### UHI_Logo_CMYK_Sm

**Web Programming — Assignment 2 2024-25**

|  |  |  |  |
| --- | --- | --- | --- |
| **Module title:** | Web Programming | | |
| **Module code**: | UI110009 | **Word-count:** | N/A |
| **Deadline:** | 13/12/2024, 5PM | **Date submitted:** | 13/12/2024 |
| **Student number:** | 21010093 | | |

**This assessment is worth 45% of the total marks for this module**.

**Completing your assessment:**

This is a programming assignment. See below for further details on the constraints of this assignment.

**Declaration of originality and authorisation to hold this assessment electronically and verify that it is original:** UHI recognises that plagiarism, where deliberately engaged in, is unacceptable and is considered serious academic malpractice. **Students are responsible for ensuring the work they submit is their own**. If you have any queries you should contact your PAT or the Module Leader before submitting your assessment.

**By submitting this assessment, I declare that the attached piece of work is my own**. I have acknowledged all the sources I have consulted and where I have used words which are not my own, I have clearly indicated this in the references.

|  |  |  |  |
| --- | --- | --- | --- |
| **Student number:** | 21010093 | **Date:** | 13/12/2024 |

Web Programming 2024-25

Assignment 2 – Create a multiplayer 2D physics game

In this assignment you will be creating a physics game that is influenced by classic 2D physics games. The actual game you produce is up to you, and it could be a variation of the game you created for the first assignment. However, it must include the following;

* Use NodeJS or similar modern framework using low latency websocket communication
* Be a multiplayer game
* Use a **server side** physics engine

Example games could include;

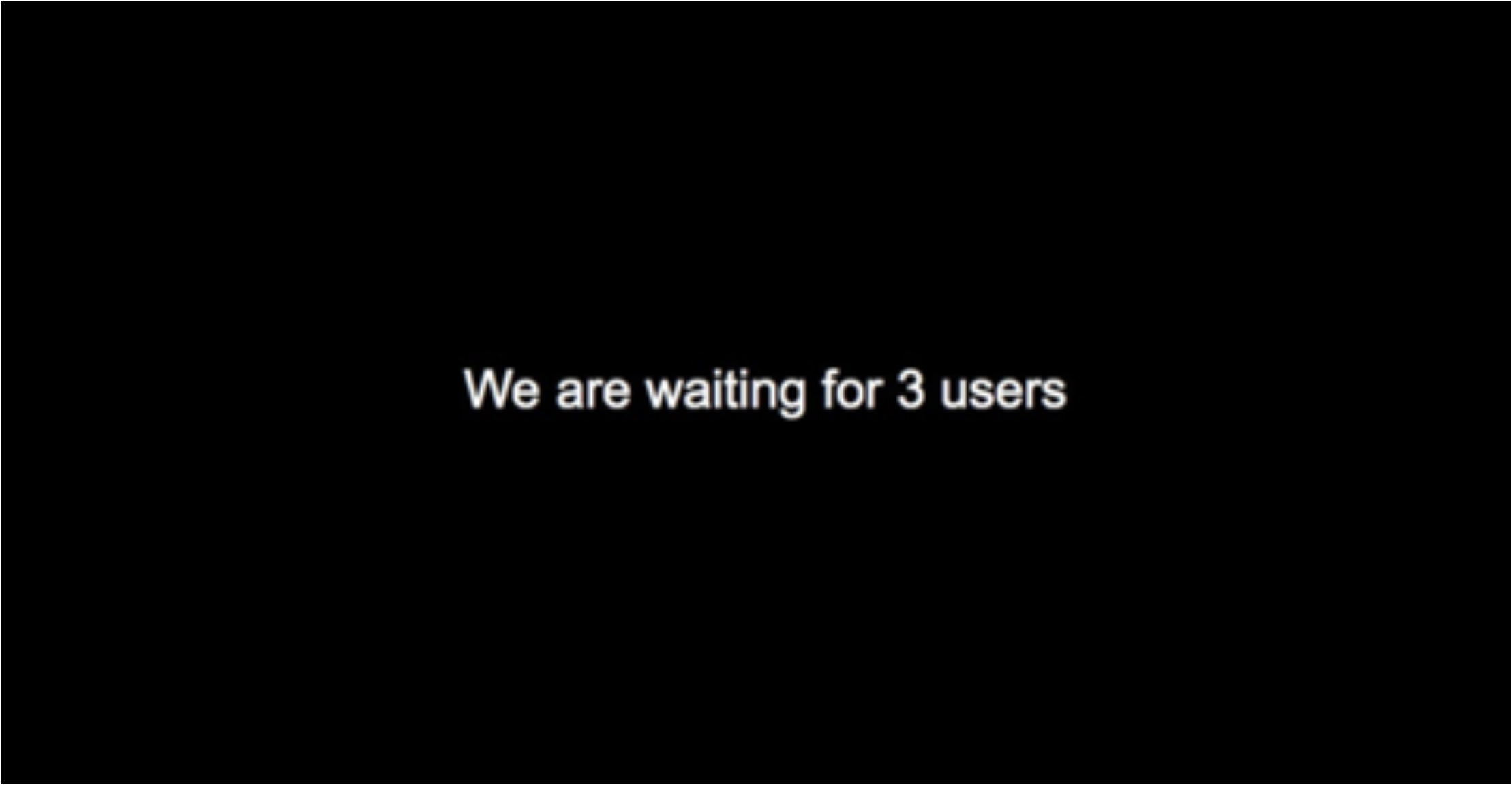
* 2D sports games – Pool, volleyball, football, air hockey etc
* .io style games – Snakes and ‘grow’ style games etc
* 2D ‘vs’ and ‘co-op’ games – tanks, stickman games, archery/artillery, top-down racing games, just about any platformer

This by no means an exhaustive list of possible game types, you should submit your ideas that you’d like to use to your lecturer.

Games must be a minimum of two players, but a good number of players will depend on the game.

On joining the game users should be required to ‘login’ with a nickname, this name will be used on the server side to associate socket connections with usernames.



The game should wait for all required users in a Lobby before starting the game proper. The lobby should offer some level of feedback to waiting players. 

Scripted or reactive viewports should be used with your game in a fashion that supports the game. This will be dependent on your game choice, so try to choose a game which naturally requires scrolling. If you’re choosing the game you developed for assignment 1 you will need to extend your levels to require viewport management.

A significant portion of the available marks have been left ‘open’ to give you flexibility to implement features within your game in line with your game’s concept.

Your submission should be done through Brightspace and include;

* All of your **code**, including information regarding the organisation / structure of any data storage
* A **Readme** with instructions on running and using the game. Include in the readme a description of the features of your game.
* A short, narrated, 5 minute **video** demonstrating running / playing the game and any solutions you implemented that you feel are of particular interest.

This must be produced using HTML5, CSS, JavaScript and a server-side physics engine. You may use any physics / graphics and DOM manipulation libraries you wish, however you may not use game developer packages. To ensure there are no issues with the tool suite you use it is recommended that you check with your lecturer should you wish to use libraries beyond Core NodeJS, Express, Socket.io, Box2DWeb-Common, jQuery, EaselJS.

There are 45 marks available for the following features;

|  |  |
| --- | --- |
| Feature | Maximum Marks |
| Server setup with NodeJS including Express, Socket.io and Physics Engine | 6 |
| Game logic – for implementing game logic with the server managing game state, scoring, and other features for example – bullets, player health, lives, teleportation, goals, win/lose conditions, and other game logic. | 15 |
| Application Flow – For implementing a fluid Splash, Login & Game Lobby, Game, High Score, and About page interface within the client / server application architecture. | 5 |
| Game Controls – For the implementation of coherent and responsive game controls, control inputs should be collected on the client and processed on the server. | 3 |
| Graphics & Audio – Adding in bitmap / vector graphics and audio. Please see \*notes on the user of external assets. | 6 |
| Viewports – For including scripted and reactive viewports | 6 |
| Maintainability and readability – Adherence to good patterns in software construction, using modular design, good quality internal documentation. | 4 |
| Total | 45 |

\*Notes – You may use any Public Domain, Open Source, or Creative Commons licensed graphics and sounds. All externally sourced graphics and sounds must include a reference to their source in your Readme file.