CS4815 Week04 Lab Exercise

Lab Objective: We will continue our introduction to `OpenGL` again this week by

- observing how selection of an object on the screen can be achieved
- observing the deficiencies of single buffering when drawing moving objects
- by creating and using a menu to modify program behaviour
- though not immediately related to graphics programming we will also see in action the `make` utility that is a life-saver when developing any type of software

Here's a quick summary of the tasks:

❶ Copy the source code for this week's lab from the class directory `~cs4815/labs/week04`

❷ Learn how to do a beginner's `make` that will compile programs easily for you

❸ Examine the source code and behaviour of the `glut7` and `glut8` programs; do likewise with the program `spin`

❹ Modify the behaviour of `spin` so that it responds to *menu* events

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

In Detail

❶ Following the procedure of previous weeks create a directory for this week's lab in `~/cs4815/labs/week04`. Now copy over from the class directory the three source files `glut7.cc`, `glut8.cc` and `spin.cc`, and the makefile `Makefile`. These are to be found in `~cs4815/labs/week04`.

❷ This week we will introduce you to makefiles, one of the foundation stones of developing even simple programs.

While our programs here are necessarily short and self-contained I hope that you can imagine that this is not how mid-sized – never mind large – systems are developed. Code is

distributed across many source files and include files (`.h` files) and when code is changed in one or several maintaining a current executable is not always so easy. Enter `make`.

The main thing that the `make` program seeks to assist in is managing the compilation process when your code is spread out across several files. In the `Makefile` you tell it about your files, your programs and their dependencies. That is, you tell it what files are used in the compilation of what programs and how your programs should be compiled (made) and what to do if you update the source code of one of them. In a properly written makefile when a change is made *only those programs affected by the change* should be updated / remade.

There are lots of tutorials available on the web that describe the `make` program so we are going to limit ourselves here to how to run `make`. Because of the way the `Makefile` is written you can either tell it to compile one of the three programs (`glut7`, `glut8` or `spin`) or you can tell it to compile them all. The command for telling `make` to just update the `spin` executable would be `make spin`. In the terminology of `make`, we call `spin` the *target*. Often you want to have the system "do everything and remake everything that is not up to date". You can do this with the command `make all` or, even shorter, just `make`, since the default target is `all`.

❸ The programs this week are `glut7`, `glut8` and `spin`. These are to be made form their respective source files `glut7.cc`, `glut8.cc` and `spin.cc`. The first two are to be had from the extremely useful site `http://www.lighthouse3d.com/tutorials` and these will be the basis for the `spin` program.

You should not be put off by the complexity of the source code of the two glut programs this week. By running the programs and watching its behaviour you can learn a lot from the source code you are presented with.

In order to run the programs you will need to compile them first. Follow the `make` instructions you were given earlier for compiling them with:

```
make glut7 glut8 spin
```

Note carefully the compile commands that `make` generates. Thus `make` is not a compiler but is, more usefully, a program to manage compilation.

There is very little difference between the `glut7` and `glut8` programs. In the former a menu is created that changes the display colour of the triangle; in the latter a second level of menus (a sub-menu) is introduced to do exactly the same thing. Your task for this week is to implement a menu system like this for the `spin` program (see below). Note in both cases how a menu is

1. created
2. associated with a mouse button
3. tied to an event handler

All of this goes on in the `createGLUTMenus()` function. The function `glutAddMenuEntry("name", VALUE)` adds the entries to the menu saying that the menu entry should appear as `"name"` and

the corresponding return value should be `VALUE`, which will be processed in the `processMenuEvents()` function.

Note how simple the structure of the code is in all three cases. While you may not *understand* every line down at the detail level, since they are so similar to the programs you have been seeing over the past weeks you should certainly be able to follow the general flow of control of the program.

You will spend the remainder of the lab modifying the `spin` program.

❹ You should now modify the program so that its behaviour can be changed from a menu. The menu should be associated with the **right** button and it should have three top-level options:

1. direction
2. speed
3. quit

The direction option should, in turn, have three options:

1. reverse
2. clockwise
3. anti-clockwise

This will control the direction of rotation of the square.

The speed option should have at least a way of increasing or slowing the speed of rotation but you might want to provide something a bit more flexible such as adding (subtracting) different increments and having a twice (half) the speed option.

What would be nice also would be adding a fourth menu option that controls the *colour* that the square is painted in.

❺

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

```
handin -m cs4815 -p w04
```

That'll do pig, that'll do.