Requirements:

- First, recreate UFO 2D game as described in Lecture 1.
  - Create a New Project named *hw1* and select Template as 2D.
  - The 3 sprites (and a font file) needed for the game are available on Canvas (`ufo2d.zip`). Import them into your game.
  - Follow the instructions in Lecture 1 to recreate the entire UFO 2D game (Fig. 1).
- Then, we'll make some modifications to the game as follows:
  - For moving Player (UFO), do not use *AddForce* method. Instead, use *velocity* (see below).
  - Now, the Pickup objects also move (in random directions at a slow pace).
  - Pickup objects must also bounce off the walls, just like Player object.
  - The object of the game is now to avoid contact from any Pickup object for 60 seconds.
  - Once Player is hit by any Pickup object, it's Game Over (display *Game Over* on screen). Use *BradBunR.ttf* for displaying all UI texts.
  - Instead of showing *Count* number on UI, display *Timer* that counts down from 60 to 0 (must be updated every second).
  - If the Player survives without getting hit for 60 seconds, display *You Win!*
  - As soon as the timer hits 0, the timer must stop.
  - After the game is over (win or lose), object collisions may still happen.
  - For this new game, 12 Pickup objects might be too many to handle, so reduce them as you see fit (depends on the speed of Pickup).
  - Make sure to add *Restart* button to let user restart once the game is over. (*Missing restart button will lead to point deduction*)
- Add one more twist of your own to the game, and be creative.
  - For example, you may introduce a special Pickup object that you CAN hit, and when you do, you are allowed to devour any Pickup objects for 5 seconds... as in Pac-Man. But do not use this Pac-Man twist, use your own. :-)

- Finally, build your game as follows:
  - First, create *Build* folder under your Unity project directory.
  - Select *Windows* as your target platform (even if you're using Mac).
  - Click *Build* and select Build folder you just created.
  - The build file (.exe) must be created under Build folder. (*Missing build file will incur point deduction*)

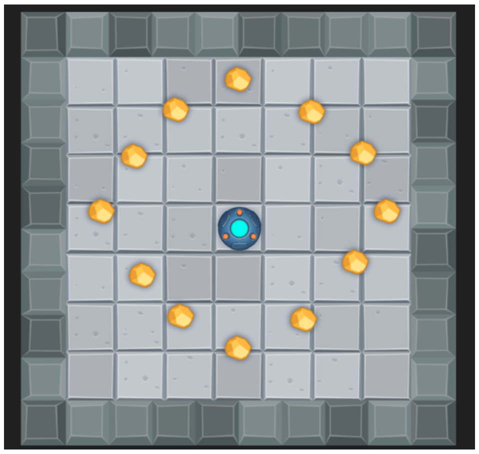

Figure 1: UFO 2D game

How to count seconds:

```
float timer = 0.0f;
void Update()
{
    timer += Time.deltaTime;
    int seconds = (int)timer % 60;
}
```

How to generate random number between -10.0f and 10.0f:

```
Random.Range(-10.0f, 10.0f)
```

How to make Player movement easier to control:

Replace

```
rb2d.AddForce(movement * speed);
```

with

```
rb2d.velocity = movement * speed;
```

How to detect a collision that does not involve "Trigger" collider:

Use

```
void OnCollisionEnter2D(collision2D col)
```

with

```
void OnTriggerEnter2D(collider2D col)
```

What to submit:

- *Build* folder of your Unity project. Must be zipped and submitted on Canvas.
- Your entire Unity Project directory (containing all assets and build files). Must be submitted to your public GitHub repository.
- Check out `unity_github.pptx` posted on Canvas for instructions on how to upload your Unity project to public GitHub repository.

How to submit:

- Make sure to zip your *Build* folder into `hw1.zip`, then submit your `hw1.zip` as a single file on Canvas.
- After submission, click "Add Comment" on Canvas to describe what special twist you have added to the gameplay. (*Missing description will incur point deduction*)
- Also, "Add Comment" to specify your GitHub repository URL. (*Missing URL will incur point deduction*)

Policy

- At the top of each Script file (.cs), provide comments specifying the author, date, and a brief description of the file.
- Each Script file (.cs) must contain enough comments here and there to make it easy to follow your code. Insufficient comments could incur point deduction.

- Incomplete project will get almost no credit (e.g., game does not run due to script errors or game terminates prematurely due to run-time errors).
- *Thou shall not covet thy neighbor's code*. If identical (or nearly identical) submissions are found among students, every student involved will get automatic zero for the assignment. The same goes for copying existing code from online source.
- If a student makes multiple submissions, only the last submission will be considered valid.