Tutorial; Enemy spawn and movement

Setup

Set up an empty game object as a place holder for the enemy spawn script. This is necessary as these objects are going to be randomly generated and cannot be attached to an object within the game that already exists. Create the script “spawning” and add it to the empty game object in the hierarchy. Also, set up walls along the plane so the cubes can be contained inside the arena set out and create a collider for each of these walls.

Coding enemy spawn

There will be about 5 enemies spawned into the game, although you can adjust this accordingly if you want. To generate cubes into the game, you can do so by calling upon it in the script using this line;

```
GameObject cube = GameObject.CreatePrimitive(PrimitiveType.Cube);
```

In order to call upon these cubes we need a value between 0 and 5 to call upon so the game knows how many to spawn in;

```
for (int i = 0; i < 5; i++)
```

```
int x = Random.Range(-70, 70);
int z = Random.Range(-70, 70);
```

The purpose of this it to spread the cubes across the terrain randomly so they don’t collate in the same location. This allows the player to spread more easily across the terrain and given the correct amount of space needed between the player and the enemy.

A Rigidbody is needed so the cubes will bounce off the wall when the colliders interact. You’ll also need to add Rigidbody constrains so that the cubes don’t end up flying off the plane. In the script set up it should look like this (including the rigid body);

```
{
    // Start is called before the first frame update
    void Start()
    {
        //Generate 5 cubes
        for (int i = 0; i < 5; i++)
        {
            GameObject cube = GameObject.CreatePrimitive(PrimitiveType.Cube);

            int x = Random.Range(-70, 70);
            int z = Random.Range(-70, 70);

            cube.transform.position = new Vector3(x, 0, z);
            cube.transform.parent = transform;

            cube.transform.localScale = new Vector3(5, 5, 5);
            Rigidbody rb = cube.AddComponent<Rigidbody>();
            rb.constraints = RigidbodyConstraints.FreezeRotationZ;
        }

}
```

Coding enemy movement

Now we have the correct positions set up, we can start moving them around the map, which is called upon in the void update. Add this script to the update;

```
{

        foreach(Transform child in transform)
        {
            if (child.name == "Cube")
            {
                //move forward
                child.transform.Translate( Random.Range(0, 0.5f) * transform.forward
);

                //turn every now and then
                if( Random.Range(0,100) == 50)
                {
                    int y = Random.Range(0, 360);
                    child.transform.eulerAngles = new Vector3(0, y, 0);
                }


            }
        }

    }
```

The foreach searches for every item in the hierarchy and when it comes across the cube asset, it moves it forward using child.transform on a random range. The eulerAngles are what causes the object to rotate and the new Vector3 is the new range in which the cubes travel along the Y axis.

Changing spawning enemy size

Changing the enemy size was already mentioned in the cube generation list, but if you wish to change the size of the cubes then refer to the line;

```
cube.transform.localScale = new Vector3(5, 5, 5);
```

the coordinates in this instance are the key to changing the shape size along the x, y and z axis. For example, if you wanted your cube to be size 10 you’d simply change it in the script so the new Vector 3 says (10,10,10).