade, the program should

1. Meet all the requirements for pass.

2. The Ocean Fighter turns as most boats normally do – it turns as it moves forward. The Ocean Fighter cannot turn in a circle that is any smaller in diameter than the length of the Ocean Fighter.
3. Implement at least one of the large additional features or two of the small additional features described above.
4. Have appropriate modularisation, including the use of separate files for different source code modules as appropriate.
5. Display the Ocean Fighter with more detail than just a polygon.
6. Use colour to enhance the realism of the images.
7. Follow the style guidelines consistently.

Distinction/High Distinction

For a distinction or higher grade, the program should

1. Meet the requirements for a credit.
2. If the user enlarges the screen window, the image should show more of the ocean and the Ocean Fighter should be able to move over the larger area. You should also consider what to do if the window is made smaller and implement an appropriate solution.
3. Implement at least: two of the large additional features described above, or one large additional feature and two small additional features.
4. Use colour and detail to enhance the realism of the images.
5. Have excellent code style.

Optional Features

Creative optional features may be rewarded with additional marks within a grade level. Here are some ideas of things you may attempt.

1. Capability to vary Ocean Fighter speed.
2. Option to launch torpedoes from a rear-facing torpedo tube.
3. Visually interesting design of the Ocean Fighter, its components, and/or the ocean scene.

Marking Breakdown

- Code style. 20%
- Program features, correctness. 70%
- Creativity. 10%

Submission

Submit your program as an Eclipse project that can be imported into the lab's Eclipse environment, then compiled and run. Be sure that your program compiles without errors – markers cannot correct programming bugs that you may introduce at the last minute so be sure to test your zip file by importing it into Eclipse, compiling it from scratch and running it. You must ensure that your program will compile and run in the lab environment, because markers cannot test your program in any other environment.