COMP305 – Game Programming 1

Assignment 3
Simple 3D Game

- ❖ **Part 1** – **Game Plan – Value 5%**
  - GDD (Excluding Screen Shots), GitHub Repo, and Basic Assets
  - Player Movement on 3D Level, Game Mechanics
  - **Due Week #9** (Monday November 14, 2016) @ midnight
- ❖ **Part 2 – Final Version – Value 5%**
  - Simple Enemies / Hazards
  - Scoring System and Collectibles
  - Updated GitHub Repo
  - **Due Week #10** (Monday November 21, 2016) @ midnight.

Total Value 10%

Simple 3D Game **Maximum Mark: 54**

Overview: Following the 3D Tutorials in class, working with a partner (or solo) you will create your own 3D game. Your 3D game may use either a **First Person** or **Third Person** Perspective. The game must include **enemies / hazards** for the player to **avoid and/or destroy**. A **scoring system** must also be included. You must use your own graphic and sound assets.

Instructions :

(18 Marks: GUI, 18 Marks: Functionality, 5 Marks: Internal Documentation, 9 Marks: External Documentation, Version Control: 4 Marks)

1. Your Game will have the following characteristics **(18 Marks: GUI, 18 Marks Functionality)**
   a. A **Gameplay Scene** where the main game occurs. (2 Marks: GUI, 2 Marks: Functionality)
   b. Player control of an **Avatar** (a vehicle or character) – use the standard keys (WASD) for player movement (optional mouse control for First Person Perspective option) and spacebar for Jumping (6 Marks: GUI, 6 Marks: Functionality).
   c. Computer control (simple AI) of the **enemies and / or Hazards to avoid**. The enemies should be abundant enough to challenge the player but not be impossible to beat. (4 Marks: GUI, 4 Marks: Functionality)
   d. Random placement of items to collect and/or obstacles to pass through or over – this will generate points for the player (2 Marks: GUI, 2 Marks: Functionality)

e. A **Scoring system** – ensure that the player's score is accurately calculated and displayed somewhere on the **Gameplay screen** (1 Mark: GUI, 1 Mark: Functionality).
f. The player must have a **life counter** or **health status** that decreases each time his **avatar** collides with an enemy (2 Marks: GUI, 2 Marks: Functionality)
g. Add sound effects for collisions with enemies and collecting points (1 Marks: GUI, 1 Mark: Functionality).

2. Include **Internal Documentation** for your program **(5 Marks: Internal Documentation):**
    a. Ensure you include a program header for each module of your game that indicates: the Source file name, Author's name, Last Modified by, Date last Modified, Program description, Revision History (2 Marks: Internal Documentation).
    b. Ensure you include a header for all of your classes and methods (1 Marks: Internal Documentation
    c. Ensure your program uses contextual variable names that help make the program human-readable (1 Marks: Internal Documentation).
    d. Ensure you include inline comments that describe elements of your GUI Design for your 2D game (1 Marks: Internal Documentation)
3. Include **External Documentation (Game Design Document)** for your program that includes **(9 Marks: External Documentation)**:
    a. **A company Logo** (0.5 Marks: External Documentation).
    b. **Table of Contents** (0.5 Marks: External Documentation).
    c. **Version History** – ensure you include details for each version of your code (1 Mark: External Documentation).
    d. **Detailed Game Description** – describing how your game works (1 Mark: External Documentation).
    e. **Controls** (0.5 Mark: External Documentation).
    f. **Interface Sketch** – this section should include wireframes of each of your game screens with appropriate labels (2 Mark: External Documentation)
    g. **Screen Descriptions** – Include at least 3 screen shots for your game: 1 for your Start State, 1 for your Gameplay State and 1 for your Game-End State (1 Mark: External Documentation).
    h. **Characters / Vehicles** – Describe the character's Avatar (0.5 Mark: External Documentation).
    i. **Enemies** – Describe the computer controlled enemies and how they function (0.5 Mark: External Documentation).
    j. **Scoring** – Describe how the player can score and how the score is calculated (0.5 Mark: External Documentation).
    k. **Sound Index** – Include an index of all your sound clips (0.5 Mark: External Documentation).
    l. **Art / Multimedia Index** – Include examples of your image assets. Each image should be displayed as a thumbnail (0.5 Mark: External Documentation).
4. Share your files on **GitHub** to demonstrate Version Control Best Practices **(4 Marks: Version Control).**

a. Your repository must include **your code** and be well structured (2 Marks: Version Control).
b. Your repository must include **commits** that demonstrates the project being updated at different stages of development – each time a major change is implemented (2 Marks: Version Control).

Optional Game Features (i.e. Potential Bonus Marks).

A. Empower the computer controlled enemies to fire bullets.
B. Include a **MiniMap** for your Game.
C. Include Particle effects for Explosions.
D. Include Particle effects for Bullet Impacts.
E. Include a final "boss monster" to avoid.
F. Add power-ups for the player's **avatar** (e.g. extra speed, a shield)
G. Add a cool soundtrack to the game.
H. Add procedural 3D map generation

I. Feature	Description	Marks
GUI / Interface Design	UI Controls meet the application requirements. Display elements are deployed in an attractive manner. Appropriate contrast is applied to application UI Controls and any background colours applied so that all text is legible.	18
Functionality	The program's deliverables are all met and the program functions as it should. No errors appear as a result of execution. User Input does not crash the program.	18
Internal Documentation	A program header is present and includes the name of the program, the name of the student, student number, date last modified, a short revision history and a short description of the program. All methods and classes include headers that describe their functionality and scope and follow commenting best practices. Inline comments are used to indicate code function where appropriate. Variable names are contextual wherever possible.	5
External Documentation	Include an external document via MS Word or PDF (or README file on GitHub) that includes all the sections that are relevant to the game. Ensure there are no spelling or grammar errors.	9
Version Control	GitHub commit history demonstrating regular updates.	4
Total		**54**

SUBMITTING YOUR WORK

Your submission should include:

1. An external document (MS Word or PDF) – Alternatively include all your documentation in a README.md file on GitHub.
2. A zip archive of your WebGL Build on e-centennial
3. A link to your complete project files on GitHub

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.