CITY
UNIVERSITY OF LONDON
EST 1894

IN3026: Advanced Games Technology

BSc/MSci Computer Science with Games Technology

Lecturer: Dr Chris Child
Email: c.child@city.ac.uk

Resit Coursework project: *Casual Game in C++ and OpenGL*

Due: 9th August 2019, 5pm

Synopsis

The aim of this coursework is to give you the opportunity to develop your games programming experience by using industry-proven languages and toolkits. You will be developing a single level of a casual game in C++ using Visual Studio and the provided game template on Moodle. The game will use OpenGL for real-time 3D graphics rendering.

This coursework is a single piece of individual work. There is no further exam on this module.

Submission details

As with all modules, the deadline is hard, and extensions may only be requested via the standard Extenuating Circumstances procedure.

A 3D platformer game

Taking on the roles of designer, developer, and tester, you will create a single unique level of a 3D casual game. The core concept will be a game containing:

- A 3D platformer game, **with at least three moving platforms**.
- Gameplay elements (power-ups, obstacles, etc.) that give the player a challenge and make your game interesting to play.
- At least one camera technique.

CITY
UNIVERSITY OF LONDON
EST 1894

The theme of this game is open to interpretation, but the game should be non-trivial to play. The game should be aimed at a primary market of players aged 12 and above, with non-complex controls.

Task

Your task is to create a 3D game level which will use a full heads-up display that includes a camera technique, scoring system, and moving 3D platforms. The world should contain objects for the player to pick up (collision detection must be implemented using standard techniques) as well as objects to avoid (such as enemies firing at your location, mines / obstacles set at coordinates, etc.).

You must demonstrate:

- At least one camera technique beyond what is provided in the template code.
- Points-based objectives to reach, acquired through an appropriate mechanism for your game (e.g., collecting different items, defeating enemies, reaching timed checkpoints).
- Use of game audio, physics, and AI.

Technology

You will use C++ for the purposes of this project, as part of Visual Studio. You are expected to make use of external libraries as needed. The computer graphics must be implemented in OpenGL.

You may NOT use any existing game engines or other templates other than those supplied to you, unless agreed upon by the Module Leader (Dr Chris Child) with a response **in writing** (which you must include in your project .zip). However, use of any additional middleware libraries and APIs (Bullet, PhysX, Havok, ODE, OpenAL, Asset Loaders, Modlib, Freetype, Boost, etc.) are welcome but must be noted in the submitted documentation. The additions must also be sufficiently packaged with both the source code (so it compiles) and the final deployed executables (so they run) on any lab machine.

You may wish to begin the project by producing a paper sketch-up identifying all the objects in the game, including actions the objects can perform in response to game events.

CITY
UNIVERSITY OF LONDON
EST 1894

Marking

The project will be built in three main parts, and include a report.

Part 1: Basic game modelling 25%

- Game level intro screen with a listing of keyboard/mouse controls. (4%)
- Write the code to produce an octahedron built using appropriate basic OpenGL polygonal shape primitives (GL_POLYGON, GL_QUADS, GL_QUAD_STRIP, GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN). For this task, GLU/GLUT shapes or meshes loaded into the game **do not count**. Texture-map the octahedron shape using an appropriate texture and texture coordinates. Also, use valid vertex normals to achieve correct lighting. Use appropriate rotation, translation, and scaling to place the octahedral objects in the scene. Also, be sure to **change the skybox and terrain textures**. (12%)

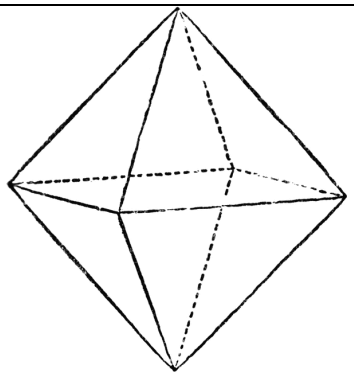

Figure 1. An octahedron (eight-sided polyhedral surface)

- Audio. At least three different sounds synchronised to game events, and a change of the background music/audio. (4%)
- Heads up display (HUD) while playing. At a minimum, this should provide a score, but may also include other gameplay elements including health, lives, time remaining, etc. The HUD can be rendered as text or text combined with graphics. (5%)

CITY
UNIVERSITY OF LONDON
EST 1894

Part 2: Camera, meshes, lighting, and FX 25%

- A camera motion technique. Examples include constraining the camera to be "on rails," behind the player, or rotatable around the player. Only one technique is required, but this must involve a significant change to template code. (6%)
- Mesh-based objects: textured, lit, and transformed (rotated, translated, scaled) into scene. Four different meshes required. Repeating the same mesh four times **does not count.** The meshes must be different but you are welcome to repeat them as much as you like. GLU/GLUT shapes do not count. (7%)
- Lighting. At least three lights required in the scene. You may want to program different light types (point light, spotlight, directional) and have one of them moving in your scene. (6%)
- Special effects that demonstrate creative uses or techniques appropriate for gameplay, such as: camera jitter; cross-fading; blending; fog, transparency (alpha); explosions; particle effects (three techniques required). (6%)

Part 3: Physics, AI, and gameplay 25%

- Use of game physics **for the player**, such as jumping, gravity, velocity, momentum, resistance, acceleration, drag, friction, bouncing that leads to a creative look and feel for the level (four techniques required). When the player is on a moving platform, the player should naturally move with the platform. (7%)
- Non-player characters (NPCs) and artificial intelligence (AI). Include at least five NPCs in the game. Program them so that they have some intelligence, based on at least one of the following techniques: finite state machines (at least four states), steering behaviours, pathfinding, decision trees (at least two levels deep), flocking. NPCs should influence the gameplay (enemies causing obstruction / damage, or agents helping the player achieve objectives). (8%)
- Gameplay elements in line with the game theme. Implement a set of at least three of the following: (10%)

1. Power-ups. Power-ups must give some advantage to the player that affects gameplay (examples: special powers, increased health, ammo, additional time, less damage when hit, inability to get hurt, switches to activate different parts of the level, etc.).

2. Combos. Combos must provide a way for the player to score additional points through gameplay (examples: defeating an enemy with a special attack that scores more points than a standard attack, defeating multiple enemies in rapid succession to score more points than defeating each more slowly, etc.)

3. Timers. A timer might count down the remaining time left in the game, or count how long the player has survived, etc.

- You are welcome to implement any three of the above, including three of the same type (e.g., three power-ups and no combos or timers for example) -- depending on the style and genre of your game.

Part 4: Project report and source code 25%

You are to provide a 7 to 15-page report documenting your project. Do not exceed 15 pages. The report should provide:

- An overview, asset listing, additional libraries (5%)
    - An overview of the project, including the game's title, description of the theme, genre of game implemented, and description of basic gameplay objectives and mechanics.
    - Complete listing of all assets (meshes, images, textures, sounds, etc.) and external libraries used to make the game, including
        - If downloaded: The URL where downloaded, date of download, AND license for use.

CITY
UNIVERSITY OF LONDON
EST 1894

        - If produced: Software used and a short description of how the asset was created.
    - Complete listing of any additional libraries added to the project.
- For each of Parts 1, 2, and 3 above, (10%)
    - A description of features implemented to satisfy requirements. If you chose to skip implementation or partially implement a requirement, state this.
    - For implemented features, design, implementation, and results
        - Use of UML (or alternative) diagrams where appropriate (e.g. use cases, class diagrams, state machine diagrams).
        - Pseudo-code or small sections of source code (code snippets) with commentary describing important algorithms developed.
        - Research conducted to implement your game.
        - Screenshots showing the game and game elements (no more than two pages of screenshots).
- Discussion section reflecting on the project. Consider the strengths and weaknesses of the game level implemented and what you have accomplished in the time provided. Also discuss what would be required to expand the project into complete game. (5%)
- Source code to be commented and follow a logical design, organisation, and coding style (i.e., use of classes). (5%)

You will submit this documentation electronically as a Word or PDF document in the zip file (in a folder named documentation).

Important note: only assets (images, meshes and audio) that are free for individual/educational use may be used in your game (this includes freeware, open-source, GNU GPL, BSD license).

Deadline and Deliverables

CITY
UNIVERSITY OF LONDON
EST 1894

The main deliverable is to be submitted via Moodle as a .zip file. This should contain three folders:

1. Documentation folder including report in Word or PDF format.
2. Folder of archived source code. This must be at a stage where a simple ‘compile and run' (F5) will start your application. Failure to have a compile and run working project will result in a **FAIL**.
3. Folder for binaries for publication, which should run with all required assets from a folder but no source or debug files. Remember, **the executable must be in a runnable state - i.e. not requiring changes to the directory structure to point to resources/assets**.

- Your source code and solution files must be submitted in a form such that a simple “build and execute” will build without errors on the university machines. Before submission you should:
    - Perform a clean in both Debug and Release modes.
    - Delete the .vs file from the root directory.

    WARNING: Failure to clean your project in this way may result in it being too large to submit via Moodle.
- Any additional libraries used, must have their license open for educational usage, and deployed with the working system without re-configuration.
- Any mesh, image and texture assets must be freely available, with license cited in your documentation.
- ***Important note***: only assets (images, meshes and audio) that are free for individual/educational use may be used in your game (this includes freeware, open-source, GNU, GPL, BSD license).

Support and tips

CITY
UNIVERSITY OF LONDON
EST 1894

There are a large number of demos and materials available on Moodle to get you started. You are welcome to use the existing template provided, or build your own as long as the rendering is done using OpenGL.

Remember, this is an individual project. You are encouraged to help each other research techniques. But your final submission must be entirely your own work.

Students who do well on this task are invariably those that have good time management. **Dedicate sufficient time to this module each week so that you make steady progress**.

Do not spend a long time on the creative aspect of the project. This is a programming module, and you are not being marked on game design or artistic merit.

This coursework should be used as a syllabus guide of things that you need to cover as a game developer.

Google any term you are unfamiliar with, including the keywords “OpenGL tutorial”. Put aside time each week to cover material and please practice coding frequently. Be sure to complete each lab as the labs have been designed to reinforce concepts in class and help you with the coursework. As a class, you are a games development studio for the duration of this module.

Ben Humphrey’s industry standard tutorials from GameTutorials.com cover the ins and outs of much of the syllabus using very well documented examples in OpenGL. There are selected examples which he has kindly given us permission to demonstrate at City University. These are available on Moodle under “Useful links”.

Other great sources of inspiration and help include the NeHe tutorials and GameDev.net, both of which have links from the class Moodle page.