# Assignment # 2

**Fall 2024**

**CSE-411 Intro to Game Development**

Submitted by: **Ali Asghar**

Registration No.: **21PWCSE2059**

Class Section: **A**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Submitted to:

**Engr. Abdullah Hamid**

Date:

20th **November 2024**

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

## Task:

As discussed in class, create a new scene with a plane the camera should be top down over the plane. You must instantiate enemy cubes on the plane on random location (they must not appear out of the plane area). Also they must destroy themselves after 2 seconds.

## Solution 1:

### Theory:

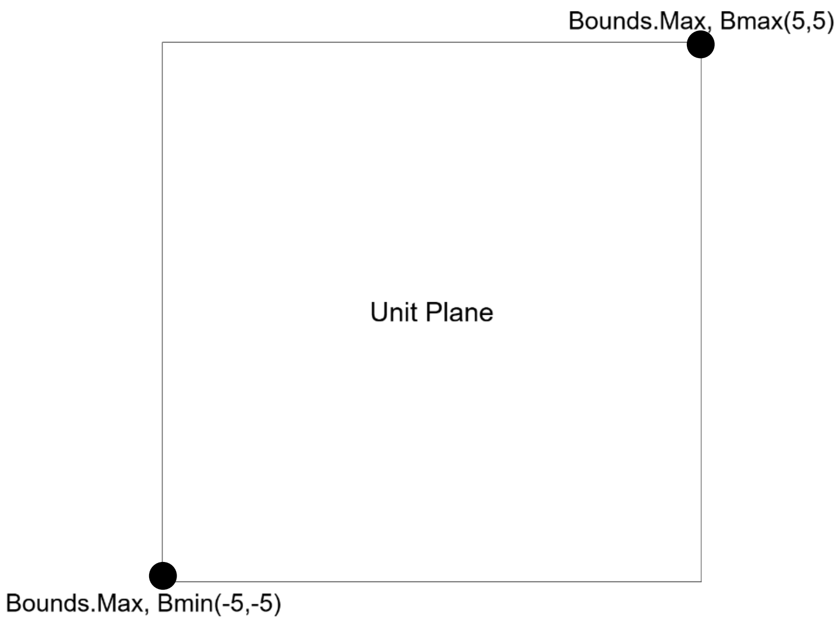
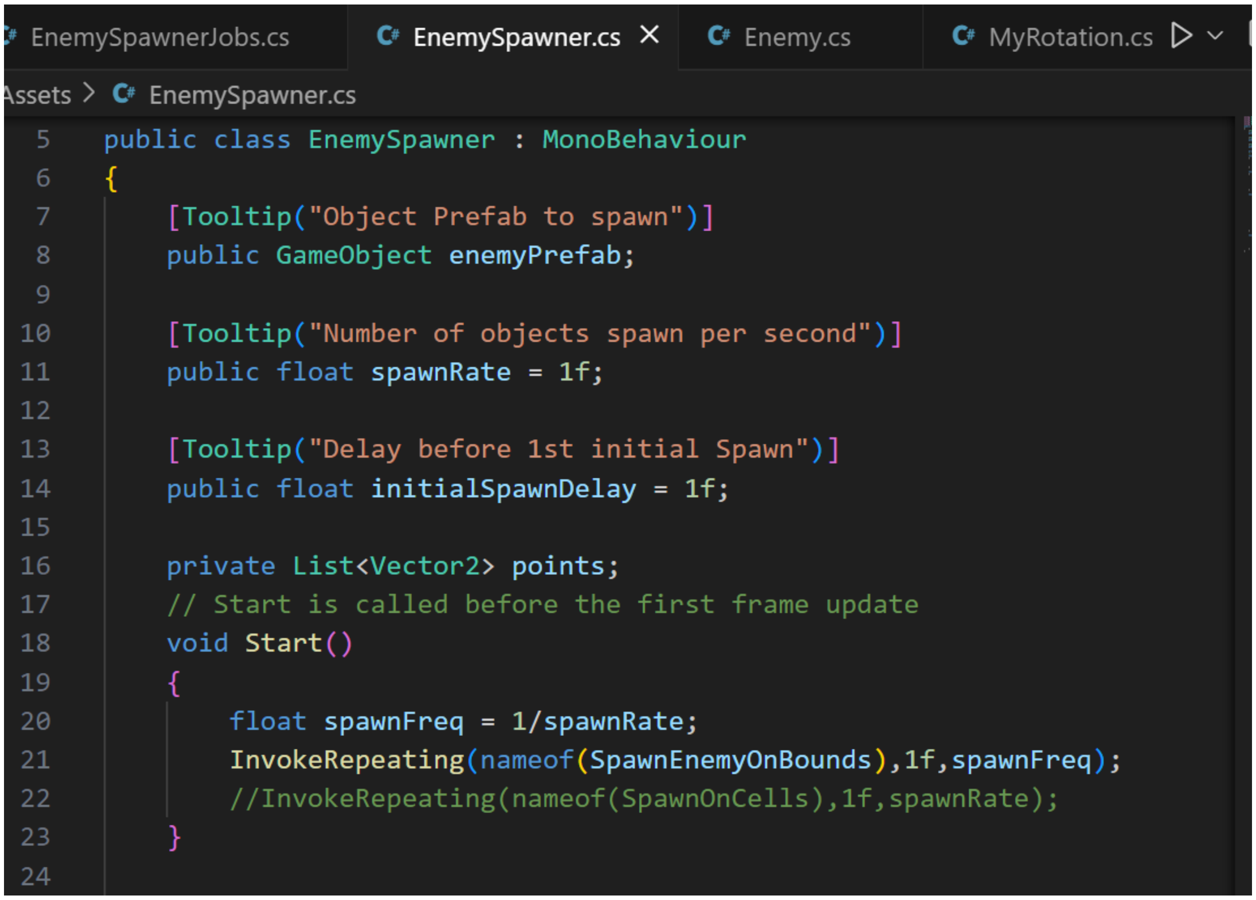


*Figure 1: Bounds of Unit Plane in Unity*

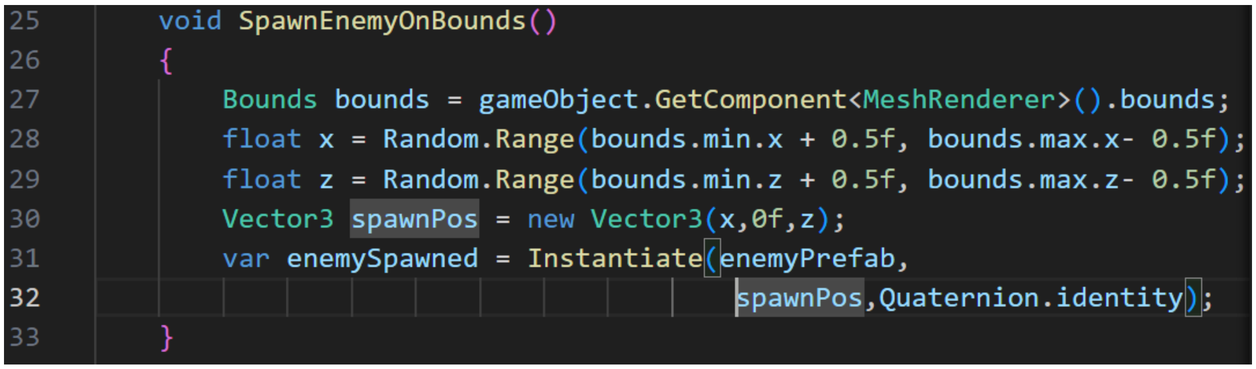
In figure 1, we can see a unit plane in Unity's 3D world space. To get the total area of plane, we can get the bounds using the MeshRenderer component of plane. Now to spawn enemy cube inside this plane's area, we need to find a random point inside this plane area using the minimum and maximum vector of bounds. For safety reasons, I have also added an offset of 0.5 to avoid instantiating the enemy cube outside the plane. This solution gets continuous random positions of the plane.

**Code:**

**EnemySpawner class**

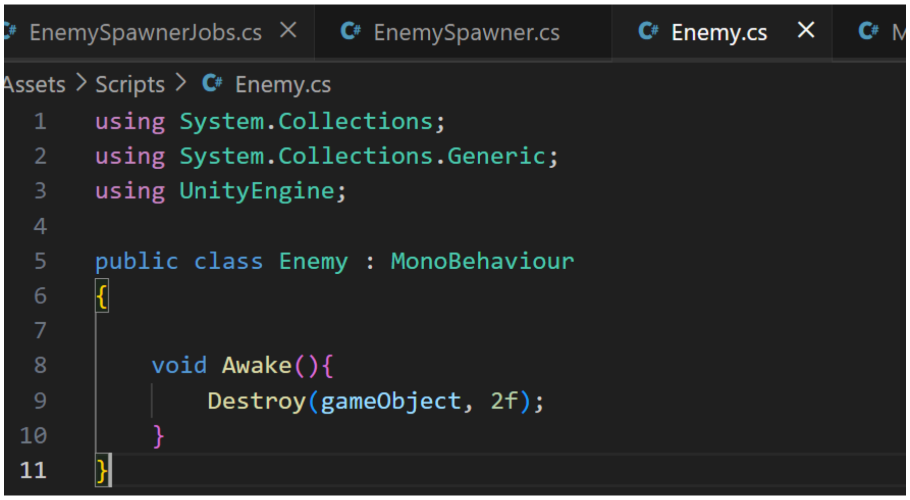


```csharp
public class EnemySpawner : MonoBehaviour
{
    [Tooltip("Object Prefab to spawn")]
    public GameObject enemyPrefab;

    [Tooltip("Number of objects spawn per second")]
    public float spawnRate = 1f;

    [Tooltip("Delay before 1st initial Spawn")]
    public float initialSpawnDelay = 1f;

    private List<Vector2> points;
    // Start is called before the first frame update
    void Start()
    {
        float spawnFreq = 1/spawnRate;
        InvokeRepeating(nameof(SpawnEnemyOnBounds),1f,spawnFreq);
        //InvokeRepeating(nameof(SpawnOnCells),1f,spawnRate);
    }

```



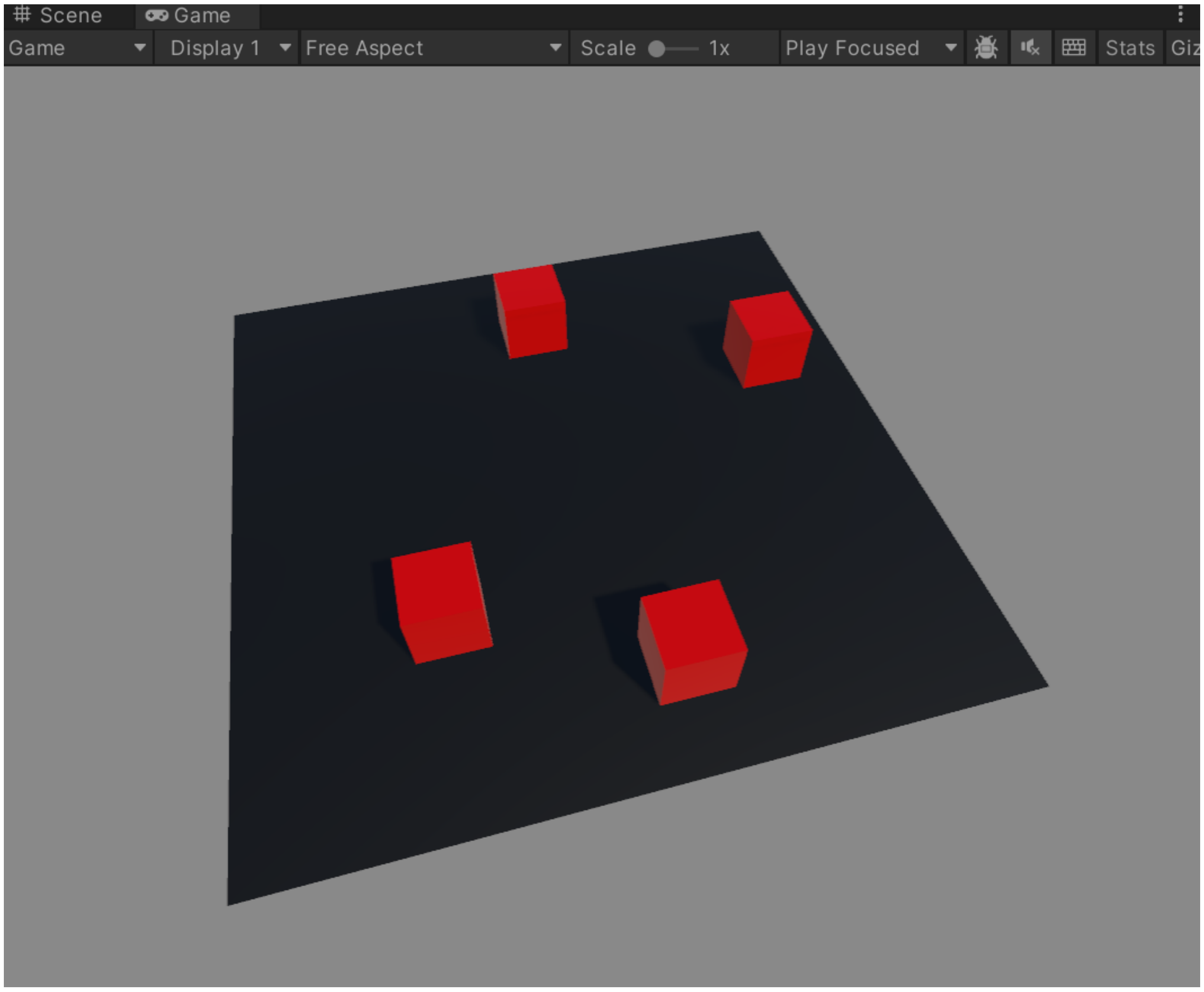
```csharp
    void SpawnEnemyOnBounds()
    {
        Bounds bounds = gameObject.GetComponent<MeshRenderer>().bounds;
        float x = Random.Range(bounds.min.x + 0.5f, bounds.max.x- 0.5f);
        float z = Random.Range(bounds.min.z + 0.5f, bounds.max.z- 0.5f);
        Vector3 spawnPos = new Vector3(x,0f,z);
        var enemySpawned = Instantiate(enemyPrefab,
                                        spawnPos,Quaternion.identity);
    }
```

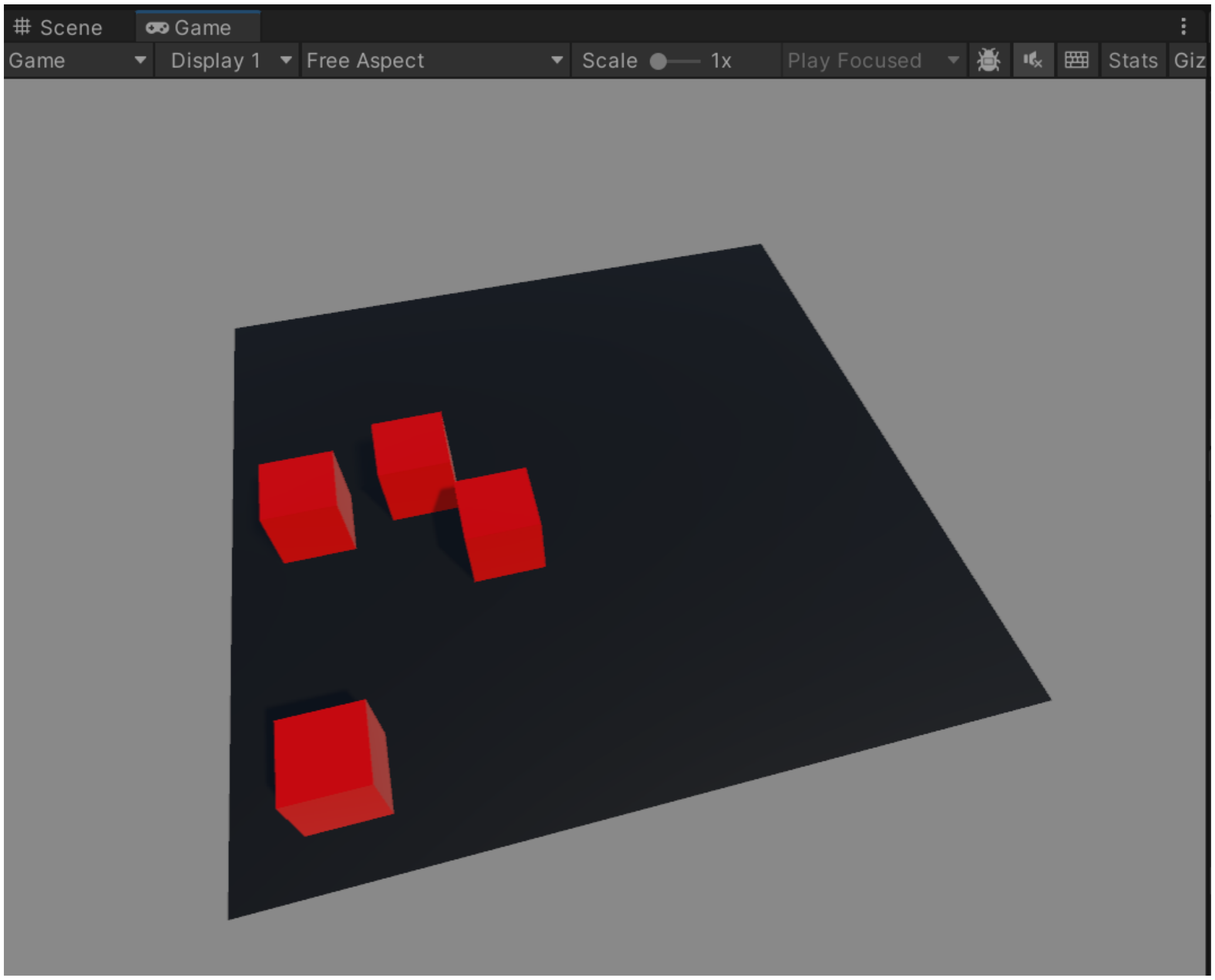
**Enemy class**



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Enemy : MonoBehaviour
{

    void Awake(){
        Destroy(gameObject, 2f);
    }
}
```

**Output:**




Scene
Game
Game
Display 1
Free Aspect
Scale
1x
Play Focused
Stats
Giz

## Solution 2:

### Theory:

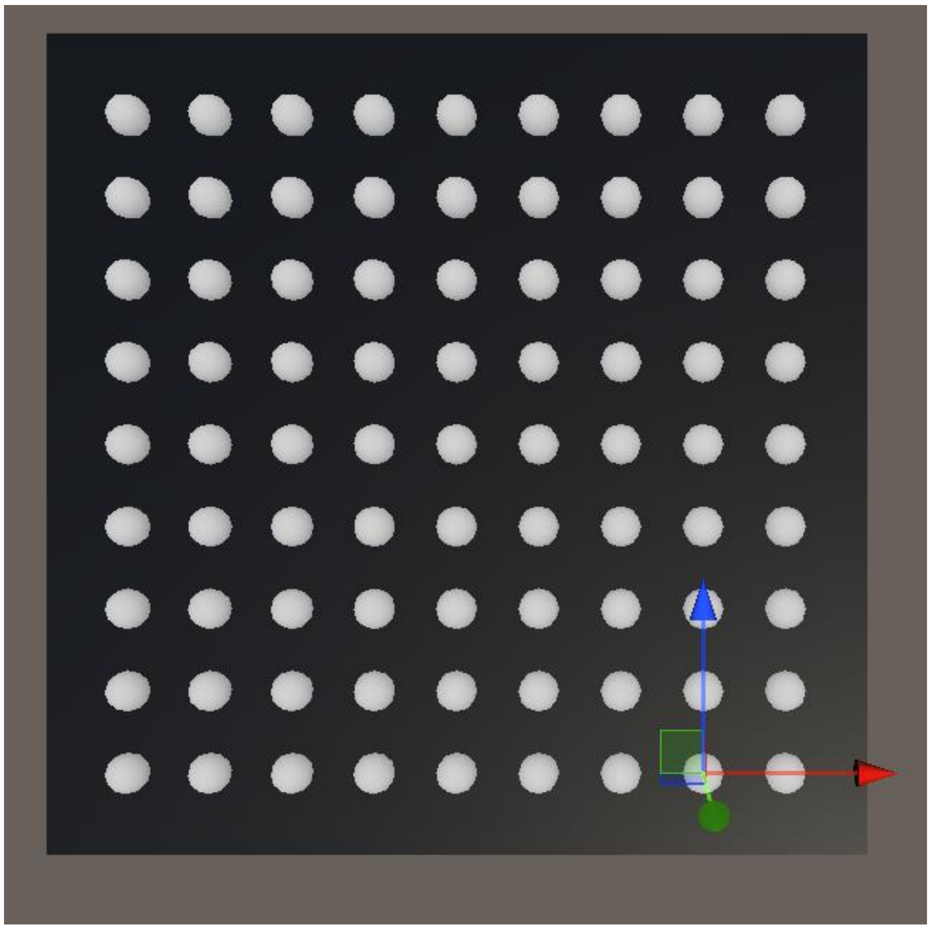


*Figure 2: Discrete Points inside Unit Plane in Unity*

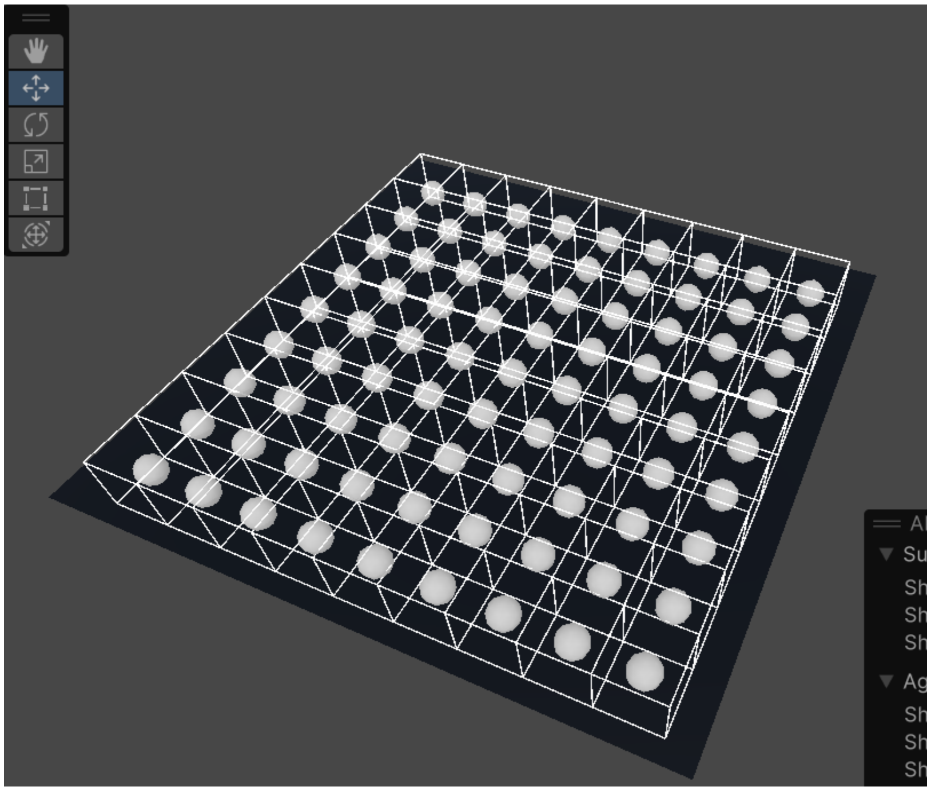
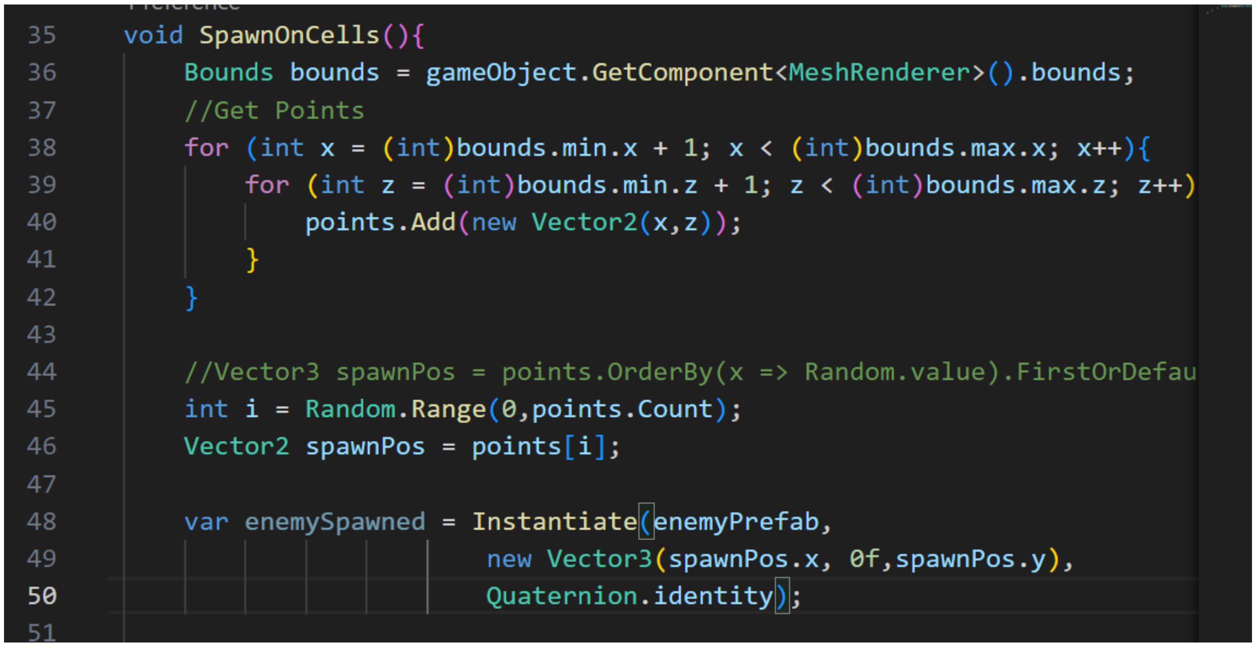


Figure 3: Discrete Cells inside Unit Plane in Unity

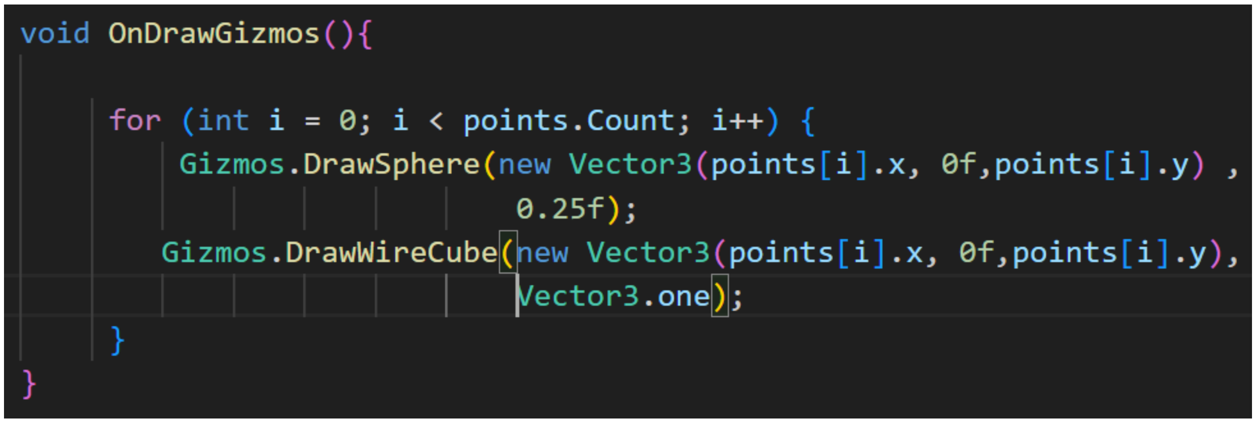
In figure 2, the concept is almost same but, in this solution, I've discretized the random points inside the plane. I have first saved all the points in a vector2 list and then selected a random point from this list for instantiating the enemy cube. These discrete points inside the plane bounds/area can be referred to as cells. For visualization, I have used the OnDrawGizmos function available in Unity for visual debugging. In figure 3, I have also draw the cells around each point.

**Code:**

**EnemySpawner class**



```
    void SpawnOnCells(){
        Bounds bounds = gameObject.GetComponent<MeshRenderer>().bounds;
        //Get Points
        for (int x = (int)bounds.min.x + 1; x < (int)bounds.max.x; x++){
            for (int z = (int)bounds.min.z + 1; z < (int)bounds.max.z; z++)
                points.Add(new Vector2(x,z));
            }
        }

        //Vector3 spawnPos = points.OrderBy(x => Random.value).FirstOrDefau
        int i = Random.Range(0,points.Count);
        Vector2 spawnPos = points[i];

        var enemySpawned = Instantiate(enemyPrefab,
                                    new Vector3(spawnPos.x, 0f,spawnPos.y),
                                    Quaternion.identity);
```

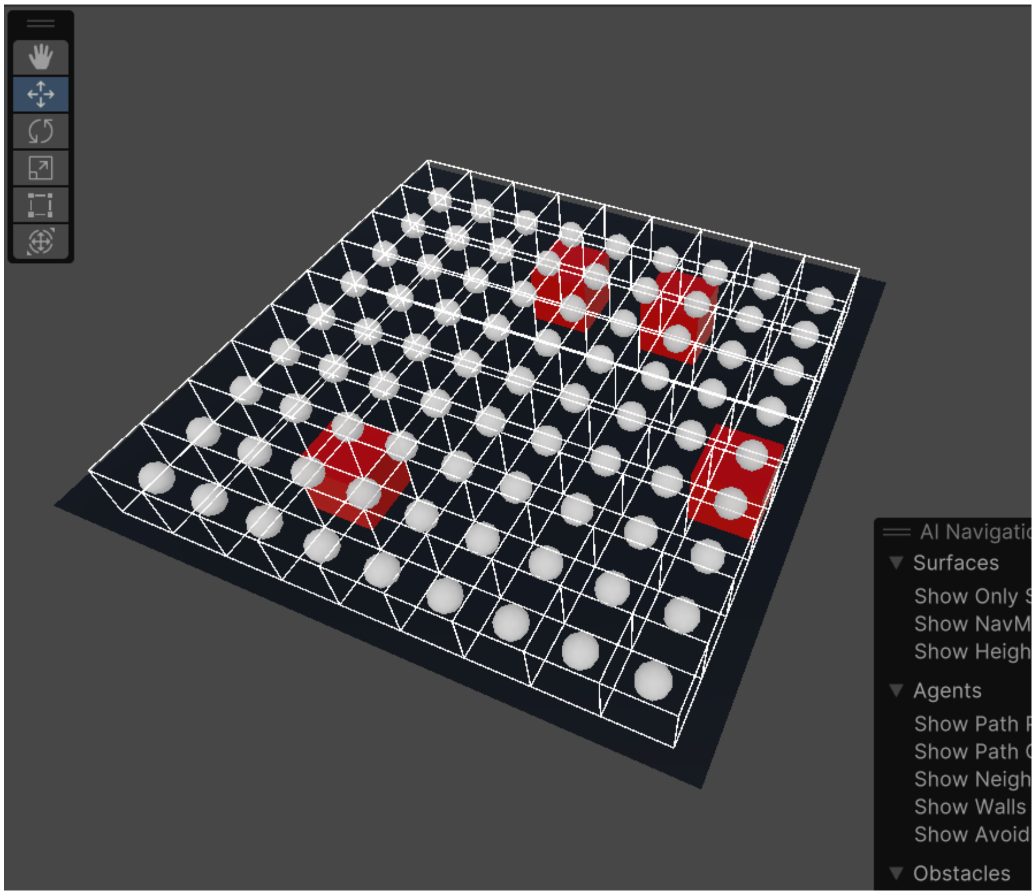


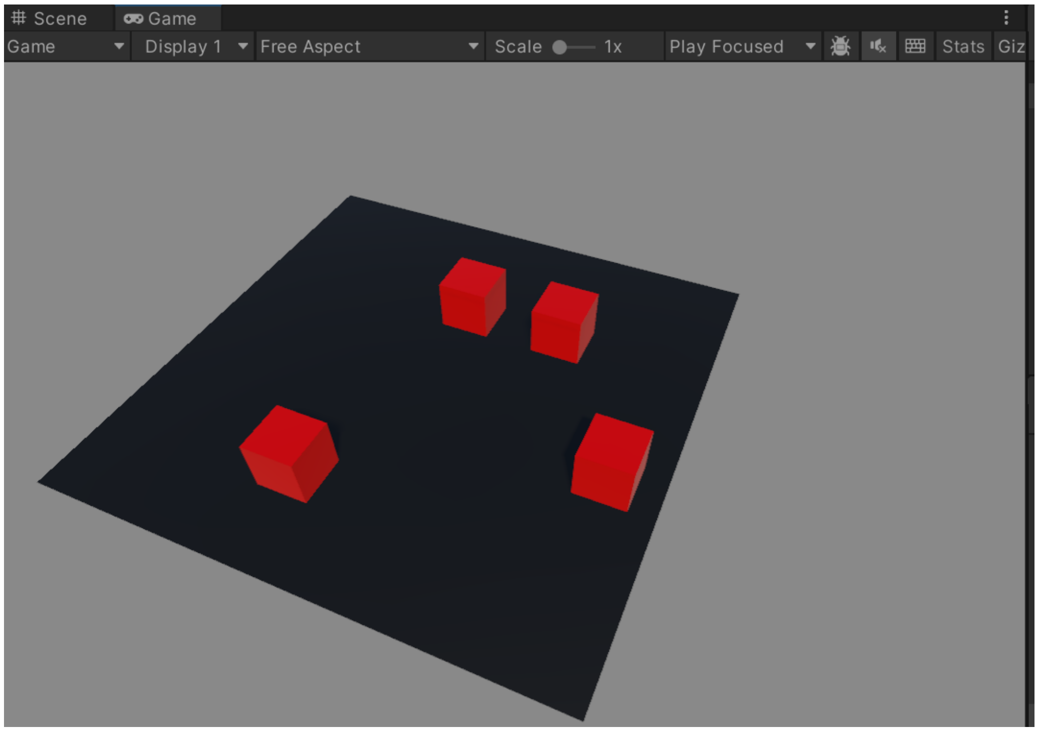
```
void OnDrawGizmos(){

    for (int i = 0; i < points.Count; i++) {
        Gizmos.DrawSphere(new Vector3(points[i].x, 0f,points[i].y) ,
                                0.25f);
       Gizmos.DrawWireCube(new Vector3(points[i].x, 0f,points[i].y),
                                Vector3.one);
    }
}
```

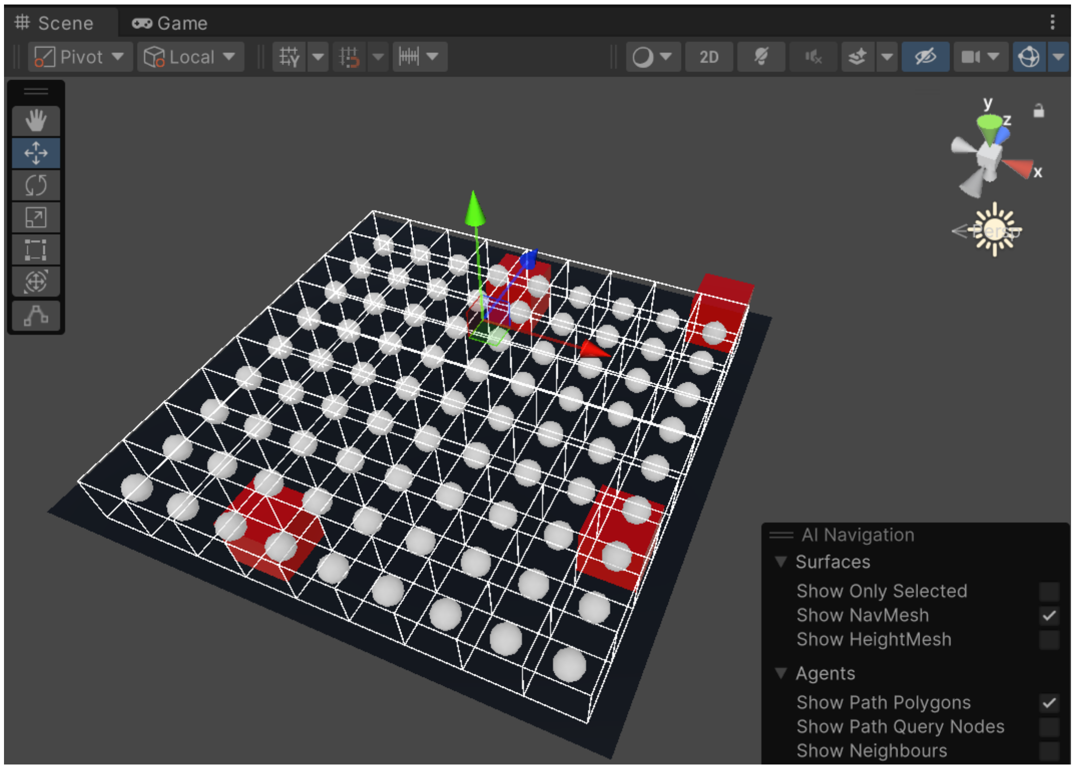
**Enemy class**

This class remains the same as before.

**Output:**


AI Navigatic
Surfaces
Show Only S
Show NavM
Show Heigh
Agents
Show Path P
Show Path C
Show Neigh
Show Walls
Show Avoid
Obstacles


Scene
Game
Game
Display 1
Free Aspect
Scale
1x
Play Focused
Stats
Giz



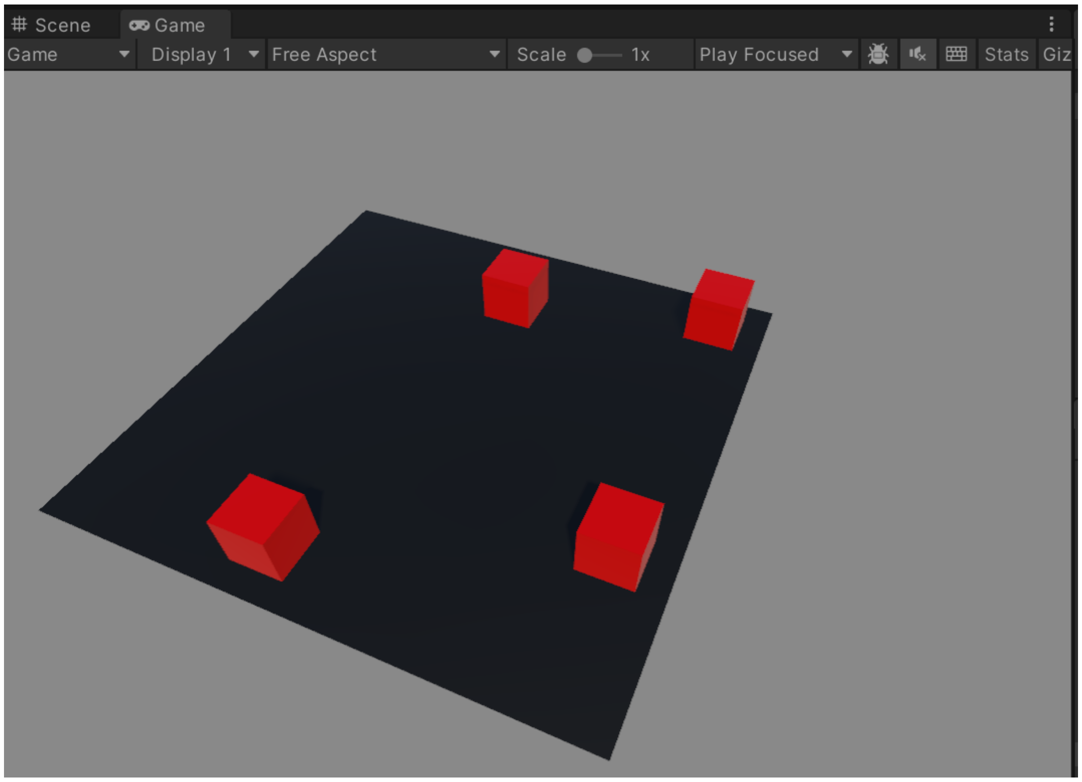
AI Navigation

Surfaces

Show Only Selected
Show NavMesh ✓
Show HeightMesh

Agents

Show Path Polygons ✓
Show Path Query Nodes
Show Neighbours


Scene
Game
Game
Display 1
Free Aspect
Scale
1x
Play Focused
Stats
Giz