# COMP3016 IMMERSIVE GAME TECHNOLOGIES

**20 CREDIT MODULE/100% COURSEWORK SUBMISSION**

**MODULE LEADER: Ji-Jian Chin**
**MODULE TUTOR: Sebastian Outram**

**MODULE AIMS:**

- **To gain and apply programming concepts and techniques that are used in demanding real-world industrial settings, e.g. programming in an unmanaged environment.**
- **To gain experience in software design and engineering working from concept to finished prototype.**
- **To gain experience with different programming languages and environments.**
- **To establish the mathematical concepts required for the development and programming of high-performance real-time graphics, such as vector and matrix maths.**
- **To provide experience using a graphical API such as OPENGL.**

**ASSESSED LEARNING OUTCOMES:**

1. **Apply core programming principles (including mathematical concepts within a high-performance real-time environment such as graphical programming/game-engine programming.**
2. **Design, conceptualise and implement a working prototype with clearly defined features.**
3. **Articulate method of approach and rational for solution.**

## OVERVIEW

This module introduces concepts & programming techniques for working with un-managed code, i.e., using graphical APIs like OpenGL through C++. It provides industry relevant skills for domains such as high-performance real-time graphics or closer-to-the-metal programming. The module uses a bottom-up approach; it is not about the use of game engines and high-level prototyping tools. Students will gain highly transferrable software engineering skills.

Lectures, Seminars and Workshops are integrated into the module to introduce concepts and deepen the understanding of topics as well as to guide the project work.

Coursework elements include an Individual research presentation to inform the development of a prototype and individual projects with supporting documentation.

**100% Coursework Comprising Two Individual Elements**

**Assignment A: Text-based Game 30%** Working individually on a text-based game using OOP design paradigms and programming skills in C++.
**Assignment B: Individual Software Project 70%** Working individually on a project, find, defend & agree a topic with staff. Research and produce a short industry standard portfolio piece (polished GitHub page) and a video presenting your product/research.

**MODULE DELIVERY**

| | |
|---|---|
| **Delivery format**: | Weekly 2hr lecture Thursday from 1pm – 3 pm |
| | Weekly 2hr lab Thursday 11am - 1 pm |
| **Delivery staff**: | Ji-Jian Chin – ji-jian.chin@plymouth.ac.uk |
| | Sebastian Outram – sebastian@outram.org |
| **Duration:** | 12 weeks Semester 1 |

**Assessment Offences:**
For this assignment you may be using information from differing sources:

- Books, journal articles
- Course/module materials
- Websites
- Existing Open-Source Projects

It is ***very important*** for you to note that this assignment is *an **individual effort.** It **should make the contributions from the student** and the use of external resources, or an initial start base clear.*

Thus, do not simply copy existing sources, i.e., other students work, interspersed with a few lines of code or words of your own. This is paraphrasing, and it is not encouraged, it is not likely to get you a good mark and, in some cases, it could be seen as plagiarism. In a similar vein, do not simply copy material from elsewhere without citing it properly. For more **information** on how to write texts, reference source material and plagiarism in general, see:
https://www.plymouth.ac.uk/student-life/your-studies/essential-information/regulations/plagiarism

If you have any doubt as to what constitutes '***an individual effort and in your own words'*** then either see your student handbook or see me.

**Coursework Part C1 – Console-based Game using C++ 30%**
**DESCRIPTION**

This is an individual assignment to develop a game in C++.

The ultimate aim is for you to demonstrate that you can understand writing unmanaged code in the object-oriented paradigm, interacting with different data types and the underlying code for moving things to and from memory, thus handling some of the complex aspects needed for writing game engine code, at the same time having a hand at some creativity to develop your own flavour of a text-based game in the terminal console.

For this assessment, you should use *modern C++* without making use of C primitives and procedures as much as possible (e.g. using string over char*).

Your source code should compile without errors with a working executable file that runs on its own, accompanied by simple design document detailing its designs.

Other than that, the other requirements should be that it is fun and accounts for quality (e.g. original mechanics, does not crash with improper inputs).

To complement the development, you should write a brief introduction and description which details the setup, usage and how the code fits together to be used in other cases. You should use proper UML diagrams to showcase your design. For the video you may use this to walk us through your game. As part of the video, you need to show how the software runs and compiles.

**DLE DELIVERABLES:**
Create a single ".zip" archive (max 150MB) containing a txt link to the public GitHub readme/report, the executable and required libraries to run it. Detail how to run the executable (put that in the write-up below). Don't forget to include any other resources required; meshes, sounds etc. Submit your zip file to the DLE electronic submission system. And don't forget to remove any unneeded files and folders. **Compress any additional resources or assets to stay within the space limit.**

Zipfile checklist:

- An executable version of the game which does not rely on VS. Include any resources such as extra assets that your project needs.
- Full source code that can compile (teaching staff will test). Check that all dependencies are included and properly referenced.
- Link to a public git that contains source code and a report in markdown format (*.md) describing:
  - **A link to the Video (Maximum 10 minutes, 1080p, unlisted YouTube). Your video should be a verbal articulation and walkthrough of the report points below, including a playthrough demonstration. Remember to include your face on camera for authentication.**
  - Dependencies used.
  - Game programming patterns that you used.
  - Game mechanics and how they are coded.
  - UML design diagram if any
  - Sample screens.
  - Exception handling and test cases
  - Further details that help us to understand how your prototype works.
  - A (brief) evaluation of what you think you have achieved, and what (if anything) you would do differently, knowing what you now know. Feel free to blow your trumpet!

**Deadlines:**

| Part | Description | Deadline | Percentage |
|---|---|---|---|
| C1 | Standalone CPP Game: Demonstrate expertise in working with unmanaged code: C++ | | 30% |
| | Console-based game: Implement a game in C++ and upload to your git repository, then share the link in the readme. Your game needs to adhere to OOP concepts and additionally be able to load file content during execution. Do enough documented testing that it can be built and compile, and that it doesn't crash upon unexpected input. Lastly, *add in as many features as you can to make it story rich and/or fun!* | | |
| | Demo Video: Do a code walkthrough, including functional playthrough in a short 10min video. Highlight uniqueness of your game designs. | 4th November | |

**Marking Rubric**

**To pass (0-40)**

- Written in C++ (with classes) and must compile.
- Report, git and video included.
- Program must be a playable game.
- Must be original work (cite your references and be articulate original design aspects).

**Basics**

1) **Fun factor (10%)**
   - A chore to play (0-3%)
   - Captures attention short term (4-7%)
   - Engrossing to immersive (8-10%)
2) **Code (10%)**
   - Simple and messy with minimal OOP design, testing and inline documentation (0-4%)
   - OOP design evident with sensical design choices, moderately complex algorithms, simple testing evidenced and clear articulation (5-8%)
   - Adheres to solid well-tested OOP programming principles that are well articulated (9-10%)
3) **Game mechanics (10%)**
   - Basic: simple, unoriginal mechanics with varying success affecting gameplay (0-4%)
   - Good: some innovation on existing game mechanics (5-8%)
   - Excellent: original and fun mechanics (9-10%)

**Aesthetics (10%)**

- Overall evaluation on sound, visual and atmospherics.
-

**Advanced – half the marks will come from articulation in the report and video.**

1) **Game programming patterns (5%)** – maximum 2
2) **Unique selling points (5%)** – maximum 2
3) **Implementation of Current Research (10%)**
   – check with module leader if making this claim on what features are permissible. Can only claim on features authorised by module lead.

**Penalties will be incurred on all criteria must be not properly articulated in the report and video. more comprehensive rubrics will be uploaded on scoring system.**

## Coursework Part C2 – Individual Topic: The Interactive Software Prototype 70%

**DESCRIPTION**

This is an individual piece of work. *"Create something which does something in OpenGL".*
The actual project is negotiated based on the submitted proposal but has two strict additional requirements:

- **Key requirement A** for this part is to **develop your interactive software in C++ incorporating OpenGL.**
- **Key requirement B** for the assessment is to have a **meaningful original contribution** of **at least two features** to your project**.**

You must create a working prototype using **OpenGL4.X** which allows for real-time user interaction.

For the coursework you are required to use an unmanaged language (C++) and are **not allowed to use existing game engines such as Unity3D, UE, Godot or Ogre.**

C2 has the following criteria:

- Start with a proposal for **formative feedback** by the module leader; The main intention is to lock down design specifications on what you intend to do before you get too far into the production process. The proposal should also contain external libraries you intend to use and must be approved for it. *(Not having accepted your project will cap the marks. A similar cap will be incurred for using libraries without approval.)*
- Your project must use **OpenGL4** and only libraries featured in the labs (which means you cannot use the fixed function pipeline, so the software requires at least an external fragment and vertex shader). For context window only **GLFW or SDL** are allowed. For wrangling only **GLAD or GLEW** are permitted. For models, meshes and textures, use **ASSIMP** and for sound, use **Irrklang**. Physics engine should be limited to **PhysX,** if any. Any other libraries you intend to use must be mentioned in the proposal and receive prior approval.
- You may use other higher-level tools for assistance – for instance you may use "Photoshop" to produce the textures files, and 3D modelling tools such as "3D Max", "Blender" to produce meshes, and ChatGPT for sample templates of code (but be careful do not lift the code in its entirety!). And you can also use such resources from any legal source you like, but you must mention it on your GitHub project page. The results of these resources, however, are not the primary focus and should only be used to augment your project.
- Adhere to good software engineering and OOP programming principles – tidy, decoupled, efficient, properly tested code with documentation. Optimisation features that are well-articulated are a plus.

What C2 should look like:

- A scene with an environment (3D space) and at least a subject (usually a model/mesh).
- Animation on lights and/or vertices of the mesh.
- Keyboard and mouse interaction/movements.
- Elements of gameplay preferred but optional.

**DELIVERABLES**

Create a single ".zip" archive (max 150MB) containing a txt link to the public GitHub readme/report, the executable and required libraries to run it. Detail how to run the executable (put that in the write-up below). Don't forget to include any other resources required; meshes, sounds etc. Submit your zip file to the DLE electronic submission system. And don't forget to remove any unneeded files and folders. **Compress any additional resources or assets to stay within the space limit.**

Zipfile checklist:

- An executable version of the prototype which does not rely on VS. Include any resources such as extra assets that your project needs.
- Full source code that can compile (teaching staff will test). Check that all dependencies are included and properly referenced.
- Link to a public git that contains source code and a write-up in markdown format (*.md) describing:
  - **A link to the Video (Maximum 10 minutes, 1080p, unlisted YouTube). Your video should be a verbal articulation and walkthrough of the report points below, including a live demonstration. Remember to include your face on camera for authentication.**
  - Dependencies used.
  - Game programming patterns that you used.
  - Game mechanics and how they are coded.
  - Are there any software engineering issues, such as the trade-off between performance and good practice?
  - UML design diagram if any
  - Sample screens.
  - Exception handling and test cases
  - Further details that help us to understand how your prototype works.
  - A (brief) evaluation of what you think you have achieved, and what (if anything) you would do differently, knowing what you now know. Feel free to blow your trumpet!

| **Part** | **Description** | **Deadline** | **Percentage** |
|---|---|---|---|
| C2 | Individual project: Implement an interactive software prototype that builds upon the content from the lectures and demonstrates the required skills for game engine development. ALO2.<br>Document and defend your design decisions through your portfolio github page and in a 15min pitch and walkthrough video. ALO3. | | 70% |
| | Interactive Prototype Pitch: Name your project, describe the concept and how you want to achieve it in the given time. | Thursday, 2 November | |
| | Individual Project: *Upload your Project to the DLE and have the final version on the classroom including the report and all required files.* | Monday, 7th of January | |
| | Video Release: Upload and share the walkthrough and pitch video to YouTube or OneDrive. *(The link should be on the GitHub page as well)* | Monday, 7th of January | |
| | Demo Session: Walk us through your prototype and answer questions regarding its programming; Get Verbal Feedback regarding your performance | TBC between 8th and 16th January | |

**Marking Rubric:**

| | | |
|---|---|---|
| **To pass** | Use C++ | To pass, you need to have a scene written in OpenGL C++ with vertex and fragment shaders, with a quad and signature displayed. Code must be compilable without errors. Your submission must be accompanied by video and git link which matches your submission. You must cite all external resources. |
| | Have report and git with readme | |
| | Submit 10 min video on showcase | |
| | Code compiles with GL window and basic polygon | |
| | Signature on scene | |
| | No plagiarism (must attend viva for this) | |
| **40-70** | **All features claimed must be covered in Git and video.** | |
| 5 | MVP implemented | 2.5 marks for CPP implementation. 5 marks for vertex shader implementation. |
| 5 | Textures | 2.5 marks for single texture. 5 marks for mixed. |
| 5 | 3D Polygons with scene animation (triangles and quads not included) | 2.5 marks if static scene with 2 or more objects, 5 marks with timed animations. |
| 5 | Keyboard and mouse movement | 2.5 marks for keyboard 2.5 marks for mouse. Mouse movement must be fluid and intuitive. |
| 5 | Load model(s) with textures | 2.5 marks for single model. 5 marks for more than 1. Must be multiple formats. |
| 5 | Procedural content generation (PCG) | 2.5 marks with height variation. Full 5 marks for more than 2 biomes. Flat planes not considered. |
| | | |
| Cap at 30% | | |
| **Aesthetics** | 10 | Aesthetics scale of 1-10. Nominal mark is at 5. |
| Advanced | **All features claimed must be cross examined during viva. These points can only be claimed if the 40-70 points are maxed.** | |
| 5 | Mixed textures | 2.5 marks for advanced mixing (normal map, noise techniques), 5 marks if able to combine more techniques in same scene. |
| 5 | Model animation in assimp | 2.5 marks for basic animation (single torus). 5 marks for more advanced movements. |
| 5 | Interaction/adjust model params. | 2.5 marks for for keypress adjustments. 5 marks for scaling features with GUI. |
| 5 | BlinnPhong/PBR lighting | 2.5 marks for static light source. 5 marks for dynamic moving lights. |
| 5 | Audio | 2.5 marks for background music/ambient noise. 5 marks for interactivite audio playback |
| **Cap at 10%** | **So choose maximum 2. Other features to be claimed depends on pitch approval, if any.** | |
| Last 10% | Research-related implementation (cite a source within 5 years) | |
| **Note you cant reuse for COMP3015, so learn the skill but don't expect to resubmit same coursework.** | | |

| Penalties | | |
|---|---|---|
| | | |
| 10 | Using other libraries without authorisation (Only those covered in the module and those on learnopengl.com are permitted) | |
| 10 | Final outcome differs too much from pitch | |
| 5 | No OOP paradigm in main CPP | |
| 5 | Insufficient internal documentation | |
| 5 | Insufficient error handling | |
| **Penalties will be capped at 40%. You will not fail on penalties.** | | |

**Please refer to all the lecture content & further study resources on the DLE. A more comprehensive rubrics will be uploaded on scoring system.**

**Grades:**
A total of 40% is needed for passing the module. When re-taking the assessment, no score from a previous assessment is taken forward. Thus, the entire coursework needs to be re-done.