Assignment 1 – Web Game

COMP397 - Web Game Programming

```
**Due**: See Due Dates below
```

Overview: This is the first part of a three-part Assignment. Using the Unity Game Engine and the C# Programming Language (for Game Types 1-3) or the JS Game Engine we are building (for Game Tyeo 4 - see Game Types below) you will work in a small studios (morw than 1 amd less than5) to create a 3D Game with a **single polished level** (types 1-3) or 2D game (type 4). The game must also include a **Menu Screen, Options Screen**, and a **Game-Over Screen**.

For this first project your **target platform** will be *WebGL* (types 1-3) and HTML5+JS+WebGL2 (type 4).

Contents

1 Project Description:

1.1 Game Types to be selected

You will select one of three possible games Unity to WebGL Types (1-3) or 2D JS Game Engine based Type (4):

1. Platform Runner / Shooter
2. Isometric Tower Defense
3. Sandbox Crafting Survival Game
4. 2D Game Engine game (HTML5 + JS + WebGL2)

1.2 Deliverables - Synopsis

1.2.1 Part 1

No Functional logic is required for Part 1. In **Part 1**, you will complete the following tasks (all game types 1-4):

- Develop your game's concept
- Create an initial Game Design Document (GDD)
- Create and / or acquire assets (graphic and audio).
- Create a GitHub repository for your game
- Select a Project Management Tool to track your Game's progress.

1.2.2 Part 2

In **Part 2**, you will complete the following tasks (game types 1-3; game type 4 has freedom to implement any functionality they can):

- Add logic to the assets and UI elements you acquired and/or created in Assignment 1 Part 1. At the end of this assignment, the following features will be functional:
  - Menu Screen – all features except Load Game option
  - Game Play Screen – the game should be playable, and the main mechanics should be functional
  - Game Over Screen – all features should be fully functional
  - The player will be able to control their Avatar through a fully functional character controller (or camera)
  - Enemy AI Sensing suite will be functional
  - Pausing the game – this feature will be operational
  - Sound Effects – impulse sounds should work throughout the game
  - Music – soundtracks should be enabled for each Screen.
- Create a second draft of your Game Design Document (GDD)
- Update your GitHub repository for your game
- Update your Project Management Tool and continue to track your Game's progress.

1.2.3 Part 3

In **Part 3**, you will complete the following tasks (game types 1-3; game type 4 has freedom to implement any functionality they can):

- Add logic to the assets and UI elements you acquired and/or created in Assignment 1 Part 1 and modified in Part 2. At the end of this assignment the following additional features will be functional:
  - Game Play Screen – All game play elements for the game type selected should be functional
  - Health System – this feature should be fully implemented
  - MiniMap – this feature should be fulling functional
  - Enemy AI pathfinding will be functional
  - Saving the Game State – this feature will allow the game to be saved.
- Create a final draft of your Game Design Document (GDD) for your WebGL game
- Update your GitHub repository for your game

- Update your Project Management Tool and continue to track your Game's progress.
- Build your game for WebGL

1.3 Game Requirements - Common

Each Game of types 1-3 will include the following common requirements (the games of type 4 are going to be necessarily with much less required functionality due to time constraints of implementing the engine itself up to athe end of chapter 5):

1. **Character / Camera controller** – you may choose either a First-Person or 3rd Person character / camera controller – this includes control of the character or camera with both mouse/keyboard and game controller. Use of the new Input system is preferred.
2. **Inventory System** – this must include an appropriate User Interface and related programming structures. The feature must be activatable with player input.
3. **Health System** – this feature must include appropriate user interface components (health bars or other visual elements) and related attributes for the player, enemies and / or environment props. Player respawn and level checkpoints must also be included.
4. **MiniMap** – a secondary camera will project a radar-like view of the level, the player's position, enemy positions, and other important game landmarks. Include appropriate UI elements to support this feature.
5. **Enemy AI** – Basic intelligence will be baked into every enemy that may include but is not limited to an AI Sensing Suite (Line of sight, Sensing Radius), pathfinding (A*, NavMesh Agent) and simple steering behaviours (seek, flee, pursue, evade, arrival, flocking, pack tactics).
6. **Pausing, Saving and Loading** – you must include a mechanism to pause game play, save and load your game. You must capture the entire game state (player's health, position, inventory, enemy location and state, level changes from baseline). Include appropriate UI elements to support this feature. This feature must be activatable with player input.
7. **Sound Effects and Music** – you must include sound effects and music for each screen (Main Menu, Options, Game Play, and Game Over). Impulse sounds should provide feedback to the player for weapons used, damage dealt or sustained, explosions, UI elements selected, etc. Include at least 3 Music tracks (Menu / Options screens, Game Play Screen, Game Over Screen).

1.4 Game Requirements - Game-Type specific

Your game must also include **Game Type specific requirements**

1.4.1 Platform Runner / Shooter

a. **3D Platforms** – create at least 3 types (e.g., static, moving, ramps, elastic, spinning, swinging etc.). The player must be affected by physics. Long falls should kill the player.

b. **Hazards** – create at least 3 types (e.g., spike pit, collapsing platform, timed flames)

c. **Goal** – your game must include a goal to reach. This could be a door, elevation, tunnel chest or non-player character (NPC) to save.

1.4.2 Isometric Tower Defense

a. **Resource Gathering** – the player will start with an amount of in-game currency or resources that may be used to build or purchase towers. At least 3 different resources will be available on the map (e.g., wood, stone, electricity, gunpowder, metal). Towers should require at least 2 different resources to be built. Towers take time to build but cost less resource points than purchasing them outright.

b. **Tower Types** – create at least 3 tower types (e.g., single target, multi-target, force field, resource gathering). Towers may be purchased with resource points and are upgradable. Your game must include a tower limit (e.g., maximum 5). Towers may be placed anywhere on the map. Towers are destructible.

c. **Enemy Types** – create at least 3 enemy loadout variations (e.g., higher movement speed, higher health, damage resistance, flying, tunneling). Include at least 3 enemy goal variations (e.g., tower destroyers, resource stealers, base attackers, enemy spawners).

1.4.3 Sandbox Crafting Survival Game

a. **Single themed biome** (small sandbox) – 64 Voxels / Unit square recommended. The level must be Procedurally generated.

b. **Day / Night Cycle** or **Countdown** – Enemies come out at night. They seek the player. They are destroyed during the day.

c. **Destructible environment** – at least 3 types of tiles / voxels (e.g., Rock, Dirt, Grass) and at least 3 terrain features (e.g., trees, water, lava). All destructible tiles should have at least 3 states (whole, damaged, destroyed)

1.4.4 2D Game Engine game (HTML5 + JS + WebGL2)

a. Because of the limited time and game engine functionality, this option will be allowed to have much more **limited scope** and **free rein** as to the features to be included, as long as the studio is using all the functionalities of the game engine implemented up to end of week 6 (predicted to be end of chapter 5 - keyboard input, scene level loading, audio, simple shaders, textures, sprites and fonts).

1.5 Assignment Deliverables - Detailed:

1.5.1 Common Requirements / 40

1. This version of your application will have the following characteristics and components:

2. A **Menu Screen** (the Game Start State) – that will allow the user to get ready and displaying at least 4 options: **New Game, Load Game, Options** or **Exit**. Ensure you acquire (or create) UI Elements that are appropriate for this Screen. Include an appropriate background image or tile map.

3. An **Options Screen** – will display **audio options** (e.g., Music Volume, Sound Volume), **control options** (e.g., key mapping options, inverted Y axis).

4. A **Gameplay Screen** This is where the main game occurs. Your 3D game will include **one polished Game Level**. For this assignment part, you will create only the UI elements. Place your player's avatar and enemies on the Screen in a Static Pose. For this Assignment part you **must include** an appropriate skybox, lighting and environment assets.

5. A **Game-Over screen** (the Game End State) – this will display the player's final score (or other vital statistics) and give the player the option to **Play Again** or **Return to the**

Main Menu Screen. This Screen can be mostly completed. Create a UI element that allows the player to **Go Back to the Game-Play Screen**. Include an appropriate background graphic or tile map.

6. The Player will control an **Avatar** (a vehicle, character or camera) – the main input may be a combination of mouse and keyboard as well as game controller. The player's avatar may have **weapons** or other **devices** that they can use to defeat the computer-controlled enemies. Select an appropriate Avatar for your game and place it in the **Game-Play Screen** where it would normally start.

7. An **Inventory System**. You will develop the User Interface for this Inventory system. You will Acquire (or create) appropriate assets to support this feature. This feature does not need to function for this assignment. However, it must be **activaed** by a key press or other type of player input (e.g., Game controller key press).

8. A **Health System** (e.g., health bars or visual elements for the player, enemies or other environment props). You will Acquire (or create) appropriate UI elements to support his feature in the Game-Play Screen.

9. A **MiniMap** that uses a secondary camera to project the Game Play level onto the Game's Canvas. Acquire or create appropriate sprite assets to use for the User Interface. The MiniMap does not need to function for this assignment.

10. Enemies with the **AI behaviour** detailed above (*sensing suite*, *pathfinding* and *steering behaviours*). The enemies or hazards in the game should be abundant enough to challenge the player but not be impossible to beat. You will acquire (or create) appropriate enemy assets and place them in the Game-Play Screen in an appropriate pose (see below in the Game Type section for more details)

11. Your Game Play Level must include a User Interface that allows the player to **Pause** the Game, **Save** the Game State and **Load** the game from a specific game state snapshot. Acquire (or create) appropriate user interface elements and sprites to support this functionality. This feature does not have to be functional for this assignment. However, it must be activatable by a key press or other type of player input (e.g., Game controller key press).

12. **Sound effects** for collisions with enemies, collecting points, shooting attacks, explosions, UI element selection, etc. Acquire (or create) appropriate assets and place them in an appropriate folder in your project.

13. Your game will include several **Game soundtracks** (one for each Screen). **Acquire (or create)** appropriate assets and place them in an appropriate folder in your project

1.5.2 Game Type Specific Requirements / 30

A. Platform Runner / Shooter (option)

- **3D Platforms** – Acquire (or create) at least 3 platform types (e.g., static, moving, ramps, elastic, spinning, swinging etc.). Build your level by placing these platforms in the appropriate locations.
- **Hazards** – Acquire (or create) at least 3 types (e.g., spike pit, collapsing platform, timed flames). Place these hazards in the appropriate locations.
- **Goal** – Acquire (or create) assets to represent your level goal. Place these assets in the appropriate locations.

B. Isometric Tower Defense (option)

- **Resource Gathering** – Acquire (or create) appropriate resource assets for the game. Create appropriate UI elements to support this feature.
- **Tower Types** – Acquire (or create) appropriate tower assets for the game. Place the towers in various locations to test their size (height) and lighting.
- **Enemy Types** – Acquire (or create) appropriate enemy assets for the game. Place these assets in a static pose to represent their potential locations in your level.

C. Sandbox Crafting Survival Game (option)

- **Single themed biome** (small sandbox) – Acquire (or create) appropriate voxel assets for the game. Procedurally generating your level is not required at this point. However, you should manually build out a small portion of your biome to demonstrate how your level might appear.
- **Day / Night Cycle or Countdown** – Acquire (or create) appropriate UI Assets to support this feature (e.g., create Labels in your Canvas that would notify the player that they should be aware that the enemies are approaching).
- **Destructible environment** – Acquire (or create) appropriate voxel animation and / or particle effects for each stage of voxel tile destruction (whole, damaged and destroyed).

D. Game Built with the newly created 2D HTML5 + JS + WEbGL2 Game Engine (option)

- Because of the limited time and game engine functionality, this option will be allowed to have much more **limited scope** and **freee rein**, as long as the studio is using all the functionalities of the game engine implemented up to end of week 6 (predicted to be end of chapter 5 - textures/sprites and fonts).

1.5.3 Game Design Document (all types) / 10

Include a first draft of the Game Design Document (GDD) for your game that includes:

- **A Tile page** with Company Logo, Game Name, Authors Name(s) and Student ID(s)
- **Table of Contents**
- **Version History** – ensure you include details for each version of your code
- **Game Type** – describing which Game Type you've selected (Platform Runner / Shooter, Isometric Tower Defense / Sandbox Crafting Survival Game or JS 2D Game Engine - based game)
- **MDA** – describe your Game's Mechanics, Dynamics and Aesthetics.
- **Controls**. Include how your game will be controlled, including the Input scheme (mouse and keyboard and/or game controller).
- **Level and UI Sketches** – this section should include wireframes of each of your game screens (Menu Screen, Options Screen, Game-Play Screen, Game Over Screen) with appropriate labels. Include sketches for the Health System, Inventory System, and your Pausing/Saving/Loading feature. Please use an appropriate tool to create these. Hand-drawn sketches will not be accepted. This is an important part of the planning process for your game.
- **Screen Captures** – Include at least 4 screen shots for your game: 1 for your Start Screen, 1 for your Gameplay Screen, 1 for your Game-End Screen and 1 that captures your Health System, 1 that captures your Inventory System, 1 that captures your Pausing/Saving/Loading feature. The Screen Captures should only be created after you acquire (or create) appropriate assets for your game.
- **Characters / Vehicles / Camera** – Describe the character's Avatar or camera - Enemies (AI) – Describe the computer-controlled enemies (AI) and how they function. Describe their desired behaviour
- **Weapons** – Describe any weapons available to the player
- **Sound Index** – Include an index of all your sound clips
- **Art / Multimedia Index** – Include examples of your image assets. Each image should be displayed as a thumbnail

1.5.4 Internal Documentation (all types) / 5

Include **Internal Documentation** for your program:

- Ensure you include a program header for each module of your game that indicates: The Source file name, Author's name, Student Number, Date last Modified, Program description and Revision History

1.5.5 Version Control (all types) / 5

Share your files on **GitHub** to demonstrate Version Control Best Practices:

- Your repository must include **your code** and be well structured
- Your repository must include **commits** that demonstrates the project being updated at different stages of development – each time a major change is implemented

1.5.6 Demo Video (all types) / 10

Create a **Short Video** presentation with your favourite screen capture and streaming tool (OBS Recommended) and upload it to your Learning Management System. You must also include a short PowerPoint (or Google Slides) Slide Deck that includes a **single slide** to start your video:

- The first (and only) Slide of your Slide Deck must include a **current image of you (and your partner)** (no avatars allowed) that is displayed appropriately on the page. You must also include your Full Name(s), Student ID(s), the Course Code, Course Name, and your Assignment information.
- You will **demonstrate** your game's Screens. Ensure you include a simple mechanism to switch Screens. Your UI must be clearly visible
- You will **describe** the design for your Game
- Sound for your Video must at an appropriate level so that your voice may be clearly heard. Your Screen should be clearly visible
- Your Short Video should run no more than 5 minutes.

Note: Your project will not be accepted without your video demo!!!

1.6 SUBMITTING YOUR WORK

Your submission should include:

1. An external Game Design Document (MS Word or PDF). You should use the example document provided as a template. This will be your first draft.
2. A working link to your project files on GitHub. Ensure that the repo is appropriately named.
3. Your project files zipped and submitted to your Learning Management System. Rar files will not be accepted.
4. Your short Video Demo link uploaded to your Learning Management System.
5. Indicate in your submission which agile project management tool you will use to track your progress (e.g., Trello, Jira)

1.6.1 Marking and Due Dates

Part #	Mark (% of Final Mark)	Due Date/time
Part 1	5%	May 31st, 2024 11:59 PM
Part 2	10%	June 10th, 2024 11:59 PM
Part 3	15%	June 17th, 2024 11:30 AM

1.6.2 Late submissions:

- 20% deducted for each additional day.

1.6.3 External code

- External code from the internet or other sources) can be used for student submissions within the following parameters:
- The code source (i.e. where you got the code and who wrote it) must be cited in your internal documentation.
- It encompasses a maximum of 10% of your code (any more will be considered cheating).

- You must understand any code you use and include documentation (comments) around the code that explains its function.
- You must get written approval from me via email.

GOOD LUCK :)