COMP3016
Immersive Game Technologies

20 CREDIT MODULE

ASSESSMENT: 100% Coursework **W1: 30% Coursework 1**
W2: 70% Coursework 2

MODULE LEADER: Dr Ji-Jian Chin
Dr Lauren Ansell

MODULE AIMS

- To gain and apply programming concepts and techniques that are used in demanding real-world industrial settings, eg. Programming in an unmanaged environment.
- To gain experience in software design and engineering working from concept to finished prototype.
- To gain experience from 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 (ALO):

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

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: A 2D 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
	Lauren Ansell – lauren.adams@plymouth.ac.uk
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*** *clear and include any use of external resources.*

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

The sections that follow will detail the assessment tasks that are to be undertaken. The submission and expected feedback dates are presented in Table 1. All assessments are to be submitted electronically via the respective DLE module pages before the stated deadlines.

	Submission Deadline	Feedback
Report Outline (30%)	**3th November 2025 (15:00)**	**2nd December 2025 (15:00)**
Report (70%)	**9th January 2026 (15:00)**	**4th February 2026 (15:00)**

Table 1: Assessment Deadlines

All assessments will be introduced in class to provide further clarity over what is expected and how you can access support and formative feedback prior to submission. **Whilst the assessment information is provided at the start of the module, it is not necessarily expected you will start this immediately** – as you will often not have sufficient understanding of the topic. The module leader will provide guidance in this respect.

Moderation
This assignment brief has been moderated in line with the university policy.

Moderator Name	Moderation Date
Rory Hopcraft	

Useful Assignment Information

Please see below some useful information regarding submissions for your modules.

Good Coding Practices

Like all things, no code is perfect. Just because your code compiles and runs does not mean it is perfect, there is always room for improvement. No code will achieve 100% of the available marks. Where code submission is required by a module, it will be assessed against the following criteria:

1) **Functionality**
    a. Does your code meet all the specified requirements of the assignment?
    b. Does your code behave correctly across typical and edge-case scenarios?
    c. Does you code have appropriate error handling that does not leading to it crashing or undefined behaviour?
2) **Efficiency**
    a. Is your code optimised in terms of performance and resource use?
    b. Does you code handle functions, variables, data calls efficiently?
    c. Is there any unnecessary repetition, complexity or processing reducing efficiency within your code?
3) **Readability**
    a. Is your code logically structured and easy to read?
    b. Have you maintained standard formatting conventions like indentations, spacing and naming, within your code?
    c. Are variables, functions, and classes clearly named and purposeful?
4) **Documentation**
    a. Are there helpful comments explaining non-obvious parts of your code?
    b. Do your comments document your process, development or rationale clearly?
    c. Could someone unfamiliar with the code understand the approach you have taken?

Use of Generative AI for Creating Code

Each assignment element for each module will have a clear indication of the permitted level of use of AI (solo, assisted or partnered) including the generation of code. Please ensure that you **read and understand the permitted use**. If you are unsure of a particular use, please reach out to the Module Leader and ask.

We strongly encourage you to read and explore beyond the core content delivered within the modules. However, this must be done correctly following academic process. Wherever code is drawn from an outside source (e.g. you have not written it yourself), regardless of whether this is AI generated or from an online repository (GitHub, Stack Overflow etc) **you must reference the original source**. This can be done in your documentation as comments or part of your write ups. Students found to be utilising code without referencing could face an academic offence. **If in doubt speak to your Module Leader.**

Versioning

A critical part of a development (software or academic) is versioning. Keeping a number of iterations over the course of development ensures you always have backups to fall back on. It also provides clear demonstration of the development process you implemented.

Using online/cloud-based storage solutions and repositories provide additional peace of mind, alongside being industry practice. The university provide OneDrive to all students which offers inbuilt version history for Microsoft products. GitHub is the university recommended repository for versioning code, which includes great integration with a number of IDEs.

Submitting Code

It is vital that you confirm that all code that you submit is in the **correct format** and **compiles correctly** or **runs without error**. If you clean up your code before submission, please ensure that all dependencies (libraries, functions and variables) are included so that the marker can compile your code.

We can only mark what has been submitted. If the code does not run correctly, it is not our responsibility to spend time error handling to award you marks.

It is also important to **show your working**. Tidying up your code is a critical part of coding practices, but if you remove the workings that provides you with the required output, we cannot see how you got there.

In a worst-case scenario, if you remove a key part of your working, and on compiling your code it gives an error or different result, we have no way of confirming what went wrong, or indeed if you fabricated those outputs.

Please note, just because a piece of code has given the wrong output, it does not mean it should be deleted. Seeing these attempts with comments documenting your attempt allows us to understand where you may have made an error and provide feedback that addresses this. In some cases, you could also be awarded marks for the attempt.

Coursework 1

Task:

This is an individual assignment to develop a 2D game in C++.

Implement a 2D 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, a*dd 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. Talk about how you used AI in the development process.

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*).

Submission Requirements:

- 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.
- Video upload featuring your game. **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.**
- Link to a public git that contains source code and a report in markdown format (*.md) describing:
    - **A youtube link to the Video (Maximum 10 minutes, 1080p, unlisted YouTube). This can be the same video. The above is for the DLE requirement. This YouTube version is for your own publicity/showcase.**
    - Gameplay description.
    - Dependencies used.
    - Use of AI description.
    - 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!

Acceptable level of generative AI tool use in this assessment element

The acceptable level of GenAI use for this element is detailed with the allowed uses listed below. This is split into three categories (**Solo Work** – work must be your own with no AI support, **Assisted Work** – some uses of AI tools allowed, **Partnered Work** – AI tools integral part of the work). If you have any questions, please contact the module leader.

Solo Work	You must not use generative AI tools.	☐
Assisted Work	You are permitted to use generative AI tools in an assistive role.	☐
Partnered Work	Generative AI tool use is required as an integral part of the assessment, but transparency is required.	☒

Table 2: Acceptable Level of generative AI use

In line with the indicated acceptable level of AI use above, the following uses are acceptable:

- Creating game assets (models, sprites, font and title, art, music and sound effects).
- Code assistant.
- Programming testing.
- Readme report crafting.

Any use of AI in your work must be declared within your documentation. You **must also include a signed Generative AI Declaration as an appendix to your submission**. The declaration form can be found on DLE (or Here on the Programme Page). This form will not be included in any word count associated with this assignment.

Assessment Criteria:
The following marking scheme will be used for this part of the assessment:

- Passing requirements (40%)
    - Must be an original work.
    - Can compile and run.
    - Report, git and video included.
    - Program must be a playable game.
    - Use of AI declared.
- Fun factor (10%)
- Code quality and testing (10%)
- Originality of game mechanics (10%)
- Aesthetics (10%)
- Advanced Features (10%)

An illustration of the feedback template is presented in Table 2.

Criteria	Fail (<40%)	3rd (40%–49%)	2:2 (50%–59%)	2:1 (60%–69%)	1st (70%–100%)	Weight
Passing Requirements (Can compile and run, report/git/video included, playable, original, AI declared)	Missing any one required component.	Basic functionality, minimal report, AI mention vague.	Functional game, report covers most points, AI use described.	Well-structured report, game runs smoothly, AI use explained clearly.	All components excellent, AI use well-integrated and justified.	**40%**
Fun Factor (Rated by anonymous play testers)	Not fun or broken.	Mildly engaging, lacks polish.	Some fun elements, but inconsistent.	Generally fun, some bugs.	Immersive and enjoyable, minimal complaints.	**10%**
Code Quality & Testing (Includes sprint progress and AI-assisted development)	Poor structure, no testing, no AI use.	Basic structure, minimal testing, AI used superficially.	Good structure, some testing, AI used for snippets.	Clean code, documented testing, AI used for debugging or design.	Excellent code, thorough testing, AI used extensively and effectively.	**10%**
Originality of Game Mechanics (Evaluated via report and gameplay)	Generic or copied mechanics.	Some originality, limited scope.	Good ideas, some innovation.	Creative mechanics, well implemented.	Highly original, clever mechanics, well executed.	**10%**
Aesthetics (Visuals, sound, interface, polish)	No effort in presentation.	Basic visuals, minimal polish.	Decent visuals, some polish.	Good aesthetics, consistent style.	Excellent visuals, polished experience.	**10%**
Advanced Features (Game programming patterns, USPs, AI elaboration)	No advanced features.	One or two basic patterns/USPs.	One or two patterns with partial effectiveness. One or two USPs with originality.	Two patterns effectively explained. Two USPs well integrated and explained.	Two patterns used expertly with clear articulation in video and report. Two strong USPs creatively implemented and articulated.	**10%**
Implementation of Current Research (Must be approved by module leader)	No research-based features/feature not approved by module lead.	Attempted feature, lacks validation.	One approved feature, basic implementation.	One or more approved features, well implemented.	Approved feature(s) implemented with clear link to research, well explained in report/video.	**10%**

Table 3: Feedback Template for Assessment 1

Coursework 2

Task:
This is an individual piece of work. *"Create something which does something in OpenGL"*.
The actual project is negotiated based on an **interactive prototype pitch** with 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 covered in the lecture (check Marking Rubric below).

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.

Submission Requirements:

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.
  - Gameplay description.
  - Dependencies used.
  - Use of AI description.
  - Game programming patterns that you used.
  - Any game mechanics and how they are coded.
  - UML design diagram if any
  - Sample screens.
  - Exception handling and test cases
  - 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!

Acceptable level of generative AI tool use in this assessment element

The acceptable level of GenAI use for this element is detailed with the allowed uses listed below. This is split into three categories (**Solo Work** – work must be your own with no AI support, **Assisted Work** – some uses of AI tools allowed, **Partnered Work** – AI tools integral part of the work). If you have any questions, please contact the module leader.

Solo Work	You must not use generative AI tools.	☐
Assisted Work	You are permitted to use generative AI tools in an assistive role.	☐
Partnered Work	Generative AI tool use is required as an integral part of the assessment, but transparency is required.	☒

Table 2: Acceptable Level of generative AI use

Inline with the indicated acceptable level of AI use above, the following uses are acceptable:

- Creating game assets (models, sprites, font and title, art, music and sound effects).
- Code assistant.
- Programming testing.
- Readme report crafting.

Any use of AI in your work must be declared within your documentation. You **must also include a signed Generative AI Declaration as an appendix to your submission**. The declaration form can be found on DLE (or Here on the Programme Page). This form will not be included in any word count associated with this assignment.

Assessment Criteria:
The following marking scheme will be used for this part of the assessment:

- Passing Requirements (40%)
  - Must be an original work.
  - Can compile and run.
  - Report, git and video included.
  - Program must be a playable game.
  - Use of AI declared.
- Basic Features (30%)
  - MVP
  - Textures
  - 3D Polygons with scene animation
  - Keyboard and mouse movement
  - Complex models loaded
  - PCG
- Aesthetics (10%)
- Advanced Features(10%) – Choose 2
  - Mixed textures
  - Multiple models in assimp
  - Interactive model parameters
  - BlinnPhong/PBR lighting
  - Audio
- Gamification/Research (10%)

An illustration of the feedback template is presented in Table 3.

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 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 interactive audio playback
Cap at 10%	**So choose maximum 2. Other features to be claimed depends on pitch approval, if any.**	
Last 10%	Gamification or Research-related implementation (cite a source within 5 years)	How much of the programme is a playable game?
Note you can't 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.		

Acceptable levels of AI use:
The table below provides the acceptable use categories for GenAI. Each assessment element may allow different uses. Please check the brief for each element carefully to see what uses are allowed.

Solo Work	**S1 - Generative AI tools have not been used for this assessment.**
Assisted Work	**A1 – Idea Generation and Problem Exploration** Used to generate project ideas, explore different approaches to solving a problem, or suggest features for software or systems. Students must critically assess AI-generated suggestions and ensure their own intellectual contributions are central.
	A2 - Planning & Structuring Projects AI may help outline the structure of reports, documentation and projects. The final structure and implementation must be the student's own work.
	A3 – Code Architecture AI tools maybe used to help outline code architecture (e.g. suggesting class hierarchies or module breakdowns). The final code structure must be the student's own work.
	A4 – Research Assistance Used to locate and summarise relevant articles, academic papers, technical documentation, or online resources (e.g. Stack Overflow, GitHub discussions. The interpretation and integration of research into the assignment remain the student's responsibility.
	A5 - Language Refinement Used to check grammar, refine language, improve sentence structure in documentation not code. AI should be used only to provide suggestions for improvement. Students must ensure that the documentation accurately reflects the code and is technically correct.
	A6 – Code Review AI tools can be used to check comments within the code and to suggest improvements to code readability, structure or syntax. AI should be used only to provide suggestions for improvement. Students must ensure that the code accurately reflects their knowledge and is technically correct.
	A7 - Code Generation for Learning Purposes Used to generate example code snippets to understand syntax, explore alternative implementations, or learn new programming paradigms. Students must not submit AI-generated code as their own and must be able to explain how it works.
	A8 - Technical Guidance & Debugging Support AI tools can be used to explain algorithms, programming concepts, or debugging strategies. Students may also help interpret error messages or suggest possible fixes. However, students must write, test, and debug their own code independently and understand all solutions submitted.
	A9 - Testing and Validation Support AI may assist in generating test cases, validating outputs, or suggesting edge cases for software testing. Students are responsible for designing comprehensive test plans and interpreting test results.
	A10 - Data Analysis and Visualization Guidance AI tools can help suggest ways to analyse datasets or visualize results (e.g. recommending chart types or statistical methods). Students must perform the analysis themselves and understand the implications of the results.
	A11 - Other uses not listed above Please specify:
Partnered Work	**P1 - Generative AI tool usage has been used integrally for this assessment** Students can adopt approaches that are compliant with instructions in the assessment brief. Please Specify: • Creating game assets (models, sprites, font and title, art, music and sound effects). • Code assistant. • Programming testing. • Readme report crafting.

General Guidance

Extenuating Circumstances

There may be a time during this module where you experience a serious situation which has a significant impact on your ability to complete the assessments. The definition of these can be found in the University Policy on Extenuating Circumstances here:
https://www.plymouth.ac.uk/student-life/your-studies/essential-information/exams/exam-rules-and-regulations/extenuating-circumstances

Plagiarism

All of your work must be of your own words. You must use references for your sources, however you acquire them. Where you wish to use quotations, these must be a very minor part of your overall work.
To copy another person's work is viewed as plagiarism and is not allowed. Any issues of plagiarism and any form of academic dishonesty are treated very seriously. All your work must be your own and other sources must be identified as being theirs, not yours. The copying of another persons' work could result in a penalty being invoked.
Further information on plagiarism policy can be found here:
Plagiarism: https://www.plymouth.ac.uk/student-life/your-studies/essential-information/regulations/plagiarism
Examination Offences: https://www.plymouth.ac.uk/student-life/your-studies/essential-information/exams/exam-rules-and-regulations/examination-offences
Turnitin (http://www.turnitinuk.com/) is an Internet-based 'originality checking tool' which allows documents to be compared with content on the Internet, in journals and in an archive of previously submitted works. It can help to detect unintentional or deliberate plagiarism.
It is a formative tool that makes it easy for students to review their citations and referencing as an aid to learning good academic practice. Turnitin produces an 'originality report' to help guide you. To learn more about Turnitin go to:
https://help.turnitin.com/new-links.htm?Highlight=guide
The university also had Draft Coach available on the online version of Microsoft Word. Draft Coach is a Turnitin plugin and provides instant feedback on how to address citation issues, grammar mistakes and matches with Turnitin's database.

Referencing

The University of Plymouth Library has produced an online support referencing guide which is available here: http://plymouth.libguides.com/referencing.

Another recommended referencing resource is Cite Them Right Online; this is an online resource which provides you with specific guidance about how to reference lots of different types of materials.