# Module Code: COSE50581



# Module Title: Further Games and Graphics

# Concepts

# Module Leader: Steve Foster

# Title: DirectX Graphics Application

# Year: 2018-2019 Level 5, Semester 1

# Weighting: 35% of module marks

**Submission**

**Hand-in: Monday 14th January 2019**

**Demo by: Friday 18th January 2019**

|  |  |
| --- | --- |
| **Learning Outcome** | |
| 1) IMPLEMENT STRUCTURED 3D ENVIRONMENTS EXPLOITING THE PROGRAMMABLE PIPELINE TO RENDER A RANGE OF LIGHTING AND SHADING MODELS USING AN APPROPRIATE GRAPHICS API VIA AN OBJECT ORIENTED APPROACH. | Application |
| 2) UNDERSTAND CRITICALLY THE MATHEMATICAL OPERATIONS UNDERPINNING COMPLEX MODEL MANIPULATIONS USED TO IMPLEMENT PHYSICS-BASED AND AI-BASED SIMULATION MODELS FOR COMPUTER GAMES. | Knowledge & Understanding |

**Overview**

Over the last few weeks you have been using 'C++' and the DirectX 11.0 API to create and adapt an application that contains elements comprising a graphics engine.

This assignment gives you the opportunity to use these skills to create an application that achieves a range of important features of 3D graphical simulations. The rendering of realistic graphics depends on the selection and implementation of lighting, shading and texturing models in conjunction with appropriate model and viewing specifications.

Ideally, your code should exploit object-oriented design principles to allow the core aspects common to all 3D applications to be specifically configured for the application developed for this assignment.

Your code should have minimal ‘hard coding’, to show the versatility of your design approach. However, you should focus primarily on implementing the graphics techniques that have been presented throughout the module, choosing appropriate algorithms for the tasks set in this assignment.

**Application Requirements**



For this assignment you will be implementing the graphics rendering component of the game “Rocket Ball” ™. Rocket League is a popular physics based game where players compete with rocket powered cars in a football style game. The company creating Rocket Ball hopes to cash in on the success of “Rocket League”, by making a direct copy of this game in as many ways as possible (given time and personnel constraints). In this first phase of development you are asked to implement a graphics application that showcases the algorithms and techniques that will be used in the final game implementation. Your aim is to implement the graphical techniques necessary, the visual aspects (I.e. how good your models/textures are) is not important.

You are asked to produce a 3D graphics application by adaptation of a framework developed during tutorials. Your application should be used to demonstrate the functionality, range, robustness and adaptability of the engine framework you develop.

The application should enable simple user interaction by any suitable means, which will allow parameters of the functions and components comprising the engine to be manipulated by the user to perform specific tasks (E.g. Turning a car, rotating the camera, etc). In addition the environment should implement 3D graphics rendering concepts such as texturing, lighting, shading and projection mechanisms to demonstrate a level of realism relevant to the demands of the application

You will be provided with suitable graphics and models but will be able to use your own assets provided they meet specific requirements shown below.

The assets required must include:

1. **A car**

For high marks this 3D model should be rendered correctly with texturing and lighting applied, and also be controllable by the user to allow basic movement (no physics is required). Sample models will be supplied.



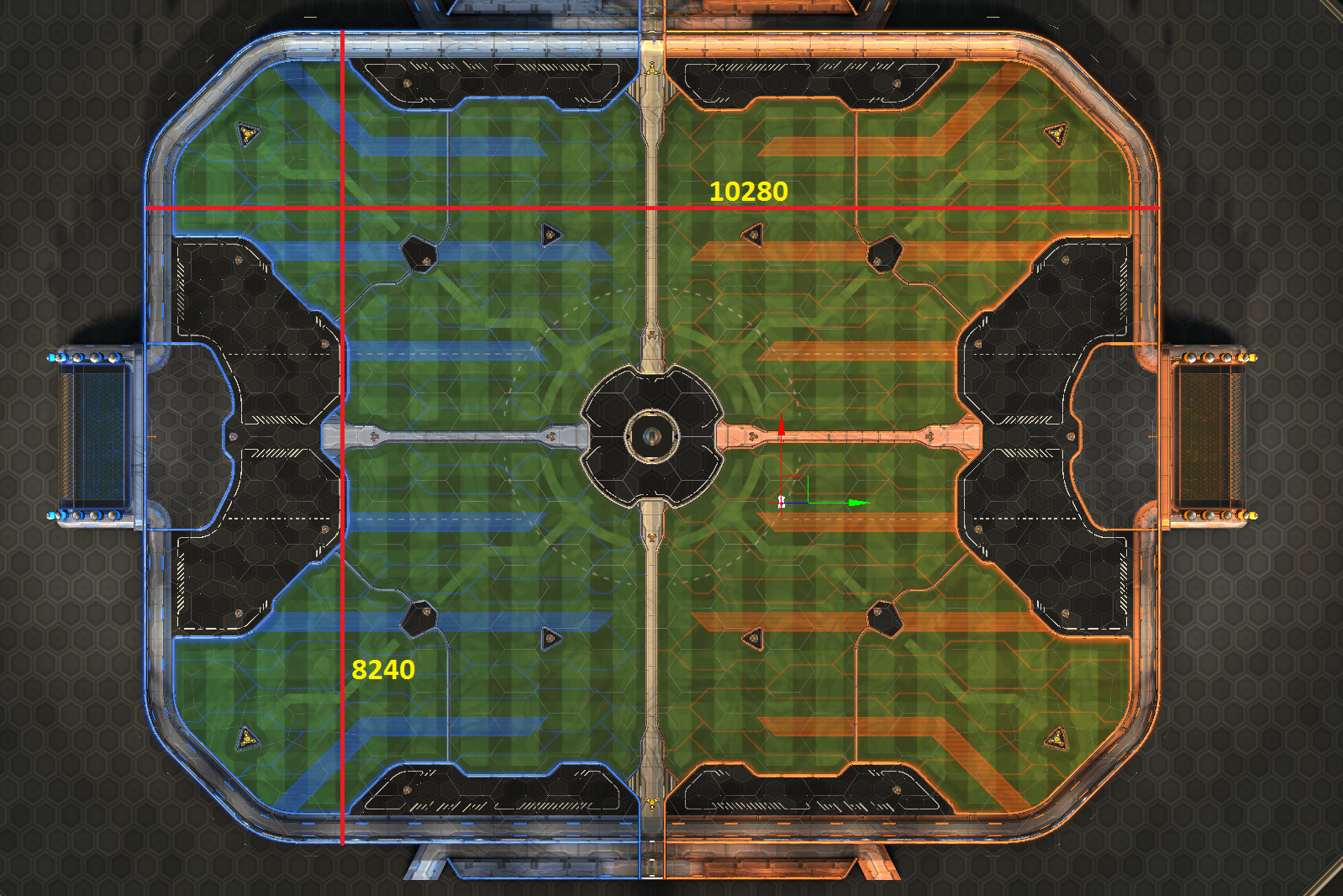
1. **Ground**

You must display the ground below your car. A simple implementation might just be a single textured quad, whereas an implementation worth higher marks might have grass rendered, and spaces for power ups etc.



1. **Outer Cage and Goals**

There should be a large outer cage surrounding the play area, this case does not need to be curved and can be shaped like a cube. The cage should be see through, and have a suitable texture applied to it (this texture will be supplied). There should also be a space in either end of the cage for the goals to be placed, with a goal area behind this. See the image below for a top down view of this:



1. **Ball**

A ball should be placed in the centre of the play area. This should be textured suitably, but does not need to move or have any physics applied to it (this will be added next semester)

The application should contain at least one model constructed using a third party modelling application (You do not personally have to create the model!) and imported into the application.

**Explanatory report**

You should write a short report (Max 2000 words) including the following elements

1. Brief instructions for installation/operation of the application
2. Description of the graphics features you have implemented, and the architecture of your application
3. Overview Class diagram of system
4. Test documentation

**Features**

You are to add interactivity and simulation to the scene to achieve the following features:

1. Load the models employed in the scene
2. Implement texture mapping, lighting and shading for the models
3. Provide 2 general viewpoints (cameras) external to the car: A static fixed view and a top down view
4. Provide 2 dynamic viewpoints fixed to the car: Third person (behind the car), and attached to another part of the car (E.g. bonnet, wing mirror, etc)
5. You should allow simple control of the car such that it can be manipulated relative to the other objects in the scene, rotating and moving around the world appropriately
6. Position the objects in the game using a suitable initialisation file (E.g. a text file)
7. Render a ‘ground’ for the world, suitably textured and lit
8. You should also include one or more additional features to make up the final 20 marks of the total. These features will be marked on individual merit to be assessed by performance, visual enhancement and optimisation employed.

Additional features may include:

1. Clouds and other environmental features outside the arena
2. Skybox
3. Day / night effects
   1. Illumination from car lights and arena lights (night)
   2. Blur
   3. Lens Flare (day)
4. Billboarding (grass, smoke, etc.)
5. Interactive game elements such pickups to charge boost, etc
6. Menu or other GUI components to select and alter rendering of lighting, shading, wireframe and texture detail
7. Particle system applied to the scene (E.g. Smoke behind cars, fog in scene, etc)
8. Point, Spot and Directional lighting. Multiple light sources, Specular mapping, etc
9. Other relevant features by agreement with your tutor (please ask your tutor first if you would like to implement something that is not on this list to check its suitability)

**You are expected to submit:**

1. Using the On-line electronic submission in DIS.
2. A Zip file containing
   1. The source code
   2. A working exe of the application
   3. Brief instructions for installation/operation of the application.
   4. Report.

**You must organise an appointment to present your software and answer questions from your tutor between 14th and 19th January 2019. Details of appointment slots will be provided for prior to the submission date.**

**Failure to attend an appointment could result in the allocation of zero marks for the assignment.**

# Sample mark scheme

***Assessed through inspection of code, report and questioning in viva***

|  |  |  |
| --- | --- | --- |
| ***Mark Component*** | ***Marks***  ***Available*** | ***Description*** |
| **Report** | 10 | Report highlighting sound OO principles and coding standards. Describes features implemented. Class diagrams present. |
| **Loads game assets** | 10 | Loads model/mesh data correctly into appropriate data structures.  Higher marks for:   * Loading model files (OBJ, FBX etc) * Storing and accessing objects using sound OO principles |
| **Renders game assets** | 10 | Assets loaded are rendered correctly from vertex and index data. At least two objects rendered, including basic ground plane. Wireframe mode and depth buffering is also implemented |
| **Lighting and Shading** | 15 | Implements lighting and shading using programmable pipeline. Graded according to level of lighting quality.  Higher marks for:   * Phong shading * No lighting issues/glitches * Ambient, Diffuse and Specular |
| **Control** | 10 | Car moves and rotates appropriately based upon user input, camera follows correctly.  Higher marks for:   * Correct rotation * Car speed increase/decrease |
| **Texturing** | 5 | Implements texture mapping on objects and combines correctly with lighting and shading calculations |
| **Positions assets** | 5 | Assets positioned using configuration file  Should include:   * Starting position of car and ball * Position of assets (Cage, goals etc) |
| **General viewpoints** | 5 | Fixed viewpoints (Top-down and Static) provided |
| **Dynamic viewpoints** | 10 | A third person view and a view fixed to the car are provided |
| **Additional feature** | 20 | Assessed by tutor to reflect the complexity and technical merit of the feature or effect |
| **Total** | 100 |  |