CPSC 314: Programming Assignment 1

Due 4pm, Friday September, 21 2012

The goal of this assignment is to introduce you to OpenGL and event-driven frameworks in general, and to provide you an opportunity to experiment with drawing and callback commands. In the assignment you will implement a simple 2D game where the user controls a rocket ship and tries to land it, base down, on a moving landing pad. You will implement this game building on top of an instructor provided template, adding your code into appropriate callback routines. The assignment includes both a required (80%) and a free-form component (20%). The goal of latter is is to let you experiment with computer graphics and have fun.

- Getting started (15%):
  - Download and untar a1.tar.gz (from http://www.ugrad.cs.ubc.ca/~cs314/Vsep2012/a1/a1.tar.gz) It includes a source template (files a1.cpp, rocket.hpp/cpp, pad.hpp/cpp) and two executables, a1_sol for Linux and a1_sol.exe for win32.
  - 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 glut libraries and headers on your machine. Check

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

    for installation instructions.
  - Run 'a1_sol' to get a sense of what a possible assignment solution should look like. Use the mouse and arrow keys to control the rocket. Press 'r' to reset the game if your rocket falls below the bottom of the screen and disappears. Use the 'b' and 'a' keys to switch between basic and advanced mode. You can also change the size of the landing pad with '<' and '>'.
  - Build the template executable. In Unix use the provided makefile. In Microsoft VS use the provided project files. Depending on your home environment you might need to do other changes to make the code run.
- For a basic version of the game make the following changes to the provided template (65%):

  - Have pressing the up/down arrow keys move the rocket up and down and pressing the left/right arrows move it 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 rocket depending on the state of the arrow keys. Use 'Rocket::move()' to tell the rocket how to move. Finally, you will need to modify 'Rocket::draw()' in order for the rocket to be drawn at the proper position. Use the GLTranslate command to change the rocket position during drawing.
  - Provide mouse control for rotating the rocket, such that moving the mouse to the left/right rotates the rocket 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 'Rocket::rotate()' to tell the rocket to rotate. Again, modify Rocket::draw() to use GLRotate to perform the actual rotation during drawing.
  - Have the pad and rocket react when the rocket lands.
    - Rocket: There is a colour variable associated with the rocket, your task is to set this variable to a randomized valid colour value when the rocket lands. This should be done within the 'Rocket::setRandomColour()' function. You will also need to modify the rocket drawing function to use this colour for the rocket body. Hint: look at how colours are set right now for parts of the rocket and other elements (RGBA).
    - Pad: Have the pad light up when the rocket lands. When contact occurs, generate one or more lights inside the pad. To set the light modify the 'Pad::setupLight()' function. Hint: Look at '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. Have the lights go down when the rocket departs.

- 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:

  - give the rocket momentum so that it continues moving even when no arrow keys are pressed.
  - make the rocket "bounce" off the walls when it runs into them.
  - change the movement of the rocket to be consistent with its orientation, so that the up/down keys move the rocket along the direction it is aligned with.
  - differentiate between "proper" rocket landings, where the rocket lands with its back to the pad, and a "bad" one where it bumps into it in any other way, and it is destroyed.

- add gravity (you can enable gravity in the demo by pressing 'g').

Other possible changes not found in the demo include:

- flying objects that can collide/interact with the rocket.
- a visual indication of the rocket being destroyed, such as the rocket falling apart.
- additional visuals, either on the landing pad or rocket. These could be animated or static.

Use your imagination to do 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

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