CITY
UNIVERSITY OF LONDON
EST 1894

INM710: Advanced Games Technology

MSc Computer Science with Games Technology

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

Re-sit Coursework Project: *Casual Game in C++ and OpenGL*

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 game template provided on Moodle. The game will use OpenGL for real-time 3D graphics rendering.

This coursework is an individual work, worth **100% of the final course module mark**. There is no further exam on this module.

Submission Details

You must submit your project via Moodle. As with all modules, the deadline is hard, and extensions may only be requested via the extenuating circumstances procedure.

A 3D Platformer Game

Taking on the roles of designer, developer and tester, you will create a single 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.

The theme of this game is open to interpretation, but the game should provide different degrees of challenge in the level. 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 include a camera technique, scoring system, full heads-up display, 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* the one provided in the template code.
- Points-based objectives to reach, acquired through an appropriate mechanism for your game (e.g., collecting 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 can obtain this from Microsoft or use university computers with Visual Studio already installed). You are expected to make use of external libraries as needed. 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 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 (such that it will compile) and the final deployed executables (such that it will run on any lab machine).

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

Part 1: Basic Game Modelling (25%)

- Final game level intro-screen with a listing of keyboard/mouse controls. (4%)
- Inclusion of at least four primitive-based game object shapes, built using basic OpenGL primitives. The shape from Milestone 1 counts for this task, but at least three more are required. As before, meshes loaded into the game are not accepted. Repeating the same shape three times does not count. The shapes must be different, but you are welcome to repeat them as often as you like. Texture-map each shape using an appropriate texture and texture coordinates. Use valid vertex normals to achieve correct lighting. Use appropriate rotation, translation, and scaling to place the objects in the scene. (9%)
- Change the skybox and terrain textures. Ensure lighting is off for the skybox and there are no seams. Terrain textures should not be stretched. (3%)
- Audio. At least three different sounds synchronised to game events, and a change of the background music/audio. (5%)
- Heads up display (HUD) while playing. As a minimum, this should provide a score, but may also include other gameplay elements including health, lives, time remaining, etc. depending on the nature of your game. The HUD can be rendered as text or text combined with graphics. (4%)

Part 2: Camera, Meshes, Lighting, and FX (25%)

- Final 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 the 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 often as you like. Meshes used in Milestone 1 count towards this requirement. (7%)
- Lighting. At least three lights are required in the scene. You may want to program different light types (point light, spotlight, and 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; crossfading; blending; fog, transparency (alpha); explosions; and particle effects (three techniques required). (6%)

Part 3: Physics, AI, and Gameplay (25%)

- Use of game physics, such as gravity, velocity, momentum, resistance, acceleration, drag, friction and bounce that lead to a creative look and feel for the level (four techniques required). (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 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 two of the same type (e.g., two power-ups and no combos or timers for example) depending on the style and genre of your game.

CITY
UNIVERSITY OF LONDON
EST 1894

Part 4: Project Report and Source Code (25%)

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

- An overview, including asset listing and 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.
        - If produced, the 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 a complete game. (5%)
- Source code to be commented and follow a logical design, organisation, and coding style (i.e., use of classes). (5%)

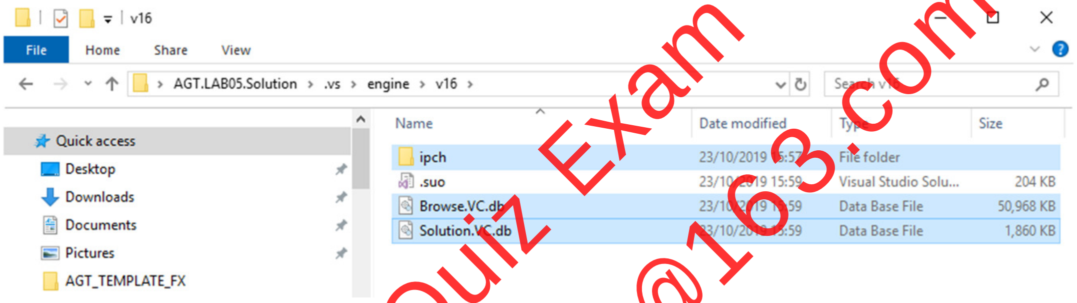
Deliverables

The coursework is to be submitted via Moodle as a .zip file. This file should contain two 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**.

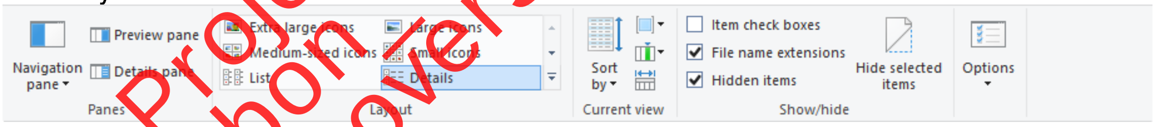
- Your source code and solution files must be submitted in a form such that a simple "build and execute" will build without errors on the university machines. Before submission you should:
    - Perform a clean in both Debug and Release modes.
    - Delete the contents of .vs\engine\v16 directory (or .vs\engine\v17 directory if you are using VS 2022), apart from the .sou file (required to tell VS to keep the additional projects unloaded). See below:



Name	Date modified	Type	Size
ipch	23/10/2019 15:5[illegible]	File folder	
.suo	23/10/2019 15:59	Visual Studio Solu...	204 KB
Browse.VC.db	23/10/2019 15:59	Data Base File	50,968 KB
Solution.VC.db	23/10/2019 15:59	Data Base File	1,860 KB

WARNING: Failure to clean your project in this way may result in it being too large to submit via Moodle.

You may need to enable hidden items under the view menu in explorer to see the .vs directory:



- Any additional libraries used, must have their license open for educational usage, and deployed with the working system without re-configuration.
- 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).

Support and tips

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 uses OpenGL.

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

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. You should be continuously reading up on related topics throughout the term.

Google any term you are unfamiliar with (e.g., 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.

Coding & Referencing

This is, in large part, a coding assignment. If you use code (or other materials) written by someone else, you should cite that code (or other material) in Harvard format. Your code itself should reflect sources in your comments. If you do not cite work appropriately you will have committed academic misconduct. Making superficial changes to the code does not make it yours. You are also expected to make a coding contribution, so if you use a large amount of code written by someone else, and cite it appropriately, your contribution will be low and your work marked accordingly.

Academic Misconduct/Plagiarism

If you copy the work of others (either that of another team or of a third party), with or without their permission, you will score no marks and further disciplinary action will be taken against you. The same applies if you allow others to copy your work. This is a group responsibility.

Sample code (labs) available on Moodle as term progresses. Study and use of sample code from the internet is *encouraged but* must be referenced if you include it in your work.