MTRN3500 - Setting up Programming Environment

Supplied Code and How to Set Up Programming Environments

1 Supplied Code Overview

You should begin this assignment by downloading the base code ZIP file from Moodle. A number of files have been provided to allow you to view and navigate a simple 3D world so you can test your code. See Section 2 for details on downloading and setting up the given base code.

Once the downloaded code is up and running the list of controls in Table 1 will allow you control the vehicle and to move the virtual camera.

Table 1: Base code commands

Control	**Description**
Arrow keys	Drive vehicle forward/left/backwards/right.
W,A,S,D	Move camera forward/left/backward/right.
C	Descend camera vertically.
(space)	Ascend camera vertically.
Mouse drag	Rotate the camera's viewing direction.
0 (zero)	Move the camera to the origin.
P	Move the camera to vehicle pursuit position.

Additionally, you can exit the program by pressing Escape. See the use of the "`KeyManager`" class in the main source file for more information on how keyboard events are linked to the virtual OpenGL Camera. In short, keys that are pressed once (such as getting the virtual camera to move to the origin) are handled the normal way via GLUT (Open**GL U**tility **T**oolkit). To handle multiple keys being held down at the same time, the additional functionality of the custom "`KeyManager`" class is used.

In the supplied code, zero degrees of rotation means you are facing a direction parallel to the positive x-axis. The vertical axis is the y-axis and we are using a right hand coordinate system.

The entire vehicle class is given and you simply have to use it. No class derivation is required.

2 Setting Up

The following steps will get you up to compiling and running the base source code provided on Moodle. There is a separate section for each of Windows, Linux and Mac OS X. Although there are different steps for each operating system, all of the base assignment code can be found in the **AssignmentGL-Base.zip** file on Moodle. All supporting headers, libs and dlls can be found in the **AssignmentGL-Support.zip** file on Moodle.

2.1 Windows

These steps are based on Visual Studio 2010, but should also work for later Visual Studio versions. As a student of the University of New South Wales, you are eligible to download a free student-licensed version of Visual Studio Professional by visiting Microsoft Dreamspark (`www.dreamspark.com`) online.

2.1.1 Setting up a new project : Windows Console Win32

1. Open Visual Studio and create a new project.

2. From the Templates panel on the left select **Visual C++** and then **Win32**.
3. Select **Win32 Console Application**.
4. Enter a project **Name**. (For example "*Assign2*").
5. Choose a **Location** to place the project.
6. Leave the **Solution Name** the same as the project **Name**.
7. Click **OK**.
8. Click **Next** to go to the "Application Settings" page.
9. Make sure **Application Type** is "Console Application". Make sure **Additional options** has "Empty project" checked and "Precompiled header" and everything else unchecked. Make sure **Add common header files for** has every option unchecked.
10. Click **Finish** to finish setting up the project.

2.1.2 Setting up a new project : Windows CLR Empty Project in Visual Studio 2019 Community Edition

1. Open Visual Studio and create a new project.
2. In the dialogue box that opens titled "Create a new project", type "CLR Empty Project" in the search box at the top. This should show CLR Empty project in the list which is a C++/Windows/Console application. Click on "Next".
3. Enter a project **Name**. (For example "*Assign2*").
4. Choose a **Location** to place the project.
5. Leave the **Solution Name** the same as the project **Name**.
6. Click **Create**.

2.1.3 Including the base source code

1. Download the Assignment ZIP (**AssignmentGL-Base.zip**) file and extract to somewhere easy to find.
2. Back in the new project you've just created find the **Solution Explorer**. If you cannot find the **Solution Explorer** go to the **View** menu then click **Solution Explorer**.
3. A cpp file will have been created with the same name as the project, for example *Assign2.cpp*. Open this file and delete the contents (we will be using our own pre-defined main function).
4. In the **Solution Explorer** right-click "Header Files", select "Add" and then "Existing Item".
5. Locate the folder where you extracted the Assignment ZIP files. Highlight all the `hpp` files and click "Add". (You can select a group of files by holding Control and clicking on each file).
6. In the **Solution Explorer** right-click "Source Files", select "Add" and then "Existing Item".
7. Locate the folder where you extracted the Assignment ZIP files. Highlight all the `cpp` files and click "Add".

All the given header and source files are now set up and included correctly.

2.1.4 Setting up OpenGL headers, libs and dlls

1. Download the `AssignmentGL-Support.zip` file from Moodle and extract the files to an easy to find location. In the Win32 folder there should be three folders: `dlls`, `include` and `lib`.
2. Copy the `include` and `lib` folders to an easy to find location, for example `C:\include` and `C:\lib`.
3. Open the `dlls` folder and copy the contents to `C:\Windows\System32`. Alternatively you can place the dll files in the same directory that the compiled `exe` file will eventually be compiled to. If you keep Visual Studio in "Debug" mode (this is the default mode), the `exe` will eventually be compiled to a "Debug" folder in your project files.

2.1.5 Compiling and Linking with OpenGL

1. Back in the Visual Studio project open project properties (either via `Alt+F7` or via the "Project" menu then "Assign2 Properties").
2. Navigate to the **Configuration Properties**, **C/C++**, **General** section.
3. Set "Additional Include Directories" to the location where you placed the GL files `include` directory, for example `C:\include`.
4. Navigate to the **Configuration Properties**, **Linker**, **General** section.
5. Set "Additional Library Directories" to the location where you placed the GL files `lib` directory, for example `C:\lib`.
6. Navigate to the **Configuration Properties**, **Linker**, **Input** section.
7. On "Additional Dependencies" click the down arrow and the "Edit".
8. Add to the textbox, one per line: `opengl32.lib`, `glu32.lib`, and `glut32.lib`
9. Click **OK**.
10. Click **OK**.

Everything should be properly set up and ready to compile and run. Press `F5`. This should compile and then run the program. If a window appears then it worked. Try and move around the basic 3D world using the controls outlined earlier in Table 1 of this document.

If it didn't work look at the **Output** window, usually at the bottom of Visual Studio. If the **Output** window is not visible you can select it from the "View" menu. If you can self diagnose the problem that is great, but you are also encouraged to post a topic on the Assignment Two **Discussion Forum** in Moodle.

2.2 Linux and Mac

These steps are based on using the g++ compiler.

Linux and Mac OS X are relatively the same, except on most Linux distros the compiler and libraries are installed by default. In Linux, if you try to run the command g++ and the shell complains that it cannot find the command, you should install the compiler by running the following command:

```
Ubuntu/Debian:
apt-get install build-essential
RedHat:
rpm install build-essential
```

On Mac OS X you will need to install Xcode. Xcode is available via the App Store for free, but a link can also be found on `http://developer.apple.com` if you have a free Apple developer licence. Xcode is something like 3 or 4 GB, so be warned.

2.2.1 Setting up a folder structure

1. Create a new directory called, for example, `assign2`.
2. Download the **AssignmentGL-Base.zip** file and extract the `hpp` and `cpp` files to the newly created directory.

2.2.2 Setting up OpenGL headers and libs

On Mac OS X, downloading Xcode will automatically set up OpenGL, GLU and GLUT in the correct place. On Linux you should be able to `apt-get install` (Debian/Ubuntu) or `rpm install` (RedHat) the correct packages. If on Linux and you can't install the headers and libs using a packing service, use the following backup steps:

1. Download the **AssignmentGL-Support.zip** file from Moodle and extract the files to an easy to find temporary location. In the Linux folder there should be two folders: `include` and `lib`.
2. Copy the `include` folder to a location that's easy to find (for example: `~/include`).
3. Copy the `lib` folder to a location that's easy to find (for example: `~/lib`).

Please ask for assistance in the Assignment Two **Discussion Forum** on Moodle if you run into problems.

2.2.3 Compiling and linking with OpenGL

It is recommended to put the following text in a `Makefile` file in the assignment code directory.

```
LIBS = -lGL -lGLU -lGLUT
SRC =  <list all your cpp files>
LIBDIR = -L~/lib
INCDIR = -I~/include
all:
<tab> g++ -o run $(SRC) $(LIBS) $(LIBDIR) $(INCDIR) -g
```

On Mac OS X you may not need to specify the `LIBDIR` or `INCDIR` for OpenGL support files as Xcode might have set them up properly for you. Also, Mac OS X users might find it easier developing directly in Xcode instead of using g++ directly. There are plenty of tutorials online for setting up an OpenGL/GLUT project with Xcode, but if you need further assistance, please ask in the Assignment Two **Discussion Forum** on Moodle.

By creating a `Makefile` you can simply type `make` at the command line in the assignment directory and it will compile and link your code to from an executable. Remember, when you add more cpp files make sure you make the necessary changes to the `Makefile` as well.

You can then run the program by typing:

```
./run
```