CSC8502 Coursework 2018/19
Due: 23rd November 2018

When a video game is due to be released, the developers will often create a trailer video, which demonstrates all of the most exciting aspects of their new game. These will usually comprise of a number of short scenes that demonstrate a particular level or gameplay mechanic, before quickly cutting to the next, allowing the trailer to pack in as many exciting things as possible into a short trailer video.

You are to develop a program using C++ and OpenGL that replicates the look and feel of these game trailers by showing a series of graphical scenes, each of which demonstrates a different set of graphical elements. The exact number and contents of these scenes is up to you, as is how the transition from scene to scene is handled – it could be an instant cut to a different piece of scenery, a progressive blur, or a fade out and fade in to somewhere else in your graphical environment. Each 'scene' should set up the camera to quickly allow viewers to see the features it has been designed to showcase – the user shouldn't have to touch the mouse and keyboard, but the program may move and rotate the camera over time if you think it is necessary for showing off your work. Not every scene has to contain every feature you have implemented, not does there have to be a narrative connection between them; the scenes are purely a method by which you can demonstrate your abilities in a concise manner.

The goal of the coursework is to demonstrate as many high quality graphical techniques as possible, in a well-structured and computationally efficient manner. You may use the material provided to you in the graphics tutorial series, and are recommended to use this as your starting point. Marks can be gained by showing a good working knowledge of the tutorial material, as well as by moving beyond it by researching and implementing more advanced graphical techniques. You will not lose marks for using textures and models from external sources, but remember to reference and attribute them as appropriate.

The coursework is divided into two parts. The first part is to use the knowledge gained from the tutorial series to develop a program that can display a series of short scenes, each of which containing a different combination of graphical elements, with the program automatically cycling through each scene. The second part is to develop advanced features that demonstrate your full understanding of the graphical techniques outlined during the module, and in your ability to implement new features. Both parts, and all of the graphical features you wish to show off should be present in a single executable project - part of the skill of graphics programming is in implementing a coherent rendering system that can handle multiple different combinations of effects.

Coursework part A – Standard Features:

You must implement the following set of basic features:

- At least 3 small scenes, each containing a different combination of graphical features, such as a different heightmap, lighting setup, animated mesh, a cube mapped reflection, and so on
- At least one of these scenes must show a landscape, with ambient, diffuse and specular lighting applied as appropriate
- At least one of these scenes must show effective use of real time shadows
- At least one of these scenes must show usage of environment mapping
- Pressing the Left and Right arrow keys should cycle forward and backward through the scenes
- Pressing the Pause key should toggle whether the program will automatically cycle through the scenes.
    - Allows us to stop and focus in on a particular effect
- The camera must be able to be freely moved around the scenes with the mouse and keyboard
    - Tie this into above, let's marker move around and see effect up close
- Display the framerate on screen
    - Look at extra tutorial into text rendering
    - Text on screen will help in general
- Appropriate usage of texture data for colouring and adding detail to your shader calculations
    - At least diffuse textures where appropriate
    - Possibly multiple textures if necessary
    - Heightmap might have multiple textures for rocks / sand/ mountains / snow?
    - Textures might contain data for calculations
        - Roughness, normal, specular colour, texture blend factor
- Usage of scene management techniques for efficiency and correctness of scene rendering
    - Scene graph
    - Frustum culling
    - Sort your nodes
    - Enhance the scene node class to make your job easier in the long run
        - Store texture list in nodes
        - Store uniform values in nodes
        - Switch off parts of scene graph not related to current active scene?
        - Or multiple graphs!

Coursework part B - Advanced Features:

You are free to develop your own advanced ideas for enhancing the visuals of the coursework – you are not limited to this example list. The more complex, and well implemented and integrated into the appropriate scenes an effect is, the higher the mark that will be awarded.

- Post processing effects: Blurring, colour correction, lens flare, HDR bloom effects etc. These may be applied to individual scenes or transitions, or used throughout.
    - Deferred rendering – 100s of different lights
    - Post processing – depth of field, bloom, motion blur, tone mapping
- Advanced Lighting: Inclusion of new light calculations such as spot lights, projective lighting, or alternate methods of calculating the contribution of a large number of lights.
    - Lighting part B – additional lighting data, moving towards Physically Based Rendering
- Advanced Shadowing: Use of multiple shadow maps, or omnidirectional shadow maps via cube mapping.
    - Shadow mapping – Cascaded, larger blurring kernel
- Multiple scenes on screen simultaneously using a split screen effect.
- Animated objects (skeletal animation, a progressively evolving plant mesh, or a disintegrating landscape being hit by asteroids / lasers / spaceships).
- Weather Effects such as rain, splashing puddles forming on the landscape, or lightning strikes.
    - Particles – weather system?

Deliverable Items – Ness Submission

- Source code. Clean your solution in visual studio, and then zip your work folder.
- A document containing at least four screenshots with descriptions, a list of any key/mouse presses that perform actions, and a link to a YouTube video of your coursework running.
    - Use OBS Studio to record your project.
- Demonstrate the program running in the lab. The program should run full screen, and you should be able to describe the features present in each scene.

Marks Available (50)

Coursework Park A (25 Marks)

- Implementation of the standard features gains up to 25 marks.
- A “first class” submission consists of these features implemented with well-structured code, running at 60fps in full screen on the lab PC.

Coursework Part B (25 Marks)

- Advanced rendering techniques (the optional extras above) gain up to 25 marks.
- A “first class” submission entails a balanced scene of graphical effects including at least four advanced graphical features (well implemented and integrated with the standard features), running at a minimum of 60fps in full-screen mode. The code must be clearly formatted.

- Learning Outcomes
- Identify appropriate techniques for rendering graphics in real-time.
- Describe graphical representations mathematically.
- Realise which advanced techniques are required to achieve realism.
- Use advanced techniques associated with lighting to create realism in graphical scenarios.
- Balance processing and memory requirements of multiple graphical effects at cinematic frame rate.