School of Computing, Engineering, and Physical Sciences
Assessment Guidance Coversheet

Module Code:	COMP09041	
Module Title:	AI Programming for Games	
Module Co-ordinator:	Dr. Paul Keir	
Assessment set by:	Dr. Paul Keir	
Learning Outcomes Assessed:	L2 Investigate, via programming, the techniques covered in the course. L3 Select an appropriate technique for any desired aspect of intelligence L4 Apply these techniques within software relevant to videogame development	
Issued:	Tuesday, 10th February, 2026	
Deadline:	5pm, Friday, February 27th, 2026	
Feedback:	Friday, March 20th, 2026	
Individual / Group assessment:	Individual ❑	Group ☑

Anonymity

The University Regulations makes it clear that all assessments are marked anonymously (Regulation 3.4) unless the assessment itself renders anonymity impossible e.g. placements, presentations, practical assessments. Unless otherwise stated, do not put your name or Banner ID on your submission.

Referencing

The standard referencing style at UWS is **Cite Them Right (CTR) Harvard** and your references should be in accordance with these guidelines. The University provides a drop-down menu that lets you see examples of how you should reference journals, books, websites etc. Guidance on referencing styles can be found on the UWS Library website: https://uws-uk.libguides.com/referencing.

Extenuating Circumstances

The University recognises that, from time to time, you may encounter issues which may prevent you from being able to submit or undertake an assessment. Where this is the case, you can complete an Extenuating Circumstances Submission (ECS) for consideration. The ECS will be forwarded to the School Assessment Board to take account of this declaration in recording your module marks[1]. Guidance on ECS claims can be found: https://www.uws.ac.uk/current-students/supporting-your-studies/exams-assessment-appeals/academic-appeals-extenuating-circumstances.

[1]Note that any mark / grade you receive is provisional and may be subject to change until there has been internal moderation, external examination and ratification at the School Assessment Board.

Generative AI

The type of Generative AI you are allowed to use within your assessment is:

Type	Description	Allowed
1	**Restricted** Only the use of routine and established tools, such as auto-transcription, spell checkers, grammar check is permitted.	☑
2	**Specified** Generative AI may be used for clearly delineated tasks as appropriate / allowed / recommended, although its use is **not** mandatory in order to complete the assessment.	❑
3	**Open** No specific restrictions but with requirement to track key stages / tools utilised. The use of such tools is **not** mandatory in order to complete the assessment. This may include: (i) Socratic chatbot, (ii) summarisation, simplification, synthesis, and translation, (iii) generative illustrative media content, and (iv) support production of multimedia artefacts.	❑
4	**Embedded** Generative AI is a feature of the assessment itself. Here the use of Generative AI is a focal aspect of the assessment. This may include (i) using named tools for specific outcome, (ii) output comparison and critique, and (iii) develop or error check code.	❑

Assessment Student Declaration Coversheet

All students are expected to complete the Student Declaration and insert that as the first page of your submission. The Module Co-ordinator will have uploaded this to Aula for you to complete.

- For written assessments, insert the Declaration as the first page of your document.
- For assessments not in the written format (e.g. video, audio, presentation, or practical work), submit the Declaration as a separate file.
- For group assessments, unless otherwise directed by your lecturer, the group may submit a single shared declaration.

First Assignment (Group Project)
AI Programming for Games
COMP09041

Issue Date: Tuesday, 10th February, 2026
Due Date: **5pm, Friday, February 27th, 2026**

LLM Board Game

In this assignment you will create a simple two-player board game in which a human player competes against a large language model (LLM). You are provided with a starter C++ program that combines the SDL3 graphics library, the Dear ImGui user-interface library, and the Polaris LLM chat API. The starter program is a synchronous chatbot with a colour-cycling background; you will replace this with your own board game.

This is a group project. Your team allocations are provided to you on Aula.

Your zipped submission (just one file) should include your source code and a short video. Only one team member should submit.

Background

The provided `llm-game.cpp` is a synchronous ImGui chatbot. It uses the `ChatClient` class (in `src/aipfg/`) to communicate with the Polaris LLM endpoint. Each message is sent via `chat_client_.send_message(...)`, which *blocks* until the LLM responds — causing the colour-cycling background to freeze.

During the lab you have already seen how `std::async` and `std::future` can be used to make this call non-blocking. In this assignment you will need to apply the same technique to your board game, so that the game's visual activity continues while the LLM is "thinking".

Communicating with the LLM

Your game should send the current board state to the LLM as a compact text string, and parse the LLM's reply to extract its move. For example, a noughts and crosses board might be encoded as a 9-character string such as `"_x_oxo_ox"`, read left-to-right, top-to-bottom, where _ denotes an empty cell. The LLM's system prompt should instruct it on the game rules and the expected response format. For instance:

```
"You are playing noughts and crosses. You are O.
 You will receive a 9-character board string.
 Reply with only the index (0-8) of your move."
```

You are free to choose any simple board game (e.g. noughts and crosses, Connect Four, Reversi, or another game of similar complexity). Noughts and crosses is perfectly acceptable.

Rendering the Game Board

The game board can be rendered using ImGui widgets, SDL3 rendering calls, or a combination of both. For example, a 3×3 grid of ImGui buttons can be created using `ImGui::Button`; or the board can be drawn directly with SDL3 functions such as `SDL_RenderFillRect` and `SDL_RenderLine`, appearing behind the ImGui chat pane. Either approach (or a mixture) is acceptable.

Assignment Brief

Attempt the following 8 tasks.

1. Render the game board on screen (e.g. a 3×3 grid for noughts and crosses), using ImGui widgets, SDL3 rendering calls, or both. The board should clearly display the current state of the game, including each player's pieces. **(4 points)**
2. The human player can make valid moves by clicking on the board or using keyboard input. Invalid moves (e.g. choosing an occupied cell) should be rejected or prevented. **(4 points)**
3. After the player's move, send the current board state to the LLM as a text string, and parse the LLM's response to extract its chosen move. Use a suitable system prompt to instruct the LLM on the game rules and the expected response format. **(5 points)**

4. Implement complete game logic: detect when a player has won or the game is drawn; display the outcome; allow the players to play again; and alternate who makes the first move each new game. **(4 points)**

5. Display game information and status: show whose turn it is, display a "Thinking..." indicator (or similar) while the LLM response is being awaited, and maintain a running tally of wins, losses, and draws across games. **(4 points)**

6. Average of your peer review marks from your team. Assign a mark for each team mate's efforts, along with a sentence supporting each mark. A separate Aula assignment for this is provided. **(3 points)**

7. A video (up to 2 minutes) showing your game in action, demonstrating the LLM opponent responding and gameplay continuing. Upload the video to Aula alongside your zip submission. **(3 points)**

8. No noticeable freeze or pause in the game's visual activity (e.g. the colour-cycling background, a sprite animation, or another continuous visual element) while an LLM response is being awaited. This requires using `std::async` and `std::future`. **(3 points)**

Resources

You are provided with the main C++ file `src/llm-game.cpp`, along with the utility files in the `src/aipfg/` directory (`chat-client.hpp/.cpp`, `http-client.hpp/.cpp`, `imgui-context.hpp`, `sdl3-context.hpp`, and `sdl3-typedefs.hpp`). A `CMakeLists.txt` is also provided. Use CMake and Visual Studio 2022 (or your preferred compiler) with VCPKG as in the labs.

If you choose a game other than noughts and crosses, feel free to adjust the window size, font size, or colour scheme to suit your design.

The assignment is worth 30% of the marks awarded for the entire COMP09041 module. The following provides a summary breakdown of the marking scheme:

1. Render the game board on screen (ImGui and/or SDL3)	4
2. Player can make valid moves (click / keyboard)	4
3. Send board state to LLM & parse its response	5
4. Game logic: win/draw detection, replay, alternate first move	4
5. Game status: turn indicator, "Thinking...", win/loss/draw tally	4
6. Average of your peer review marks from your team	3
7. A video showing your game in action	3
8. No noticeable freeze in the game while an LLM response is awaited	3

Plagiarism

Ensure your work is developed only by your own team. You can discuss ideas with other teams, regarding how to prepare a solution, but the *copying or sharing of code is not permitted*.

Anonymity

Please use only the Banner IDs of your team members to identify yourselves in your submission. Ensure the Banner IDs of all team members appear in a comment at the top of your source code.