CS4815 Week03 Lab Exercise

Lab Objective: We will continue our brief sampling of `OpenGL` again this week by looking at and modifying a program that relies on the mouse to draw squares on the screen. Here's a quick summary of the tasks:

❶ Copy this week's lab from the class directory `~cs4815/labs/week03`

❷ Examine / browse / ponder the `square.c` file there using a file browser or `emacs` or whatever you're having yourself

❸ Compile and run the program and play around with the number of iterations that it runs for

❹ Modify the program's behaviour in responding to mouse events

❺ Hand in your completed work (by the standard deadline) using `handin`

In Detail

❶ With the semester's hierarchy set up from last time we will firstly create this week's directory. This can be done from a terminal window with

```
mkdir ~/cs4815/labs/week03
```

Alternatively, you can do this, and other file copying / creation commands with the file manager.

Now copy to your `~/cs4815/labs/week03` directory the program we will be examining today. This is called `square.c` and it can be found in `~cs4815/labs/week03` sub-directory. Change your working directory into this week's lab directory with

```
cd ~/cs4815/labs/week03
```

The command to perform the copy is

```
cp ~cs4815/labs/week03/square.c .
```

Note the dot at the end of the command that signifies copying it into the current working directory and keeping the name the same.

To use a file manager-type program to do the copy you could do the following, for example:

1. Click on 'Places' on the Panel at the top-left corner of the screen and select 'Computer' from that menu or, click on the 'computer' icon in the top-left corner;

2. double-click on the following icons to get to `~cs4815`, the `cs4815` home directory: Filesystem, users, ug2005, cs4815

❷ This week's program is a little more sophisticated than last week's, but not by much. This time, squares of a random colour get drawn whenever the mouse is dragged within the window. This happens by the repeated response of our program to "mouse motion" events. It makes an attempt at dealing with allowing the user to resize or move the window, but it's not really useful because the current drawing gets deleted.

The program operates by registering callbacks for the events of moving the mouse within the window created by the program. This is mainly achieved with the statements:

`glutMouseFunc(mouse);` (Click here for documentation)
`glutMotionFunc(drawSquare);`

The distinction between the two calls is that when the mouse button is pressed or released a mouse event occurs that will be handled by the function registered in `glutMouseFunc(mouse);` on the other hand, if you move the mouse while *pressed* this needs to be handled by `glutMotionFunc(drawSquare);`.

When mouse motion with any button pressed is detected the registered callback function, `drawSquare()`, is called. The RGB levels are set randomly using the C library `rand()` call. Then a primitive of type `GL_POLYGON` is defined and this is drawn.

❸ Play around with the program by compiling and running it.

The command for compiling the program is

```
gcc square.c -l GL -l GLU -l glut -o square
```

The program should compile without errors. After that you can run it with the command

```
./square
```

Things to notice / observe:

- When tracing out a path (and the squares generated by the program) note the difference between dragging the mouse slowly and very quickly. In the latter case the machine isn't able to respond quickly to the speed of your hand.
- Another deficiency of the program is how it responds to being exposed. To see this, partially cover over the window with another window and then move the mouse back over the "squares" window. Notice how the entire window isn't redrawn.
- Also notice its response to right-clicks of the mouse. Is it on mouse down (press) or mouse up (release) that the program terminates? Find where in the code the program exits and verify that it is consistent with what you observed when you ran the program.

❹

You should now modify the program so that it

1. responds to an additional event: just like in the lecture you should allow the user to quit from the program using 'q' or 'Q';

2. quits from the program when the right mouse button is *released*; the reason for this will become clear in the next part;

3. distinguishes between the mouse button pressed: depending on which of the three mouse buttons are held down, different sizes of squares should be drawn; in this way we can distinguish between a path dragged out by the left-, the middle- or the right-button of the mouse.

❺

You should now hand in your completed program for marking and evaluation. The command for handing in this week's – Week03 – assignment, is:

```
handin -m cs4815 -p w03
```