DEPARTMENT OF COMPUTER SCIENCE

COURSEWORK ASSESSMENT DESCRIPTION

MODULE DETAILS:

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

COURSEWORK DETAILS:

Coursework Assessment Number:	1	of	1		
Title of Assignment:	Scaletrix Game				
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:	1500 words			

PUBLICATION:

Date of issue:	Week 2

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**	9:30 Friday 19th December 2008 9:30 Monday 19th January 2009 Between 19th & 30th January 2009	
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 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:	Mark Sheet	Feedback will be given via:	N/A
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/sites/home/student/default.aspx. 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 graphical simulation of a slot car racing game. For those of you that don't know what this is, a typical example of slot car racing can be found at http://www.scalextric.com/. For this coursework you will be required to write a program that displays the racing track and through a simple interface, allows you to control two competing cars. The emphasis will be on the visual appearance of the track, the cars, surrounding scenery and a limited number of special effects. The physics of racing will be considerably simplified and only required to a level that makes the game playable.

3.0 Concept Design

The detailed layout of the racetrack is left to you. However certain features must be apparent in your final deliverable.

1. The overall look of the track and scenery should be as close to a real physical game as possible. Therefore the track should be textured with clearly identifiable slots for the cars to run on. The racetrack should be constructed from a set of standard track sections with joints between them. Track sections should be defined as part of a library and specified in an external file. Scenery elements should at least include: grass, trees, starting grid, viewing stands, advertising

hoardings, a sky/night dome, a tunnel and speed limit signs. Textures should be used to enhance the realism throughout your visualisation. The track should have at least two crossover sections (where cars swap from one side of the track to the other.

2. Cars should be controlled by a simple accelerator/break arrangement (A/Z for car 1 and >/; for car 2). The wheels of a car should rotate at an appropriate speed based on the speed of that car. The wheel should turn appropriately as the car goes round a bend. You should be able to label track sections with a maximum speed allowable, defined in a configuration file. If the cars' speed is within 25% of the maximum speed allowed then the tyres should start to smoke using a particle system. If this speed is exceeded then the car should jump from its slot and skid off the track. If it strikes the opponent car whilst doing this, both cars should skid off the track. A time penalty (stored within a configuration file) should be defined that specifies the time passed before the car is placed back on the track at the position of the accident. The construction of the car does not have to be realistic (see the Library document for an example).
3. Day and night racing should be supported and therefore your lighting and shading should be adjusted to reflect this. In the day you should include sufficient lights to give the appearance of a sunlit day. However, you will find that a single light source is unlikely to be sufficient to give this effect. At night, cars should have headlights that illuminate the track. The light source for the headlights of the car should be implemented as a shader effect. You should also include some low intensity light to simulate ambient lighting from the moon and stars. Surfaces should react with the lighting in a realistic way dependent on the position and surface properties.
4. To help assess your work you are also required to support a wireframe viewing mode and a non-textured, smooth shaded mode as well as full texturing.
5. At least the following camera views should be supported:
    a. Overview: this should allow you to see all the track and is probably the easiest mode for racing
    b. Driver views: a camera positioned in either of the two cars looking forward
    c. Crowd views: cameras located in one or more positions around the track that give the view seen by spectators.
6. Advertising hoardings should include static images.
7. A large video screen should be placed near to the finish line that displays the driver's view of the race
8. One section of your track should be flooded. You should be able to see the track below through the water and you should also see the car reflected in the water as it passes over it. As the car passes over the water, a spray of water should be generated from the tyres. The spray should be implemented as a particle system.
9. Trees should be represented as billboards. These should operate in two modes:
    a. Static, the billboards are fixed in world coordinates. In this case the billboard should be made from two texture mapped polygons (with alpha transparency) perpendicular to each other.
    b. Dynamic, the billboards rotate to always face the camera. In this case they can be constructed from a single texture mapped polygon.
10. The games should be playable in two modes:
    a. Single player mode: In this mode the second car should be computer controlled. Since AI is not a learning outcome of this work, this may be as simple as running the second car around the track at a constant speed. However, if you want to use this coursework later when applying for jobs then you may want to include more advanced AI (but you should take care not to spend too much time on this).
    b. Two player mode: In this mode two players race against each other using the keys specified in 2.
11. The final display presented in your simulation depends upon the mode of play specified in 10.
    a. In single player mode, the camera views specified in 5 should be user selectable.

   b. In dual player mode, only two views are allowed. The first is a dual driver display (top half of the screen for driver 1, bottom half for driver 2). The second view is the overview described in 5a, allowing both player to see their cars simultaneously.

   However, as well as the 3-dimensional visualisation you should include a 2-dimensional representation of the analogue speedometer, the lap time and race time. You should also indicate the number of laps remaining (the total number of laps in a game should be set in your configuration file).

12. At the start of the game you should ask if it is a single or two player game. You should then ask for the name(s) of the driver(s).When the final lap is completed, you should indicate who won the race and the time they took. This lap time should be recorded in a log file and a league table of the 10 best laps displayed. You should also include a function to record your race and offer the opportunity for a replay of events, either at the end of the race or at a future demonstration of your work. During the replay the user should be able to select the different views outlined in 4.
13. You should include a sky dome that has moving textures of clouds according to whether the game is in day or night mode as described in 3.
14. At some point in your track layout you should have a tunnel with lights, these should be implemented as spotlights shining down. Due to numeric inaccuracies you may find that your track sections don't align correctly from start to end. The tunnel is a good way to disguise this.

4.0 Hints for Writing the Simulation

The following may help you complete your work successfully.

1. Start Early!
2. Produce a paper design of your racetrack and a storyboard for your simulation
3. Produce a top level 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 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	Initial concept design for the simulation, including storyboards for the main visual components. Specify timescales and workplan. OpenGl Tutorials: 1-6	Portfolio: Storyboards, workplan/timescales (produce a task breakdown and a Gantt Chart).
4,5	Design and implement the interface to read the information for defining the track and other configuration data. Initial perspective prototype display of the track. OpenGL Tutorials: 7-12	Portfolio: Configuration file specification and parser. Top level design of the simulator.

6,7	Further development of track display with textures. Inclusion of the sky dome and grass terrain. Implement other camera views including driver view on advertising hoarding. OpenGL Tutorials: 13-17	Portfolio: Revised software design. Implementation of perspective view. **Wk 7: Formative assessment of portfolio. A demonstration of your software may be required**.
8	Include speed limit signs and remaining advertising hoardings. Place trees into the scene. Simple car controlled by keyboard and AI car. Design particle system design and coding. OpenGL Tutorials: 18-20	Portfolio: include design of particle systems and description of how trees have been included.
9	Implement particle system for both water spray and smoke. Include reflective water into scene. OpenGL Tutorials: 21-24	Portfolio: description of how reflections done on water.
10	Include night scene. Shader code for front lights of the car. Place tunnel with lights	Portfolio: Document shader design and code.
11	Implement speedometer and other information for user. Split screen mode for two player use.	
12	Complete remaining features and bug fix.	**End of week: Submit code.**
Christmas break		
13	Complete portfolio and write report	**End of week: Submit portfolio** and documentation.

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 [keep this brief]
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. How the reflection of the car in the water was achieved;
   b. How the tyre smoke and water spray was achieved.

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