University of Southampton	*School of Electronics and Computer Science*	*Coursework (3 of 3) Instructions*
Module: COMP 3004	Title: Principles of Computer Graphics.	Lecturer: Dr. J N Carter
Deadline: 11 January 2011	Feedback: Deadline + 10 working days.	Weighting: 25%

Instructions

3D World Simulation.

Introduction

This coursework is all about building a 3D animation of a modestly complex scenario. In the past themes such as Moon Landing, Mars (as in bars and rovers), The Sea and Robots have been used. This year the theme is 'Flight'.

You may chose to implement this theme any way you wanted. Here are a few thoughts:

- Space flight – rockets and space stations.
- Early flight – bi-planes etc.
- Commercial flight – a DHL plane landing, unloading, loading and taking off.
- Man powered flight – pedal power.
- Ancient flight – Icarus got too close to the sun in Greek legends.
- Airports and landing strips.
- Acrobatic flight – Looping the loop etc.
- Sci-Fi or Steam Punk flight – use your imagination.

If the simulation is of a game then it should only run as a demonstration and aside from controlling the view point, no further interaction is allowed.

What is important is that in your animated world you should have

- Many static objects.
- At least two moving object.
- An interesting landscape, seascape or spacescape (!).
- User controls, see later.

- A viewing position controlled by the user, see later.
- ~~And an animated tour, see later.~~

Please remember that it is more important to produce something that looks good than something that is realistic[1]. To ensure this, as many marks are allocated for visual and artistic style as pure technical achievement. For example in the past excellent marks have been achieved by people who exploited a simple technical style, but using it to produce coherent and believable animation. An abstract or surrealistic design would also attract good marks if it was well executed.

Expanding on this, your coursework submission should be built on what you did for Coursework I along with the following additional elements:

1. A pleasing/interesting scene, your interpretation of my theme. There must be motion in the scene.
2. ~~An automatic tour of the scene, triggered by pressing the "T" key. This should take the viewer through the scene showing **all** its features to best advantage, and should last at least 10 seconds, generally longer. The motion of the camera must be smooth and slow enough to allow the viewer to appreciate what he or she is seeing. The scene animation should run on automatic during this period.~~
3. Pressing the "P" key must position the camera at the viewpoint from which your submitted picture was taken (see later.)
4. Pressing 'Q' should exit the program, and optionally 'H' should either cause a help screen to be displayed (advanced) or for it to be printed on *stdout* in an attached DOS or Shell window.
5. A rough and colourful landscape (a mesh of triangles for example), if appropriate.
6. Lights, materials and texture mapping are all allowed but not required.
7. The moving or animated objects should be a composite of basic primitives.
8. The user must be able to move the point of view around the environment.

To aid you in your work you will also find example programs which makes use of GLSL and vertex and fragment shaders on the module web site.. You may use these as starting points, however you must acknowledge their use.

Restrictions, Rules and Regulations

General Rules

You may choose to develop your code on any convenient platform. However you must bear the following general rules in mind:

1. The code should use the OpenGL, GLew, SDL and svl (vector) libraries, and all programming must be done in C or C++. The use of the standard template library (STL) is encouraged. You might also want to use the GLM maths library. The header files for this should be included with your files, it is not installed on the test machines by default. On a Mac you might want to use the Glee extension module. Just include the source code in your project.
2. Your submitted code must pass the automatic hand-in test which is compiled on a Windows 7 computer. It must compile without errors or warnings. The compiler is set so that warnings are treated as errors. If there are any errors then no attempt will be made to run your coursework, and you will receive zero marks.

---

[1] If you know the Hitch Hikers Guide to the Galaxy then you will understand that the power of art over powers gravity and that objects can float in mid air as long as they look good.

3. The coursework will be evaluated on my Windows computers and as such must compile with the current MinGW.
4. You may use any version of OpenGL but the handin must contain either the file opengl2.txt or opengl3.txt to indicate the version and ensure correct compilation.
5. You must submit a program that will compile and execute on any ECS workstations (i.e. Windows, Linux and OSX), with recompilation only.
6. If using a Mac you might want to use the Glee extension management module. Just include the source code in your project. You should also include appropriate pre-processor directives. This is because Apple has chosen to install header files in a non-standard place. Watch out for further instructions if you are a MAC user.
7. If there are irresolvable problems with your work I will ask you to demonstrate it to me in the computer lab. Showing compilation from te submitted source and execution.
8. The coursework will be evaluated on systems with different configuration to yours. Therefore you must make no assumptions about available screen size, frame refresh rate or the availability of any functions other than those in the core of OpenGL 3.3 or core OpenGL 2.1. The program should be written so that motion is not related to the video refresh rate.
9. Your submission must degrade gracefully on machines that are less powerful than the machine you developed on. Specifically the program must remain responsive and motion must never be jerky. You will lose marks for this.
10. You may use any extensions to OpenGL to improve performance; however you must test the run time environment to determine if the extension is supported. If it is not then your code must allow for this and degrade performance gracefully. Only OpenGL Core 3.3 (or less) is guaranteed to run on the primary test machine.

Stick to standard C/C++ and SDL/GLew, this works just about everywhere. C++ programmers may assume that the Standard Template Library is available.

The on-line checker[2] only checks that program will compile and link. If it does so but fails to compile or execute properly on my test machine then you may be given the opportunity to demonstrate a working version. If however, the package as submitted could never have run, eg. missing, corrupted or incorrect files then a penalty of 20% plus 0.95^n, where n is the number of working days late will be applied to any resubmission. If your program completely fails to work on the test machine and I am unable to re-build it within a reasonable time, and cannot contact you before the exams you will normally be awarded zero for the coursework.

Technical rules

1. C or C++ only, Objective C, C# or Java are forbidden. It is preferable that you stick to C.
2. Please don't write C++ and give the file a C extension.
3. OpenGL or operating system header files like gl.h or windows.h must not be submitted.
4. You may use either:
    - **OpenGL 3**: OpenGL Extensions and OpenGL vendor specifics or functions later than core Version 3.3 are discouraged. If you chose to use features of OpenGL 4 and later you must check for their availability using Glee/GLew. The failure of your program because it is not compliant with OpenGL 3.3 may result in zero marks.
    - **OpenGL 2**: OpenGL Extensions and OpenGL vendor specifics or functions later than core Version 2.1 are discouraged. If you chose to use such features you must check for their availability using Glee/GLew. The failure of your

[2] There have been issues with the online checking tool, so it may not be available.

program because it is not compliant with OpenGL 2.1 may result in zero marks.

5. The GLAux library is not platform independent and should not be used.
6. Any platform specific extensions/features are forbidden.
7. Coursework should run on Windows, Linux (Unix) or OS X with recompilation only as specified above.
8. Full Screen mode is forbidden.
9. Use of any SDL functions other than those needed to display the graphics, deal with keyboard or mouse events or query the system are forbidden. For example, music or other sound effects will result in an automatic mark of zero.
10. The SDL Image Extension should be used to load texture files. All such files must be stored in jpeg format with a quality factor of at most 70. All other known image formats **may be deleted automatically** in the test environment.
11. You are allowed to use third party tools to generate your graphics objects. The only proviso is that all the rendering code is your own and that you give full attribution and credit. For example if the '*glm*' module is used to read in models then you are not allowed to use the rendering functions in the package. You must write your own, interpreting the data structures with your own code.[3]

The exercise

Advanced features such as lighting, shadows, reflections, environment, bump and texture mapping may be used, but are not required to obtain a good mark. All build-in functions of OpenGL may be used. Any version of OpenGL is acceptable, see general rule 4.

While no marks will be awarded for programming style and structure, it is advised that you follow all normal good programming practices for this project.

All elements of this coursework must be handed in via the Electronic Coursework Submission system, in a zip file. The zip file should contain only ".c", ".cpp" or ".h" files, associated data files, shaders and the required files. The automated evaluation procedure will attempt to compile all the submitted ".c" or ".cpp" files into a single program.

1. Don't mix c or c++, th compiler will get a headache and things may not work.
2. Make sure all the files I need to see unpack in the top level of the directory. I will not mark files found in sub-directories.

The screen shot must be in jpeg format and must be called '*screenshot.jpg*' and the descriptive text file must be called '*readme.txt*', and must be plain text without any formatting. The marking system will not look for files with other names. Failure to provide these files will result in the loss of marks.

As in Coursework I if you use an Integrated Development environment place all the files for this in a suitably labelled directory, but then copy all the code files into the root directory of the zip file. If the test script fails to find files where it expects them you will be warned and receive zero marks for the coursework.

There are no restrictions on the names or the number of 'c' or 'cpp' files submitted. Other than that, names should not be case sensitive, i.e. names like `X.c` and `x.c` must not be used together. The automatic system will attempt to build an executable program by compiling all of the files together. Don't mix '.c' and '.cpp' together. This tends to send compilers mad.

---

[3] i.e. glmDraw and glmList may not be used or their code reused.

There are no restrictions on how many times you may submit your coursework. However only the latest submission before the submission time will be considered.

Penalties for late submission will be applied at 10% per day, up to a maximum of 5 days.

Required file layout	
No IDE	With an IDE
/ +screenshot.jpg +readme.txt +mycode.c +mycode.h	/ +idecode +mycode.c +mycode.h + … IDE files +screenshot.jpg +readme.txt +mycode.c +mycode.h

In some cases you may be asked to resubmit your coursework after the deadline has passed, usually when I judge that there is a problem, there will normally be no penalty. However re-submission must be by the Automatic system. I will not accept whole or partial coursework submissions by e-mail, disk or other media.

The usual plagiarism rules apply, and automatic plagiarism detection may be used. However, you are free to borrow from the many examples available, on the course web site or on the WWW, provided you quote your source, both in your code and if applicable in the accompanying text file.

There are NO marks for anything other than the graphics and the programs animation sequence. For example, you might submit a complex physics driven game format like Quake or Doom, in this case there would be no marks awarded for effort spent on an AI/Physics engine to control the environment.

There is no need to have a realistic behaviour. You could pre-compute the trajectory of all objects. There is no need for the trajectory to be accurate.

Key	Function
<ESC>, Q	Exit the program.
P	Move to predefined location where screen shot was taken

Y (optional)	Switch to alternative view point 1.
U (optional)	Switch to alternative view point 2.
~~T~~	~~Start the tour, ignoring all key presses except E, Q and <ESC>~~
~~E (optional)~~	~~Exit the tour mode otherwise ignored.~~
<LEFT>	Turn camera view direction to left.
<RIGHT>	To the right.
<PAGE UP> (optional)	Increase the elevation of the camera
<PAGE DOWN> (optional)	Decrease elevation.
<UP>	Increase the forward speed of the camera.
<DOWN>	Decrease or make the camera move more slowly. If the speed decreases to zero it should stay in the same place. The camera does not have a reverse mode unless it turns through 180 degrees.
<SPACE> (optional)	Stop all motion.
R (optional)	Reset all animation
Key functions (mandatory).	

The basic user interface to control the camera must use the arrow keys, LEFT and RIGHT to turn, UP to speed up and DOWN to slow down. Optionally the SPACE bar can be used as a brake. PAGE UP/DOWN may be used to control the elevation of the camera.

The user or point of view must occupy physical space in the world, and must not be allowed to penetrate the walls or static objects. Collision detection and interaction with moving objects is not required. You might like to think of the viewer as a camera equipped oil drum.

Deliverables.

Submission Contents

A single zip file containing

- All source code and data files for the project. The automatic evaluation procedure will compile all ".c" or ".cpp" files it finds and combine them into a single application. These must reside in the root directory.
- Data files must also reside in the root directory.
- A text file 'readme.txt' (2 pages maximum), which details:
  - How to operate the program(s) i.e.: which keys do what, and how the mouse is to be used.
  - A list of all the files on the disk, with brief comments on their contents.
  - A brief description of how to build the program on

your chosen platform.
    - A very brief description of how your program works.
    - Any source of inspiration or help, e.g. Code reused from GLUT examples.
- The text file should be included in your submission, as a file called *readme.txt*, should be formatted as simple text. Word or other word processor documents will not be read. In itself it is not worth any marks, so don't go overboard. However its absence or if it does not contain the above will lose you 1 mark.
- One screen dump of your world, in jpg format, this should show your world to best advantage. Its absence will lose you a mark. The file must be called '*screenshot.jpg*'.

Marking

Marks will be awarded for achieving the above goals, handing in all the required elements, technical merit and artistic interpretation. A possible mark scheme is as follows.[3]

Activity	**Note**	**Mark out of 25**
User Control	Based on the ease of controlling the viewpoint.	2
~~The tour~~	~~How interesting it is?~~	~~2~~
Static Complexity	Based on appearance of static landscape.	~~3~~ 4
Dynamic Complexity	Depends on the detail in the animation.	~~4~~ 5
Technical Difficulty	Use of OpenGl features, not necessarily advanced ones.	7
Artistic merit	This is very subjective, and is why the coursework is double marked.	7

To calibrate the lower bound to the marks, consider that all of the marks above are based on added value with respect to Coursework 1. If you submitted your coursework 1 submission or something equivalent in action and appearance you would score zero.

This coursework will be double marked and the average mark used.

Hints and FAQ (see the web site for more).

- How do I read image type ".x"? Use SDL Image, and remember all picture files submitted must be jpeg format.
- Keep it simple, don't attempt to build the whole program in one go. Write a number of simple programs to explore all the problems you encounter before attempting to put everything together. In this way you will automatically satisfy the basic requirements.

- Don't be too ambitious, get something to work, submit it, and then build on it and submit again.
- Use a structured approach, develop subroutines to draw your primitive objects and then use the geometric transformations to place them in your universe.
- On-line there are plenty of examples, and there are extensive manual pages for all the OpenGL and SDL functions that you might use. Read the relevant manual pages, and then re-read them. There are many more options than those given in the examples and lectures, and if you can discover some of these your task may be easier. Feel free to reuse any of the example code, however failure to attribute your sources will be severely penalised.
- Remember that a simple scene, which works and looks interesting, will get better marks than a complex mess that falls over.
- Try and separate modelling from drawing, i.e. don't mix up calculations relating to animation and rendering.
- Any students who feel that their preparation and background will make this project impossible should see me as soon as possible.
- The specification for this project is only loosely defined, it is your responsibility to interpret this document. If you have any doubts, either *ask me to clarify the situation* or *make your own design decisions and document them in the documentation text file*.
- The commonest question is '*What aspects of OpenGL can I use in this coursework?*' The answer is simple
- For the first coursework, you drew everything using polygon meshes that you have constructed either off-line or in your program, i.e. You draw everything with glBegin(); …; glEnd();. For the this coursework, anything goes, provided it is part of the OpenGL 2.0 standard or the OpenGL 3.3 standard.
- I'm also asked '*Can I use 3rd party (commercial) software to generate my polygon meshes?*' Again the answer is simple. For this coursework, YES, but is it worth the effort in learning to parse a new file format? Third party programs with automatic code generators are another matter, please discuss their use before committing to them. Please remember that your code will only be built against the OpenGl libraries, or in the case of C++ program the Standard Template library.
- In the past people have been successful using the Alias Wavefron '.obj' format object files. There are object files and readers on the web. Many 3D packages can import and export these files.
- I've been asked, '*Why cant I get OpenGL to work on my computer?*' a number of times. The simple answer is that I don't know as it works on all of mine (Windows XP, 7). I'm happy to chat about this after lectures but if you really need help please go to ECS helpdesk.
- In general I will answer questions after lectures and will respond to e-mail queries as soon as I can, please mark them "COMP3004 Coursework Query". Any student who wishes to talk to me for a longer period should e-mail for an appointment

Submission

Please submit your program in a zip file as specified above, to the C-Bass system. There is no penalty for multiple submissions.

Relevant Learning Outcomes (LOs)

1. Show that you can go beyond simple concepts and generate a complex animation.
2. Demonstrate a familiarity with the capabilities of OpenGL and what can be achieved with them. This does not mean use every advanced trick. Simple is often better
3. Show an understanding of advanced concepts in Computer Graphics.
4. Demonstrate some creativity in content.

Marking Scheme

See above.

Late submissions will be penalised at 10% per working day. No work can be accepted after feedback has been given.
You should expect to spend up to 25 hours on this assignment. Please note the University regulations regarding academic integrity.