**School of Computing and Digital Technology**

**Module Code**: COSE60593

**Module Title**: Low Level Game Programming

**Module Leader**: Dr David White

**Title**: Ray Tracer Optimisation & Report

**Year**: 2018-2019

**Weighting**: 100%

|  |  |
| --- | --- |
| **Learning Outcome** | |
| 1. DEMONSTRATE A SYSTEMATIC UNDERSTANDING OF THE ARCHITECTURE AND CHARACTERISTICS OF A RANGE OF TYPICAL PLATFORMS SUCH AS PCs AND THE PS4 CONSOLE, RELEVANT TO THE PROBLEM OF PERFORMANCE OPTIMISATION OF CODE. | Knowledge & Understanding |
| 2. SPECIFY, JUSTIFY AND APPLY APPROPRIATE DESIGNS AND TECHNIQUES WHICH OPTIMISE/ENHANCE THE PERFORMANCE OF GAMES-RELATED CODE. | Application |
| Problem Solving |
| 3. CRITICALLY EVALUATE THE CHALLENGES AND TECHNIQUES OF DEVELOPING GAMES FOR VARIOUS PLATFORMS. | Analysis |
| Learning |

**Submission**

**Hand-in**: Saturday 12th January 2019

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

You are expected to submit:

1) Using the On-line electronic submission in Blackboard.

a) The deadline will be set at Saturday 12th January 2019

2) A Zip file containing

a) The source code

b) A working executable of the application

c) Reports as detailed below.

You must organise an appointment to present your software and answer questions from your tutor between Monday 14th January 2019 and Friday 18th 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.

Assessment overview

The assessment will evaluate a working version of the ray tracer framework on PC and the ORBIS (PS4) platform. You will be expected to modify and enhance the ray tracer code to implement optimisations and features taught during the lectures and tutorials.

You will be awarded marks for the implementation of each additional feature as described below:

**Memory Management**

Implement C++ memory management techniques as taught during the memory management lectures and tutorials. Override the C++ new and delete functions to track how much memory has been allocated. Override a class object's new and delete functions to track the usage of memory of specific types. Additional marks will be awarded for advanced features as described in the marking scheme.

**Expansion of Framework**

Expand the framework to include timed animation, and an offline editor to manipulate the initial setup of the 3D scene in the ray tracer, view outputted timing data from the ray tracer, export and import ray-tracer setup from XML or JSON files.

**Optimisation of Framework**

Follow the techniques described in the lectures and to measure and improve the rendering and calculation speed of the framework.

**Port to Orbis (PS4) Platform**

Port your C++ code to the Orbis platform, include some PS4 specific features as described in the marking scheme.

**Marking scheme**

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

|  |  |  |
| --- | --- | --- |
| **Section** | **Task** | **Marks** |
| **Memory Management** | | |
|  | Basic use of overridden new and delete | 5 |
| Use of class specific memory allocators | 5 |
|  |  |
| Advanced memory allocation techniques:   * error checking for new and delete * walking the heap * memory pools |  |
| 2 |
| 2 |
| 6 |
| **Total Marks for this section:** | **20** |
| **Expansion of framework** | | |
|  | Animate the spheres on the x, y & z axis continually over the course of a period of time. The animation should be consistent with the frame rate e.g. a frame should be a consistent period of time, so 30 frames would equal one second, for example. | 5 |
|  | Continued expansion of framework:  A separate offline editor for:   * adding and removing spheres, * setting up animations, * changing the colours of spheres, * writing the above setup to a JSON or XML file which will be read in by the PC or PS4 program | 15 |
|  | Write-up of the offline editor - explain its functionality, how it works (in the form of a user manual with screenshots), and how it might be expanded - what sort of additional functionality might it have. You will get graded on both the content and professionalism of the report. This part of the report should be approximately 300 words. | 5 |
|  | **Total Marks for this section:** | **25** |
| **Optimisation of the framework** | | |
|  | Add c++11 chrono and demonstrate how this is used to measure optimisations | 5 |
|  | Show execution speed for framework under 2 optimisation levels (O0...O3) for un-threaded and threaded code. Provide evidence for how this makes a difference to optimisation readings - this should be documented as tabulated data. | 4 |
|  | Provide evidence of further code optimisations   * data structure optimisation * raytracer optimisation * threading optimisation * FileIO optimisation |  |
| 4 |
| 4 |
| 4 |
| 4 |
|  | Write-up of framework optimisation. Explain the differences between baseline speeds and the effect of optimisations. Tables and graphs (there must be data to provide evidence for graphs) will be useful. You will get graded on both the content and professionalism of the report. This part of the report should be approximately 500 words. | 5 |
|  | **Total Marks for this section:** | **30** |
| **Porting to the ORBIS platform** | | |
|  | Demonstrate the porting of your framework-specific features to ORBIS | 10 |
|  | Add libFiber your PS4 framework and demonstrate it in use. | 10 |
|  | Write-up on porting to ORBIS platform. Discuss the issues you had, the changes you needed to make to your code. How did the effect of optimisations differ between the ORBIS and PC platforms? This part of the report should be approximately 300 words. | 5 |
|  | **Total Marks for this section:** | **25** |