Department of Computing, Faculty of Science
Macquarie University

COMP115 Introduction to Computer Science

Session 1, 2014

Assignment Three: The Great Barrier Fish

Due: 11:00am, Monday 9th June, 2014 (Monday Week 13)

Worth: 17%

Learning Goals

This assignment contributes to the learning outcomes of COMP115 as follows:

• *Apply problem solving skills to develop algorithms that solve small to medium-sized computational problems*: Introductory experience in developing an algorithm to solve a problem from an informal specification of the problem.

• *Design and code implementations of their algorithms in an imperative programming language*: Practical experience in translating a simple algorithm into an equivalent Processing program that uses drawing, animation and variables.

• *Use standard software engineering practices to document, debug and test their programs*: Experience in developing a clear program from a specification, testing the conformance of the program to the specification, and debugging any problems that are detected.

Introduction

This assignment asks you to write a program that extends Swimmy Fish version 2 implementation that was the topic of Assignment Two. You will add more complex behaviour using arrays and loops, as well as extend the drawing performed by the program. You should only need to use material up to and including Chapter 9 of the textbook, and week 10 of the lecture.

There are videos on iLearn to show you generally how your assignment should look. As in Assignment Two, there are some effects that are just to show you what is happening and **should not be** part of your code: the circles or viewfinder that appear when the mouse button is clicked, and the text captions that explain what is happening.

We strongly suggest that you divide the assignment up into pieces, tackle each piece separately and don't move on until you get each one working.

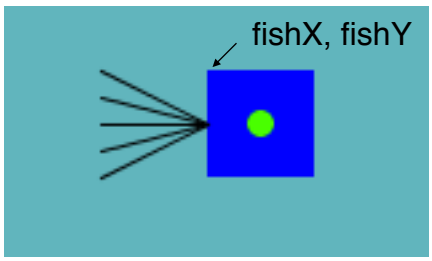
Questions

If you have questions about the programming for this assignment, ask on the Assignment Three Forum on iLearn, consult with your tutor or ask Gaurav Gupta (gaurav.gupta@mq.edu.au). No extensions will be granted except for serious and unavoidable disruption. Submit a disruptions request as soon as possible, preferably before the due date, if you have suffered a disruption that affects your ability to submit this assignment.

You have been provided a skeleton program in `ass3.zip`. It is **recommended** that you use it as a starting point for your assignment. If you don't like my template, you can ignore it and start from scratch. You can move your code from assignment two if you think it will help.

Part A: Fish (10%)

Assignment two drew a swimming fish that is supposed to escape through a barrier at the right of the display window. A partial code for drawing the fish body has been supplied in **`drawFish`** function. **Update it** to create a fish with a spiky tail. The body of the fish should be blue with a green eye in the middle. The width and height of the fish are **`fishWidth`** and **`fishHeight`** respectively (provided in **`ass3.zip`**). Each time the fish moves forward, it should do so by **`3`** pixels (no dramas if you make it 2 or 4, as long as it's "playable") and upon pressing an UP or DOWN key, it should move up or down respectively, by **`fishHeight`** pixels. In the beginning, the fish appears such that its right-most point is at `x = 0`, and its top-most point (**`fishY`**) is at a multiple of **`fishHeight`** and in the range **`[0, height[`** (including 0, and excluding **`height`**)



The tail should have 5 spikes, **drawn using a loop (no marks if a purposeful loop is not used)**, each having one end at the middle left of the blue rectangle that represents the fish, and the other ends of the 5 spikes should be evenly spread from the top of the fish body to the bottom of the fish body. The width of the tail should be the same as the width of the fish body. Apart from these specifications, the tail should resemble the example on the left as closely as possible.

continued on next page …

Part B: Barriers (35%)

As your main task, you are required to create multiple barriers, using a loop, in the path (no marks if a purposeful loop is not used). These barriers must satisfy following criteria -

I. There are **`N_BARRIERS`** barriers (declaration of **`N_BARRIERS`** provided in **`ass3.zip`**)

II. Size of each barrier should be same: **`barrierWidth ( = fishWidth)`**, **`barrierHeight ( = fishHeight)`** (provided in **`ass3.zip`**)

III. All barriers are displayed as red borderless rectangles.

IV. The basic attributes of the barriers **`(x position, y position)`** should be stored in arrays (no marks if arrays not used to draw the barriers), you can add any additional attributes you need.

V. The top-left corner of barrier corresponding to index **`i`** is at **`(x[i], y[i])`** such that **`x[i]`** is a multiple of **`barrierWidth`** and **`y[i]`** is a multiple of **`barrierHeight`** such that all barriers are completely within display window.

Part C: Reset barriers and fish each time fish crosses the screen (15%)

The entire barrier set is replaced by a new barrier set each time the entire fish (including the spiky tail) crosses the screen (fish completely disappears to the right of the display window), and also the fish restarts from the left side of the window. You are free to choose the initial y coordinate of the fish as long as its a multiple of fishWidth.

Part D: Collision and scoring (14%)

When the fish collides with a barrier, it should start again from the left and the score should be decreased by 3. Each time the fish crosses the display window without colliding with a barrier, the score should be increased by 3. The score should be displayed towards the top left corner of the display window.

Part E: Hang on! Some barriers are actually food (10%)

Roughly thirty percent of the barriers should be “food” for the fish, and displayed in green. When a fish collides with a green barrier, the score should be increased by 1, and the green barrier hit should disappear.

continued on next page …

Part F: Open ended (16%)

You can add a ***non-trivial*** functionality to the program in this part. Three examples of substantial functionality have been implemented and demonstrated in the assignment video, namely,

1. Pausing and un-pausing (with a timing component)

2. Having to press 's' to start the game, the game going on for a set time period, after which an "end game" screen is displayed (with a timing component).

3. Making sure none of the 15 barriers overlap with another barrier. Thus, all 15 barriers are visible in the beginning (and the food barriers might get eaten up during the round)

NOTE: If you have doubts about whether the functionality you intend to include is *non-trivial enough*, please contact your tutor **at least one week before** the deadline.

Template provided

We have provided you with a template (**`ass3.zip`** containing a sketch **`ass3`**, that contains one file - **`ass3.pde`**, that has variables for certain important values declared globally).

Submission

Before the due date and time, you must submit your Processing program online via the COMP115 iLearn site. You must submit your program as a single Processing source file called **`ass3.pde`** (if all functions in a single tab), or **`ass3.zip`** containing the zipped sketch folder (if functions in different tab).

Marking

80% for the correctness of your code. We will check to see that your program correctly implements the specifications given above.

20% for the quality of your code. We will check to make sure that

I. you have used reasonable names and types for your variables
II. your code is presented in a manner that makes it understandable (e.g., good formatting, using comments to explain what your code is doing).
III. your code is written in a modular manner (proper use of functions)

Use the sample programs shown in lectures or the textbook as guidance.