CPSC 314: Programming Assignment 1

Due: 4:00 PM, Friday September 20, 2013

The goal of this assignment is to introduce you to OpenGL. You will experiment with drawing and callback commands, and event-driven frameworks in general.

In the assignment you will implement a simple 2D game where the user controls a salmon swimming upstream to spawn. Can your salmon dodge the turtles that rush by? You will implement this game by building on top of a bare-bones instructor-provided template, adding your code in appropriate callback routines, etc.

The assignment includes both a required (80%) and a free-form component (20%). The goal of the latter is to let you experiment with computer graphics and have fun.

1. Getting started (15%):

   (a) Download and untar the source template, a1.tar.gz. It includes a makefile, source template files (a1.cpp, salmon.hpp/cpp, turtle.hpp/cpp) and two executables: 'a1_sol' for Linux, and 'a1_sol.exe' for Win32. Download the tar package here:

   `www.ugrad.cs.ubc.ca/~cs314/Vsep2013/a1/a1.tar.gz`

   (b) The department Linux machines should have all the libraries you need for the assignment. If you work at home (Linux or Windows) you need to install the glut libraries and headers on your machine. For installation help, consult:

   `http://www.opengl.org/resources/libraries/glut/glut_downloads.php`

   or

   `http://web.eecs.umich.edu/~sugih/courses/eecs487/glut-howto/#win`

   (c) Run 'a1' (a1_win.exe in Windows) to get a sense of what a possible assignment solution should look like. Use the mouse and arrow keys to control the salmon. Press 'r' to reset the game if your salmon collides with a turtle. Use the 'b' and 'a' keys to switch between basic and advanced mode. You can also change the speed of the river current with '<' and '>'.

   (d) Build the template executable. In Linux use the provided makefile. In Microsoft Visual Studio 2012/2010 use the provided project files. Depending on your home environment you might need to do other changes to make the code run.

2. For a basic version of the game make the following changes to the provided template (65%):

   (a) Movement: pressing the up/down arrow keys should make the salmon swim up and down and pressing the left/right arrows should make it swim left or right. Use the 'keyboardSpecial()' and 'keyboardSpecialUp()' callback functions to keep track of the state of the arrow keys. Then, modify the 'animate()' callback function to send movement commands to the salmon depending on the state of the arrow keys. Use 'Salmon::move()' to tell the salmon how to move. Finally, you will need to modify 'Salmon::draw()' in order to draw the salmon in the correct position. Use the GLTranslate command to change the salmon position during drawing.

   (b) Rotation: Provide mouse control for rotating the salmon, so that moving the mouse to the left/right rotates the salmon clockwise/counterclockwise. In the mouse_move() callback function, use previous and current mouse 'x' coordinates to decide on the rotation direction and magnitude (ignore the 'y' coordinate). Set the amount of rotation to be proportional to the mouse displacement magnitude (choose a reasonable correlation between the two). Use 'Salmon::rotate()' to tell the salmon to rotate. Again, modify Salmon::draw() to use GLRotate to perform the actual rotation during drawing.

   (c) Collision: Have the salmon and turtle react when there is a collision:

      i. Salmon: There is a colour variable associated with the salmon. Your task is to set this variable to a randomized valid colour value when the salmon collides with a turtle. This should be done within the 'Salmon::setRandomColour()' function. You will also need to modify the salmon drawing function to use this colour for the salmon body. Hint: look at how colours are set right now for the salmon and other elements (RGBA).

      ii. Turtle: Have the turtle light up when it collides with the salmon. When contact occurs, generate one or more lights inside the turtle. To set the light, modify the 'Turtle::setupLight()' function. Hint: Look at the 'setup_lighting()' function in 'a1.cpp' and use a similar setup. Note that you need to use $GL\_LIGHT0 + lightNo$ as the identifier for the light.

3. The required code changes described so far will let you earn up to 80% of the grade. To earn the remaining 20% as well as possible bonus marks you need to make the game more appealing (the size of the bonus will be at the marker's discretion). You could add features found in the advanced mode of the demo, such as:

   (a) make the salmon "bounce" off the top and bottom walls when it runs into them

   (b) change the movement of the salmon to be consistent with its orientation, so that the up/down keys move the salmon along the direction it is aligned with

   (c) give the salmon momentum so that it continues moving even when no arrow keys are pressed

Other possible changes not found in the demo include (but are not limited to!):

(a) improve the collision mechanism, e.g., can you tighten the salmon or turtle collision boundaries

(b) add additional visuals, either on the salmon or turtle (these can be animated or static)

(c) randomize the turtles' paths

(d) generate an interesting riverbed

(e) diversify the types of obstacles floating down the river, or add food that the salmon can eat

(f) did someone say bubbles?

Use your imagination to make any other changes, however please make sure you focus on tasks involving OpenGL knowledge.

To support both basic and advanced visualization and control features, you need to add a toggle option where the user switches between the two modes by pushing the 'a' and 'b' keys ('a' for advanced mode and 'b' for basic mode).

Document all the features you add in the README file you submit with the assignment. Advice: implement and test all the required tasks first before starting the free-form part.

Hand-in Instructions

1. Create a root directory for our course in your account, called cs314. Later all the assignment handin files should be put in this directory.

2. For assignment 1, create a folder called assn1 under cs314 and put all the source files that you want to handin in it, including the "makefile". Don't use subdirectories – these will be deleted. **NOTE: we only accept README, makefile and files ending in cpp, hpp, c, h, txt.**

3. The assignment should be handed in with the exact command:

```
handin cs314 assn1
```

This will handin your entire assn1 directory tree by making a copy of your assn1 directory, and deleting all subdirectories! (If you want to know more about this handin command, use: man handin.)