COMP397 – Web Game Programming

Assignment 1 – WebGL Game - Part 2

Due: Week #4 at midnight

Value 10%

WebGL Game

Maximum Mark: 100

Overview: This is the second 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.

In this part, you will complete the following tasks

1. 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:
    a. **Menu Screen** – all features except Load Game option
    b. **Game Play Screen** – the game should be playable, and the main mechanics should be functional
    c. **Game Over Screen** – all features should be fully functional
    d. The player will be able to **control their Avatar** through a fully functional character controller (or camera)
    e. Enemy **AI Sensing suite** will be functional
    f. **Pausing the game** – this feature will be operational
    g. **Sound Effects –** impulse sounds should work throughout the game
    h. **Music –** soundtracks should be enabled for each Screen.
2. create a second draft of your Game Design Document (GDD)
3. update your GitHub repository for your game
4. update your Project Management Tool and continue to track your Game's progress.

Part 1. 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.** When the **New Game option** is selected the game will transition to the Play Scene. When the **Load Game option** is selected the game will display a **UI panel** (or other UI elements) that will be used to load a saved game. This feature will not be completely functional at this point. When the user selects the **Option button**, the game will transition to the **Options screen**. When the user selects the **Exit option**, the game Application will **Quit**.
   - 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**. Write code to enable these features.
   - **A Gameplay Screen** This is where the main game occurs. Your 3D game will include **one polished Game Level.** The Game State will encompass the game world and host all of the game play. This will vary according to the game Type selected (see Game Type requirements below).
   - 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**. Sound Effects for player selections and Music for the Scene should also be included.
   - 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 controller code should allow the player's avatar to trigger **weapons** or other **devices** that they can use to defeat the computer-controlled enemies. Appropriate Animation should be included for the player's avatar
   - Your game must include an **Inventory System**. You will develop the User Interface for this Inventory system. This feature will remain a placeholder for this release and **will not** be functional.
   - Your game must include a **Health System** (e.g., health bars or visual elements for the player, enemies or other environment props). This feature will remain a placeholder for this release and will not be functional.
   - Your game must include a **MiniMap** that uses a secondary camera to project the Game Play level onto the Game's Canvas. The MiniMap should appear in the Player's **canvas** but is not required to function at this time.
   - Your game will include **enemies** with the AI behaviour detailed above (**sensing suite**, **pathfinding** and **steering behaviours**)**.** In this release the Enemies will have their **sensing suite** operational (Line of Sight and Detection radius should be functional). When an enemy detects the player, they will attack if they are in range.
   - 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. The Pause / Save / Load panel must be **activatable** by a key press or other type of player input (e.g., Game controller key press). Pausing the game will be **fully functional** at this point. Saving and Loading the game **will not** be operational.

- Your game will include **sound effects** for collisions with enemies, collecting points, shooting attacks, explosions, UI element selection, etc. Write code to enable this functionality.
- Your game will include several **Game soundtracks** (one for each Screen Write code to enable this functionality

Game Type Requirements / 35

A. Platform Runner / Shooter (option)
   1. **3D Platforms** – All 3D Platforms should be enabled at this point and allow the player to navigate the level
   2. **Hazards** – Hazards will be in place but will not be functional.
   3. **Goal** – Level Goal should exist but will not allow the player to complete the game.

B. Isometric Tower Defense (option)
   1. **Resource Gathering** – UI elements should be **activatable** for this feature. This feature will work with the Inventory system. However, gathering, storing, and expending resources **will not** function.
   2. **Tower Types** – The placing of Towers in the game world will be functional. Select 1 fully functioning tower type to include in the game at this point.
   3. **Enemy Types** – Enable one enemy type to function. It is recommended that you include the base attacking enemy.

C. Sandbox Crafting Survival Game (option)
   1. **Single themed biome** (small sandbox) – You should be able to procedurally generate several layers of your biome using Perlin noise (or another sampling method).
   2. **Day / Night Cycle or Countdown** – This feature will be **partially functional** at this point. You should create a timer that ticks in real time to dim the lighting and create a darker environment. No post-processing effects are required at this point.
   3. **Destructible environment** – This feature will not be functional at this point.

Game Design Document / 5

2. Include a second 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 an update here that the features you have enabled with your code
   - **MDA** – Update your Game's **Mechanics**, **Dynamics** and **Aesthetics** based on what you have learned during this development cycle.

- **Screen Captures –** Include an update in this section with screen shots that capture gameplay for this first playable release. You may include short animated gifs where appropriate.

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
    - Ensure you include a header for all your functions and classes
    - Ensure your program uses contextual variable names that help make the program human-readable
    - Ensure you include inline comments that describe your code.

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 **10%** 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.