

# Advanced Code Challenge

...

Let's go!  
So much fun!



# Create a script called *Spawner*

Spawner script **level 1**

- Have the spawner instantiate prefab -GameObjects every frame.  
A standard Cube or some other primitive is fine.

# *Spawner / prepare to instantiate by the manager*

## Spawner script **level 2**

- Move the instantiating to a separate method that is public. Let's call it *Spawn*.
- Create a variable like 'numberSpawned' that tracks the amount of objects instantiated.

Point is to be able to call instantiating from another script.

# Create a script called *SpawnManager*

SpawnManager script **level 1**

- Have a reference to your **Spawner** and call its *Spawn* -method in every update.

The point is to move the spawning logics from Spawner to SpawnManager.

# *SpawnManager / spawn timer*

SpawnManager script **level 2**

- Create a timer for spawning, where you can set the delay between spawns in seconds. You could start with half a second.

Point is to be able to control spawning speed from inspector and not rely on the fps of the machine.

## *Spawner / spawn position and parent*

Spawner script **level 3**

- Make the spawned prefabs instantiate where this spawner GameObject's position is. Make sure it's on a separate object from the SpawnManager.
- Make the spawned prefabs childs of this GameObject.

The point is to have spawned objects appear where this GameObject is.

# *Spawner / make spawned objects jump*

Spawner script **level 4**

- Add a Rigidbody to your prefab -GameObject.
- When this GameObject is spawned, tell the Rigidbody to jump by something like:  

```
AddForce(Vector3.up * jumpForce, ForceMode.Impulse)
```
- You might want to add a plane or some kind of floor at this point.

The point is to have fun.

# *SpawnManager / support multiple spawners*

SpawnManager script **level 3**

- Prepare your **SpawnManager** for controlling multiple spawners: change the reference to your **Spawner** into a list of **Spawners**.
- Change your **Spawn** -calls to call on every spawner. (hint: use loop)
- Make sure your spawning still works, you can drag your **Spawner** -script manually.

Point is to support multiple spawners.



## *Spawner / random chance*

Spawner script **level 5**

- Add a random chance of 20% for Spawners to instantiate.

The point is to add randomness.

# *SpawnManager / Singleton*

SpawnManager script **level 4**

- Create a singleton -reference to the SpawnManager in the script (hint: static).

The point is to find the manager with any script in the scene.

# *SpawnManager / subscribe 1*

SpawnManager script **level 5**

- Create a Subscribe -method in the script that accepts a **Spawner** as a parameter and then adds the GameObject to the list of **Spawners**.
- Clear the list of manually assigned Spawners. You may need to initialize the list in Awake with `spawners = new List<Spawner>()`

Prepare to accept subscriptions from managed scripts.

## *Spawner / subscribe 2*

Spawner script **level 6**

- Make this **Spawner** subscribe to **SpawnManager** automatically. Make sure this is after the manager initializes its singleton.
- Duplicate and scatter 6 **Spawner** -GameObjects around your scene.
- Make sure they all start spawning without you having to assign them on the **SpawnManager** list that should be empty now.

Complete the subscription from this end.

# SpawnManager / spawn cap

SpawnManager script **level 6**

- Make your **Spawner** / 'numberSpawned' a static member so every spawn by any spawner is recorded.
- Create a variable for **SpawnManager** that controls the maximum amount of spawned objects. When it's reached (hint: read from **Spawner**), no more is spawned.

Limit your spawned objects.





## *Spawner / random color*

Spawner script **level 7**

- Randomize your spawned object's color.

It's fun!



# *SpawnManager / events*

SpawnManager script **level 7**

- Refactor your code to use events (like Actions) instead of subscribing to list.

Events

# UI

## UI script **level 3**

- Make the binding happen in runtime (subscribe to event or think of another way)

Automatic binding FTW

# YOU'RE DONE! GRADING

**DISCLAIMER:**  
I reserve the right to change the  
formula after the results come in.

Your grade is `Mathf.Clamp(levels/2 -2, 0f, 5f)`

So count every “level” you could complete, divide by 2 and -2.

Max score is 6,5.

If you didn't get over 0, you didn't pass and need to try again later.