COMP397 – Web Game Programming

Assignment 1 – WebGL Game - Part 1

Due: Week #2 at midnight

Value 5%

WebGL Game

Maximum Mark: 100

Overview: This is the first part of a three-part Assignment. Using the Unity Game Engine and the C# Programming Language you will work in a small group (alone or with a partner) to create a 3D Game with a **single polished level**. 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. **No Functional logic is required for this part**.

In this part, you will complete the following tasks

1. develop your game's concept
2. create an initial Game Design Document (GDD)
3. create and / or acquire assets (graphic and audio).
4. create a GitHub repository for your game
5. Select a Project Management Tool to track your Game's progress.

Part 1. Project Description:

You will select one of three possible games Types:

1. Platform Runner / Shooter
2. Isometric Tower Defense
3. Sandbox Crafting Survival Game

Each Game will include the following **common requirements**:

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).

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

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.
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).
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)

Part 2. Assignment Deliverables:

Common Requirements / 40

1. This version of your application will have the following characteristics
   - 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.
   - An **Options Screen** – will display **audio options** (e.g., Music Volume, Sound Volume), **control options** (e.g., key mapping options, inverted Y axis) and **key-mapping options**.
   - **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.
   - 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.
   - 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
   - Your game must include 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 **activatable** by a key press or other type of player input (e.g., Game controller key press).

- Your game must include 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**.
- Your game must include 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.
- Your game will include **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)
- 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).
- Your game will include **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.
- 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

Game Type Requirements / 30

A. Platform Runner / Shooter (option)
   1. **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.
   2. **Hazards** – Acquire (or create) at least 3 types (e.g., spike pit, collapsing platform, timed flames). Place these hazards in the appropriate locations.
   3. **Goal** – Acquire (or create) assets to represent your level goal. Place these assets in the appropriate locations.

B. Isometric Tower Defense (option)
   1. **Resource Gathering** – Acquire (or create) appropriate resource assets for the game. Create appropriate UI elements to support this feature.
   2. **Tower Types** – Acquire (or create) appropriate tower assets for the game. Place the towers in various locations to test their size (height) and lighting.
   3. **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)
   1. **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.
   2. **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).
   3. **Destructible environment** – Acquire (or create) appropriate voxel animation and / or particle effects for each stage of voxel tile destruction (whole, damaged and destroyed).

Game Design Document / 10

2. 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 or Sandbox Crafting Survival 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

Internal Documentation / 5

3. 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

Version Control / 5

4. 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

Demo Video / 10

5. 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 in the Unity Editor. 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

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)

This assignment is weighted **5%** of your total mark for this course.

Late submissions:

- 20% deducted for each additional day.

External code (e.g. from the internet or other sources) can be used for student submissions within the following parameters:

1. The code source (i.e. where you got the code and who wrote it) must be cited in your internal documentation.
2. It encompasses a maximum of 10% of your code (any more will be considered cheating).
3. You must understand any code you use and include documentation (comments) around the code that explains its function.
4. You must get written approval from me via email.