DEPARTMENT OF COMPUTER SCIENCE

COURSEWORK ASSESSMENT DESCRIPTION

MODULE DETAILS:

Module Number:	08961	Semester:	1
Module Title:	Real Time Graphics		
Lecturer:	DPMW / DDM		

COURSEWORK DETAILS:

Coursework Assessment Number:	1	of	1		
Title of Assignment:	Software Portfolio				
Format:	Program	Report	Demonstration		
Method of Working:	Individual				
Workload Guidance:	Typically, you should expect to spend between	75	and	125	hours on this assessment
Length of Submission:	This assignment should be **no** more than:	please select **words** *(excluding diagrams, appendices, bibliography, code)*			

PUBLICATION:

Date of issue:	Week 5

SUBMISSION:

ONE copy of this assignment should be handed in via:	E-Bridge	If Other (please state method)	
Time and date for submission:	**Source Code** **Report** **Demonstration**	17:00 Friday 18th December 2009 17:00 Monday 18th January 2010 Between 19th & 29th January 2010	
If **multiple hand–ins** please provide details *(as appropriate):*			

The assignment should be handed in **no later** than the time and date shown above, unless an extension has been authorised on a *Request for an Extension for an Assessment* (Mit Circs) form which is available from the Office or http://www.student-admin.hull.ac.uk/downloads/Mitcircs.doc. The extension form, once authorised by the lecturer concerned, should be sent to Amanda Millson.

MARKING:

Marking will be by:	Student Name

BEFORE submission, each student must complete the **correct** departmental coursework cover sheet and attach it to your work, dependant upon whether the assignment is being marked by student number, student name, group number or group name. This is obtainable from the departmental student intranet at http://intra.net.dcs.hull.ac.uk/sites/home/student/ACW%20Cover%20Sheets/Forms/AllItems.aspx

ASSESSMENT:

The assignment is marked out of:	100	and is worth	100	% of the module marks

ASSESSMENT STRATEGY AND LEARNING OUTCOMES:
The overall assessment strategy is designed to evaluate the student's achievement of the module learning outcomes, and is subdivided as follows:

LO	Learning Outcome	Method of Assessment *{e.g. report, demo}*
1	*Design 3D graphics programs for real-time Graphics*	Software Application, Report
2	*Describe the algorithms and techniques involved in 3D graphics*	Report
3	*Implement 2D and 3D graphics programs in C/C++ using the OpenGL API*	Software Application
4	*Implement simple collision detection/response Algorithms*	Software Application
5	*Use the mathematical techniques of vectors and matrices*	Software Application

Assessment Criteria	Contributes to Learning Outcome	Mark
Quality of implementation	1, 4, 5	70
Quality of OpenGL	3	20
Quality of portfolio	1, 2	10

FEEDBACK

Feedback will be given via:	Verbal (via demonstration)	Feedback will be given via:	Mark Sheet
Exemption (staff to explain why)			
Feedback will be provided no later than 20 working days after the submission date.			

This assessment is set in the context of the learning outcomes for the module and does not by itself constitute a definitive specification of the assessment. If you are in any doubt as to the relationship between what you have been asked to do and the module content you should take this matter up with the member of staff who set the assessment as soon as possible.

You are advised to read the **NOTES** regarding late penalties, over-length assignments, unfair

means and quality assurance in your student handbook, also available on the department's student intranet at: **http://intra.net.dcs.hull.ac.uk**. In addition, **please note** that if one student gives their solution to another student who submits it as their own work, **BOTH** students are breaking the unfair means regulations, and will be investigated.

In case of any subsequent dispute, query, or appeal regarding your coursework, you are reminded that it is your responsibility, not the Department's, to produce the assignment in question.

Assignment Details

08960/08961 Portfolio Assessment

Derek Wills, Warren Viant and Darren McKie

1.0 What is a portfolio assessment?

The two modules 08960 C++ Programming and Design and 08961 Real-Time Computer Graphics are assessed through a single portfolio of work, allowing you to develop a substantial piece of work developed in C++ and demonstrating your ability and knowledge of computer graphics. Throughout the semester you will be asked to design, develop and implement a real-time simulation based upon the scenario outlined in the later sections of this document. You will be asked to complete a number of sub-tasks that together should allow you to successfully achieve the Learning Outcomes for both modules whilst simultaneously contributing to your Personal Development Plans. You will need to record the results of these individual sub-tasks since at the end of the semester you will be asked to include in your portfolio a critique of your work and to reflect upon the complete design / implementation process. This will include an indication of how you might approach future coursework assignments in semester 2. In order that you gain feedback on your progress, your work will be considered at two stages, a mid-term formative (marks will not be awarded but you will receive comments on your progress) assessment in week 7 and a final assessment at the end of the semester. For the final assessment, a clear indication will be given of the marking criteria, which will be based on your design, code, report and demonstration/presentation. A development schedule is also included, and we strongly suggest that you follow this.

2.0 Aim

The aim of this coursework is to produce a graphics demonstration programme written in C++. The demo programme will use GXBase and OpenGL and demonstrate a number of the algorithms and techniques developed within the modules 08960 and 08961. The final programme will have a number of modes of operation driven both from a script file and through user interaction.

3.0 Concept Design

The visual effects within this demo programme will be restricted to occur within a box that has, at any instant, three transparent sides. Various objects will be animated within the box and visualised via either a pre-determined script or through discrete key input from the user.

1) The box will be situated in the centre of the screen. In its 3D space, the Z axis will be vertical with the X and Y axes being horizontal. The box will have 8 spotlights situated at the corners pointing initially towards the centre of the box. Rotation of the box around the Z axis will be controlled either from a pre-defined script (identifying its angular velocity), or through keyboard input (left arrow = rotate clockwise, right arrow = rotate anti-clockwise). The faces with their outer side pointing towards the viewer should be made semi-transparent (allowing the viewing of objects within).

2) The colour, intensity and field of view of each of the eight spotlights should be definable in the script file. Lights should also be animated, being able to point at a particular object and follow it around the room. Optionally, the light colour and intensity should also change over time. Physically, the spotlights should be represented graphically by a small cone whose orientation changes as the spotlight follows an object.
3) One or more objects should be animated in the scene. Objects must include: sphere, platonic solids (tetrahedron, cube, octahedron, dodecahedron and icosahedron), glow ball, particle ball, null object. The size and material characteristics of each object should be included in the script file, along with a description of an animation path. In the case of the platonic solids, the object itself should be loaded from an object file. This file should contain sufficient information to describe the geometry of the object, the related normals, and texture coordinates. In the materials description for an object, an option should be included for texturing and therefore one or more filenames should be included for the textures used.
4) The glow ball is a special object that has 6 regularly spaced colour spotlights evenly distributed over a spherical surface. When this object is in a scene, the 8 spotlights at the corners of the box should be automatically turned off. Like other objects, the glow ball should be able to follow an animation path, illuminating all other objects in the box.
5) The particle ball is similar to the glow ball except that each of the 6 lights is replaced with a particle emitter. Your particle systems should be able to support at least two types of particle, points and textured quads. Particles will be emitted from the surface of the sphere and either fade out or impact onto the sides of the box. On impact they should leave a scorch mark – this should also fade out over time. You may include a number of parameters with each particle system but these should include: type (e.g. point or quad), material/texture, rate of generation, lifetime and scorch texture.
6) The null object has no geometry and is therefore invisible. It is included as an invisible object for lights sources to point at rather than a real object.
7) Animations paths should be definable for each object and included in the script file. Your code should at least support animation paths based on the trigonometric 'sin' function. In this case the (x,y,z) position of the object should be determined by a function of the form (Ax*sin(Bx*t+Cx)+Dx, Ay*sin(By*t+Cy) +Dy, Az*sin(Bz*t+Cz)+Dz) where t relates to time and Ai, Bi, Ci, Di are constant values (i=x,y,z). You should include a key press that starts and stops animation. You should also allow objects to tumble, however, the definition of this as an animation is left to you to decide.
8) Your program should support multiple animated objects in a single scene. Collision detection and collision response between objects is not required.
9) Visualisation of the scene will depend on the current mode selected. Options must include: wireframe, flat shaded, smooth shaded (Gouraud and Phong), smooth shaded textured. Smooth shading using the Phong model should be programmed using GLSL code and need only apply to spheres, cubes and platonic solids. No texturing effects are required from your GLSL code.
10) Special Effects.
    a) Shadows. A simple implementation will allow shadows from objects to appear on the walls of the cube. However, a more complex implementation should allow shadows to appear on all objects.
    b) Fog. A fog effect should be implemented in the box. This should include a representation of the light beams being partially visible as they go through the box.
    c) Bloom. A bloom effect should be implemented around all lights within the box.
11) Your final program should support loading of your script file by name. You will need different script files to demonstrate various effects – you will be notified later what these are. Different modes of visualisation should also be switchable by pressing the 'M' key. This should cycle through wireframe, flat shading etc. By pressing 'N' the display of normals should also be available to the user. Pressing 'A' should start and pause the animation, allowing for closer inspection of the scene. 'S' should toggle between no-shadows, wall-shadows and full-shadows. The script file should allow for the definition of a rotation speed for the box (and all the contents). 'P' should toggle this animation on and off. When off, rotation of the box should be through use of the left and right arrow keys.

12) Keep a log of what you do each week, including what worked and what didn't. You will be required to write a short report on your work at the end of the semester.

4.0 Hints for Writing the Simulation

The following may help you complete your work successfully.

1. Start now!
2. Produce a paper visualisation of your project and check it with Darren McKie
3. Produce a top level software design
4. Prototype your ideas to help produce a more detailed design
5. Test your software at each stage of development
6. Document as you go

5.0 A suggested Schedule

How you plan your time is really your own decision and will depend largely on your previous experience. You will be asked to reflect on how you developed your simulation, at the end of the project. However, included below is an initial suggestion of what you should do and when. This does not include the complete functionality requested and you should only use it as a guide to your own time management. Please note that this is in no way definitive and it is your responsibility to project manage your own time throughout the semester.
You are strongly advised to work through the OpenGL tutorials as early as possible. The plan below indicates the latest time that you should complete them.

Weeks	Suggested Work	Deliverables
2,3	OpenGl Tutorials: 1-6	Portfolio: Storyboards, workplan/timescales (produce a task breakdown and a Gantt Chart).
4, 5	Paper design and initial draft of script input specification. OpenGL Tutorials: 7-12	Check Paper design with Darren McKie. Include in portfolio.
6	Design and implement the interface to read the script information. Prototype the box display with lights. OpenGL Tutorials: 13-17	Portfolio: Document implementation issues.
7	Implement loading display of sphere and cube. Display using wireframe, flat and Gouraud shading	Portfolio: Revised software design. **Wk 7: Formative assessment of portfolio. A demonstration of your software may be required**.
8	Allow animation of objects and lights. Include loading of platonic solids. Add texturing to your objects. OpenGL Tutorials: 18-20	Portfolio: Update design and document object design.
9	Implement glow ball and GLSL for Phong shading. OpenGL Tutorials: 21-24	Portfolio: Document GLSL code
10	Design and Implement particle system.	Portfolio: Document any issues with your implementation
11	Implement Special Effects	
12	Implement Special Effects	**End of week: Submit code (details to follow).**
Christmas break		
13	Complete portfolio and write report	**End of week: Submit portfolio** and documentation.
14		**Demonstration**.

6.0 Report Details

Design (08960)

1. Class diagram(s) containing main classes
2. Class diagram(s) containing service / utility classes
3. A textual description giving the name, role and responsibilities of each class [2 page max]
4. Interaction diagram(s) for significant components of the software design
5. A critique of the design [1 page max]
   Should include details on:
   a. The merits of the design?
   b. Weaknesses of the design?
   c. What has changed in the design?
   d. What would you now do differently?
6. Reflect on you project management, indicating lessons learnt for next semester etc. [1 page max]

Graphics (08961)

1. Document and critique of the algorithms used. [max 3 pages for text/equations, an additional page of diagrams can be included]
   Should also include details on:
   a. Implementation of Phong Shading model in GLSL
   b. Implementation of your particle system

Marks will be lost if you exceed page limits (see handbook for Over length penalties)