COMP3064 – Game Programming

Midterm

Due in-class

Time: 100min

Value 20%

Practical open-book exam

Sidescroller **Maximum Mark: 50**

Overview: You are provided with a project that contains all required assets and scripts. Provided scripts, however, are not complete. Your task is to complete the scripts adding movement, reusing the sprites, detect collisions and add UI behaviours. A **scoring system** must also be also included. All assets including graphics and sound assets are provided. You are not required to import/add any new assets.

Instructions :

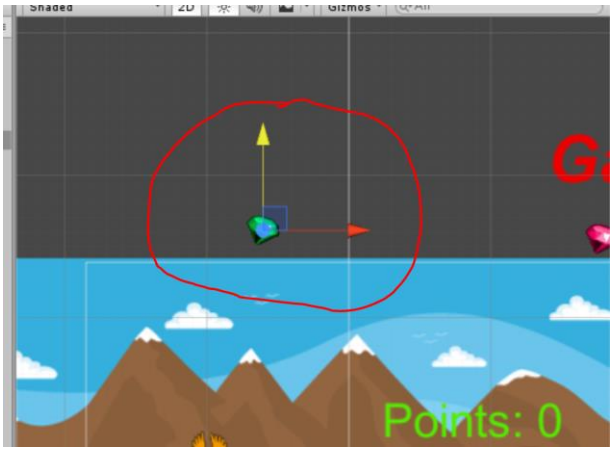
1. In each script add a following comment with your name and student ID. **NOTE: Scripts without such comment will not be graded!**

```
/*
 * Name: <First and Last name>
 * StudentId: <YourId>
 */
```

2. BackgroundController **(5 Marks)**
    a. Implement scrolling behaviour. It must scroll with a given speed set from Unity editor and do not "jump". (4 Marks: Implementation, 1 Marks: Functionality)

3. PlayerController **(5 Marks):**
    a. Player should move up and down with a given speed set from Unity editor (1 Marks).
    b. Player should be moved with arrows (up/down) and following keys (2 Marks)
        i. If your studentId ends with odd number use Q and A keys
        ii. If your studentId ends with even number use E and D keys
    c. Player cannot leave the scene (2 Marks)

4. GemController **(5 Marks):**
    a. Provide implementation for the Reset function that sets initial position for the sprite to the top of the screen beyond the view (2 Marks).



    b. Implement reuse functionality: reset when gem leaves the screen on the bottom or when gem is collected (2 Marks)

5. EnemyController **(6 Marks)**
    a. Provide implementation for the Reset function that sets initial position for the sprite. Sprite should appear in the random vertical position within the bounds of the screen. (2 Marks).
    b. Implement movement of the object. It should move with a given speed set in the Unity editor (1 Marks)
    c. Add a sound Collision that will be played when Enemy (jay) collides with the player (2 Marks)
    d. Add tag ENEMY to the object and save enemy as a prefab (1 Mark)

6. PlayerCollision **(8 Marks):**
    a. Implement collision detection function (see collider settings in the scene to decide which function you should implement) (2 Marks).
    b. Add points when colliding with Gem (2 Marks)
        i. if your studentID ends with even number, green gems add 10 points and red take 5 points
        ii. if your studentID ends with odd number, green gems add 5 points and red add 10 points
    c. Decrease number of lives when colliding with Jay (2 Marks)
    d. Log each collision to the console with your studentsID (2 Marks)

Format of the log:

```
<studentId> collision with Enemy

<studentId> collision with Gem
```

7. HUDController **(18 Marks):**
    a. Complete *initialize* function making sure that
        i. bird, jay and gems are visible and working (3 Marks)
        ii. game over label, score and button are not visible (3 Marks)
    b. Complete *endGame* function
        i. display information about score earned (1 Marks)
        ii. show *GameOver* and score label, and the button (3 Marks)
        iii. make sure that bird, jay and gems are not visible and do not interact (3 Marks)
    c. Complete *RestartButtonClick* function. It should restart the main level (2 Marks)
    d. Keep track of points and lives using a Player (singleton) class. Make sure that the class is a singleton (3 Marks)
8. Player (the singleton class) (3 Marks):
    a. Implement connection between Player and HUD that allows the Player instance to update the UI each time lives or points change (3 Marks)

SUBMITTING YOUR WORK

- Your submission should include a zip archive of your **complete** project folder.
- Please zip all files in to a single project archive.
- Your solution should be submitted on BlackBoard before the end of the exam.
- Late submissions WILL NOT BE ACCEPTED!

This midterm is weighted **20%** of your total mark for this course.

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. You have to type all the code. Do not copy-paste the code!
3. You must understand any code you use and include documentation (comments) around the code that explains its function.