WESTERN UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SE2250b – Software Construction

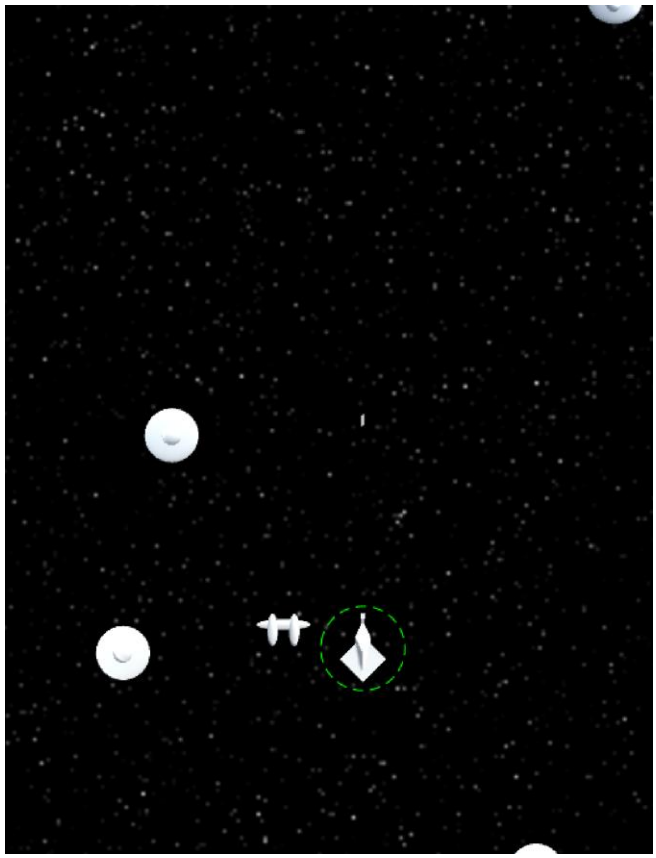
Assignment 2: Space SHUMP Game

Deadlines:

Section 002: 10:30 am, Tuesday, March 3rd, 2020
Section 003: 10:30 am, Monday, March 2nd, 2020

NOTE: Proper coding practices must be used. Examples include but are not limited to the following: naming conventions, project organization, proper code formatting, meaningful comments, clean code (no old commented out code or unnecessary code), and any other topics discussed in lectures or labs.

In this assignment you will create a very simplified space SHUMP (Shoot 'em up) game. In this game, the player character is a spaceship. Enemies move in different ways and destroy the ship on collision. The assignment is based on the Prototype 3 Space SHUMP from Bond's book. This book chapter is available in OWL under Lectures. The solution should only contain features defined in this document, and no other features described in the book.



WESTERN UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

1. (5 points) Basic ship setup
   - Create a new 2D project and import packages from Bond's book at /book/prototools.net, Part III Game Prototype Examples and Tutorials/Chapter 30: Space SHUMP. Download and import Starter Package C30_Space_SHMUP_Starter.unitypackage.
   - Set the Scene. Use an orthographical camera, aspect ratio 3:4, and directional light. You can use similar settings as in the book.
   - Make the Ship. You can use the same look as in the book without the shield.
   - Move the ship using arrow keys and WASD keys.
     To make the movement more dynamic, implement a slight rotation of the ship (similar to the example in the book).
     The ship should always remain on the screen. The script attached to the ship should be responsible for keeping the ship on the screen – name this script BoundsCheck.

2. (6 points) Enemies
   - Create two game objects to represent enemies: Enemy_1 and Enemy_2.
     They must look different than enemies in the book. Each enemy must be modeled from several shapes.
   - Move enemies
     Enemy_1 should move straight down.
     Enemy_2 should move on a 45° line randomly choosing left or right.
     Make sure you use inheritance and put all code common for all enemies into the superclass.

3. (6 points) Spawn and destroy enemies
   - Spawn enemies. All enemies should be instantiated from the same script and all should be instantiated just above the screen. Name the script Main and attach it to the main camera. A new enemy should appear every 2 or 3 seconds (spawning does not stop) and the choice between Enemy_1 and Enemy_2 must be random.
   - Destroy enemies when they go off the screen.

     You already have the code that keeps the ship on the screen. The same validations will need to be done to check if the enemy is on the screen. Therefore, change the existing BoundsCheck script to do location checking for both the ship and enemies. BoundsCheck will be attached to the ship and all enemies.

     To see boundaries of the camera view in the scene pane use `OnDrawGizmos` as below. To see the boundary when the game is running, the component needs to be selected. This will allow for checking that enemies are destroyed as soon as they exit the camera view.

```
void OnDrawGizmos () {
      if (!Application.isPlaying) return;
      Vector3 boundSize = new Vector3(camWidth* 2, camHeight* 2, 0.1f);
      Gizmos.DrawWireCube(Vector3.zero, boundSize);
}
```

4. (3 points) Make the enemies destroy the player
   - When an enemy collides with the player, both, the enemy and the ship should be destroyed.

Deliverables:

- Functionality demonstrated to a TA during the lab. Make sure you can explain your implementation of the game. This is a part of the mark.
- All game files uploaded to OWL as one zip file. This should contain all the files needed to run your game from UNITY.

If the student did not demo the solution to TA during the lab section on the due date or before because of reasonable justification (illness or other special circumstances), the student must contact the professor within the two days following the deadline to arrange the time for the demo. For no demo, 50% of the available mark will be deducted.