COMP305 – Game Programming 1

Final Project – Value 30%
Single Player Unity Game

Part 1 (Game Concept, First Working Level) – Value 10%

- **First Draft of External Game Document, Menu Scene**
- **First Working Level, Basic Game Mechanics, Scoring System**

Due Week # 12 (Monday December 5), 2016 @ midnight

Part 2 (Second and Third Level) – Value 10%

- **Second and Third Working Levels**

Due Week # 13 (Monday December 12), 2016 @ midnight

Part 3 (Finished Version) – Value 10%

- **Instruction and Game End Scenes**

Due Week #14 (Monday December 19, 2016) @ midnight.

Single Player Unity Game **Maximum Mark: 72**

Overview: Either alone or as a Team of up to 3, you will create an original 2D or 3D unity game for solo play. The game can be a scrolling game, a platform game, tower defense, First Person Shooter, Third Person Adventure or other game (with approval). Puzzle games **will not** be accepted. The game must have a **Menu Scene**, an **Instructions Scene** and at least **3 Game Level Scenes**, and a **Game Over Scene**. A **scoring system** must also be included. You must acquire and/or use your own graphic and sound assets (using assets from the Unity Asset Store is acceptable). **Note:** you may use the output from an assignment from either team member as a starter template.

Instructions :

(25 Marks: GUI, 25 Marks: Functionality, 5 Marks: Internal Documentation, 11 Marks: External Documentation, 6 Marks: Version Control)

1. Your application will have the following characteristics **(25 Marks: GUI, 25 Marks Functionality)**
    a. A **Menu Scene** – that will allow the user to get ready and displaying at least 3 options: **Play**, **Instructions** and **Exit** (2 Marks: GUI, 2 Marks: Functionality)
    b. An **Instructions Scene** – will display rules and instructions on how to win the game (2 Marks: GUI, 2 Marks: Functionality)

c. **Gameplay Scenes** – This is where the main game occurs. The game will have at least **3 Game Level Scenes.** Each Game Level must appear and function differently than other game levels. (6 Marks: GUI, 6 Marks: Functionality).
d. **Game Over Scene** (the Game End State) – this will display the player's final score and give the player the option to **Play Again** or **Exit to Menu** (2 Marks: GUI, 2 Marks: Functionality)
e. Player control of an **Avatar** (a vehicle or character) – the main input may be a combination of mouse and keyboard clicks. The player's avatar may have **weapons** or other **devices** that he can use to defeat the computer controlled enemies (3 Marks: GUI, 3 Marks: Functionality).
f. Computer control (AI) of the **enemies**. The enemies should be abundant enough to challenge the player but not be impossible to beat. (3 Marks: GUI, 3 Marks: Functionality)
g. Random opportunities to generate points for the player aside from killing enemies (2 Marks: GUI, 2 Marks: Functionality)
h. 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).
i. The player must have a **life counter** or **health status** that decreases each time his **avatar** is "killed" (1 Mark: GUI, 1 Mark: Functionality)
j. Add **sound effects** for collisions with enemies, collecting points, shooting attacks, explosions, etc. (2 Marks: GUI, 2 Mark: Functionality).
k. Add a **Game soundtrack** (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 functions and classes (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 arcade game (1 Marks: Internal Documentation)
3. Include **External Documentation** for your program that includes **(11 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 (1.5 Marks: External Documentation)

g. **Screen Descriptions** – Include at least 6 screen shots for your game: 1 for your Start Screen, 1 for your Gameplay Screen, 1 for your Game-End Screen and 1 for each level of difficulty (2 Marks: External Documentation).
h. **Game World** – Describe your game environment (0.5 Mark: External Documentation).
i. **Levels** – Describe each of your game levels or challenge levels (0.5 Mark: External Documentation).
j. **Characters / Vehicles** – Describe the character's Avatar (0.5 Mark: External Documentation).
k. **Enemies** – Describe the computer-controlled enemies and how they function (0.5 Mark: External Documentation).
l. **Weapons** – Describe any weapons available to the player (0.5 Mark: External Documentation).
m. **Scoring** – Describe how the player can score and how the score is calculated (0.5 Mark: External Documentation).
n. **Sound Index** – Include an index of all your sound clips (0.5 Mark: External Documentation).
o. **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 **(6 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).
    c. Ensure that each member of your team contributes equally to GitHub (2 Marks: Version Control).

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

A. Include a final "boss monster" to defeat. (4 Bonus Marks)
B. Add power-ups for the player's **avatar** (e.g. extra speed, a shield) that he can add to his "inventory" and use whenever he chooses. (4 Bonus Marks)
C. Create a mini-game in a section of the main game (e.g. disarm a bomb, pick a lock, etc.) (8 Bonus Marks)
D. Empower the player to gain an NPC (non-player character) computer-controlled ally. (4 Bonus Marks).
E. Make it possible for the player to save / pause / load his game (4 Bonus Marks).

EVALUATION CRITERIA

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.	25
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.	25
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.	11
Version Control	GitHub commit history demonstrating regular updates.	6
Total		**72**

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 **30%** 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.