UNIVERSITÉ Concordia UNIVERSITY

ACADEMIC YEAR: 2019-2020

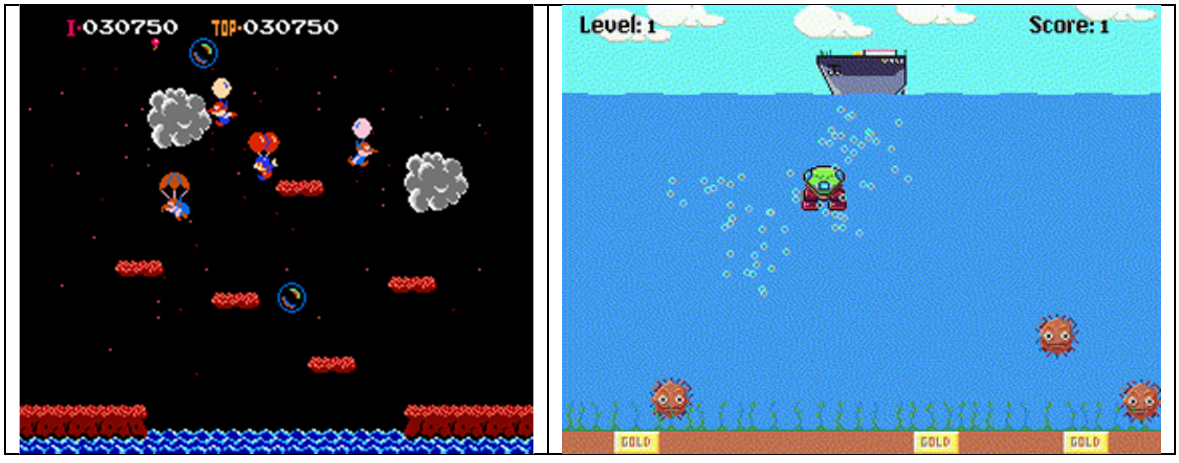
ASSIGNMENT 2 (2D Game)

Assignment Posted: September 21, 2019
Assignment Due: October 18, 2019 before 11.59pm
Final Deadline with 20% flat Penalty: October 21, 2019 before 11.59pm

Overview:

Balloon Fight (originally an arcade video game developed by Nintendo for its VS system) was a popular video game during the 1980s. The original 2D game had a balloon fighter with two balloons attached to him which help him float/fly. Enemies would shoot at the player balloons. If one of the balloons popped, it would become difficult for the player to rise higher. If both the balloons were to pop, the player would lose one life. The player too would shoot at enemy balloons in order to pop both their balloons which would kill them, giving player some points. There would be waves of the enemies, and the goal was to stay alive as long as possible.

Your first Unity programming assignment is to develop a 2D variation of *Balloon Fight*. In this variation rather than being set in air, the game will take place under water, with the balloons being replaced by oxygen cylinders that help you breathe under water. Enemies will be replaced by sharks (moving around in straight lines) which can damage the player by simply touching the player. Finally, there will be moving octopus appearing once in a while to try and kill the player. Both sharks and octopus will move from one side of the screen to the other, and disappear after a fixed time. Newer enemies will keep spawning after a fixed interval. You should have variations on the size of sharks by size and speed. The goal of the player is to retrieve gold bars available at the bottom (which should keep spawning in various locations) and bring them to a ship parked at the surface level.



For 2D assume that the enemies move in straight lines. As in the original game, the game world wraps around: if anything (player, enemies, gold bar, …) passes out of the viewable game world they reappear on the opposite side of the viewable game world (except for the octopus the only move horizontally back and forth).

Basic Game Play:

Points are awarded for gold bars collected and the value is based on weight of the gold bar:

- A smaller gold bar earns 1 point each time, a medium size gold bar earns 2 points, and a large bag of a few gold bars earns 10 points each. The collectibles (gold bars and bags) will be spawned in various places and are only available for a fixed time. The size of collectible chosen should impact player's ability to swim (increased difficulty with larger size).
- To add to the challenge, the enemy speed will increase after a fixed time hinting towards change of level.
- The player will start off with two (2) extra lives. If the player is touched by an enemy, one of the oxygen cylinder will disappear. If both the cylinders are gone, the player drowns in the water and then reappears at the ship on top.
- The octopus appear at least twice in each level at random times.
- Even if the player holds down the key (or equivalent) that helps him swim upwards, there is a small delay *i.e.* (continuous upwards motion is not possible).

Game Versions:

You have to develop two versions of the game as follows:

- ***Version-Normal*** – In this version, the basic version of the game is implemented as above. The game ends when the player runs out of lives.
- ***Variant-Special*** - This game is a variation of the above game. In this game you can get a nitro cylinder to swim super-fast in any direction (use a special key to toggle this mode) for a limited time (say, 3 seconds). The game ends when player runs out of lives.

User interface and gameplay parameters:

- Design your own user interface using keys for manipulating the player.
- The current score should be displayed on the screen.
- You will have to suitably adjust your gameplay parameters, such as number and types of enemies, how frequently are collectibles spawned, where do they spawn and are of what kind, speed and speed change factors, etc., so as to yield a playable game.

Submission:

Assignment must be submitted only through Moodle. No other form of submission will be considered. Please create a zip file containing your source code, all the assets, data files, a README (.txt) file explaining how to read the code, compile, run, and play the game on a PC with Unity. You must also record and submit a short video of the game play in the same zip file. The zip file should be named Assignment#_YourStudentID.

Evaluation Procedure:

You MUST demonstrate your program to the lab instructor during lab hours, right after the due date. You must run your submitted code, demonstrate its full functionality and answer questions about the Unity programming aspects of your game. Major marking is done on the spot during the demo. Your code will be further checked for structure, non-plagiarism, etc. However, ONLY demonstrated submissions will receive marks.

Evaluation Scheme:

1. Working implementation of Normal version with all game play: (40%)
2. Working implementation of Variant Special with all game play: (15%)
3. Setting and playability (User Interface, Game world, …): (10%)
4. Aesthetics and overall impression: (20%)
5. Q &A (to demonstrate understanding of Unity programming): (15%)

Ground Rules:

You are welcome to discuss high-level implementation issues with your classmates and others, but you should avoid actually looking at other students' code as whole, and under no circumstances should you be copying any portion of another student's code or copying complete code from the Internet. However, seeking help from other students for debugging some portion of your code is reasonable. Basically, these "ground rules" are intended to prevent a student from "freeloading" off another student or from the Internet, even accidentally.