School of Computer Science
Coursework Assessment Proforma

Module & Lecturer: CM0304 Graphics, F. C. Langbein

Date Set: 19th October 2009

Submission Date: 7thDecember 2009

Submission Arrangements:
Upload your solutions onto blackboard in the CM0304 module assignment section by the 7th of December 2009, 19:00.
If you are having problems with blackboard, send the archive to `F.C.Langbein@cs.cf.ac.uk` by e-mail by the above deadline with the Subject "CM0304 Coursework". All e-mails will be acknowledged. I will use the last e-mail sent to me before the above deadline, and only use any blackboard version if no e-mail at all has been sent to me. But I prefer to receive only one version!

Title: Graphics Coursework

This coursework is worth 25% of the total marks available for this module. The penalty for late or non-submission is an award of zero marks. You are reminded of the need to comply with Cardiff University's Student Guide to Academic Integrity. Your work should be submitted using the official Coursework Submission Cover sheet (include it with the submission to show you are aware of its contents).

Instructions

Complete the OpenGL programming task as described in the attachment. Submit a tar.gz, tar.bz2 or zip archive with the following contents:

- the *source code* with *clear, concise comments in the code*;
- a *plain text file* called 'README' *briefly* describing how to compile and run the program and on which platform(s) you have tested it;

Your program has to be compilable, but you should not include any object files or executables. Programs that cannot be compiled will be awarded zero marks. If I cannot compile your program I will contact you, but only clarifications of how to compile the submitted source code on a suitable platform will be accepted, and no amendments to the source code will be accepted.
You may use any suitable programming language on any platform with OpenGL support available at the Cardiff School of Computer Science. Your program may make use of any core libraries of your chosen programming language (standard C/C++ library, standard maths library, STL for C/C++; core API for Java). But you may only use the standard OpenGL libraries for graphics programming (GL, GLU, GLUT with platform dependent windows interface for C/C++; the JOGL API and any Java version of the GLUT code) and any other specialised libraries mentioned explicitly in the attachment.
You may reproduce small code fragments from

- the CM0304 handouts, notes, web-site, tutorials and lab classes;
- any recommended and background textbooks for CM0304 as available from the printed text, CDs/DVDs, websites, etc. for the textbook

provided that your sources contain clear reference to the origin of the code. You may use these code fragments to setup and initialise OpenGL, the windows environment, obtain example models, etc. But you have to write your own code to implement any of the features required for the tasks described in the attachment. You may not reproduce code written by any other student or code obtained from any other source not mentioned above. If you are in doubt about whether you may include a code fragment that you have not written yourself, ask the lecturer.

Criteria for Assessment

Credit will be awarded against the following criteria.

	Excellent (I)	Good (II.1)	Adequate (II.2)	Poor (III)
Functionality: to what extend does the program realise the task described? **40%**	efficient and complete implementation; all special cases are considered	feasible implementation, but not optimal; not all special cases are considered	progress towards a full implementation, but not fully working with major deficiencies	little or no progress towards implementation; approach not suitable
Design and Structure: how clear is the structure of the code and how well are data structures and algorithms used? **40%**	well structured code with highly suitable data structures and elegant, clear algorithms	good structure with suitable data structures and algorithms; sometimes not optimal	attempt at using appropriate data structures and algorithms visible; sometimes structure is confusing	code is mostly obscure; little or no structure is visible in use of data structures and algorithm design
Code Documentation: how easy is it to understand the code with the comments provided? **20%**	clear, concise comments describing ideas and high-level structure without unnecessary detail	clear comments about high-level structure and ideas; sometimes incomplete or too focused on details	some comments about the structure and ideas present, but hard to follow and too focused on details	hardly any comments or only very confusing or low-level comments about single instructions

Feedback on your performance will address each of these criteria.

Further Details

Feedback on your coursework will address the above criteria and will be returned in approximately four weeks. This will be supplemented with oral feedback in the lectures and/or tutorials. If you have any questions relating to your individual solutions talk to the tutor or the lecturer.

CM0304 Graphics Coursework (2009/10)

Write a program using OpenGL to render a 3D scene, which can be navigated in a walk-through fashion. You are free to choose the scene, but the scene and your program should exhibit the features described below. The associated marks as assigned to each assessment criterion are listed in brackets. Emphasis is placed upon the implementation of the functionality using OpenGL, the appropriate use of data structures and algorithms for the design and structure of the graphics and geometry elements in your program, and clear and concise comments in the code describing its structure and the ideas used for the implementation. Note that you have to submit a working program, which is compilable from the sources you provide and uses only the OpenGL interface as available for the programming language you are using to render the scene.

1. Rendering Framework [Design: 2, Functionality: 2]

   Implement a framework to render a small shaded 3D scene with OpenGL which properly initialises the OpenGL pipeline, the windows environment, and the light sources. It should provide the following functionality:

   (a) A rendering system which sets up the scene using display lists where possible and continuously renders the scene. It is important that the rendering system is clearly structured to easily add and remove objects, etc. from the scene (see below for the object types).

   (b) A navigation system which allows the user to display the scene from an arbitrary camera position with keyboard and/or mouse control to facilitate moving about the scene by adjusting the view direction and moving forward/backward (or a similar 3D navigation system). This should change the view in a natural and predictable manner.

2. 3D Objects [Design: 4, Functionality: 4]

   Use the rendering framework to create a scene which contains shaded objects of the types described below which can all be rendered as polygonal meshes. Your scene should contain at least one of each object type. The marks are on the way the types are realised, not the amount of objects.

   (a) Parametric surface: a curved surface described by a parameter function which is not a plane, sphere, cylinder or cone. The approximation accuracy of the surface by polygons should be adjustable by a parameter.

   (b) Stanford bunny mesh: the Teddy mesh as available from `http://www.langbein.org/fileadmin/teaching/graphics/teddy.ply` (or blackboard) in ASCII PLY file format (see `http://graphics.stanford.edu/data/3Dscanrep/` or `http://www.cc.gatech.edu/projects/large\_models/ply.html` for details about PLY; you may use the PLY tools to read the mesh, but it may be simpler to parse the file yourself under the assumption that all facets are triangles and you only read the x, y, z coordinates). Note that you have to compute suitable normals for the mesh yourself! (To save bandwidth do not submit the mesh with your sources, but assume that it is available in the current directory from which your program is run).

3. Advanced Features [Design: 4, Functionality: 4]

   Implement *one* of the following advanced features. Clearly indicate in the README file which one of these you have chosen; you will only get marks for one of them.

   (a) Smooth the Teddy mesh by applying a suitable subdivision scheme multiple times to it before you render the mesh.

   (b) Compute and render the shadow of the Teddy mesh on a single planar surface cast from a single light source. Recall that shadows are projections, similar to perspective projections.

   (c) Implement a particle effect such as steam from a teapot or smoke from a fire or bubbles in water. Make it look as realistic as possible by, e.g., using polygons and textures and not just separate points.

   (d) Create an object which consists of at least four transparent polygons moving throught the scene, e.g. a rotating glass box. Note that when you render more than one transparent polygon you have to make sure that there are no visible artefacts from every camera position.

   (e) Impress us with something else... But ask the lecturer whether your idea is suitable and clearly state in the README file what you are attempting to do.

4. Documentation of source code by inlcuding comments in the sources. The comments should describe how the program is achieving the various tasks. [Documentation: 5]