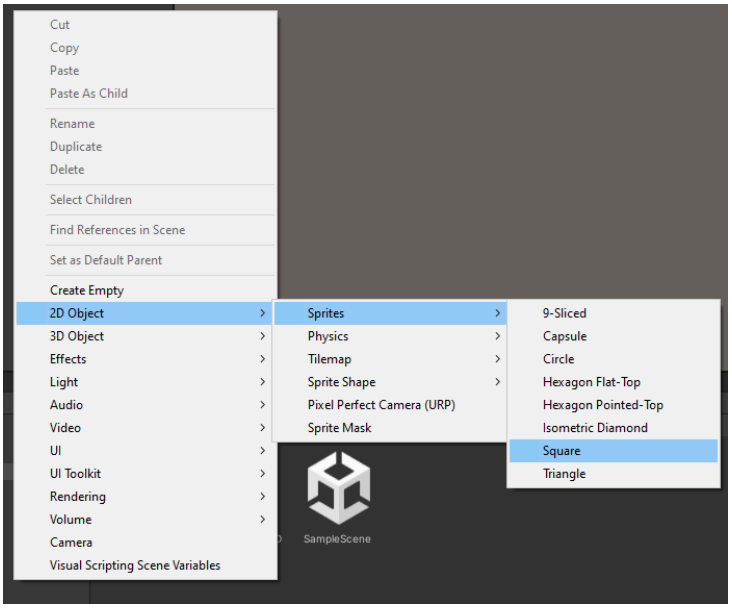
1. Instantiate a 5 x 5 grid of squares using sprites and C# Code

1.1 Project Setup and Script

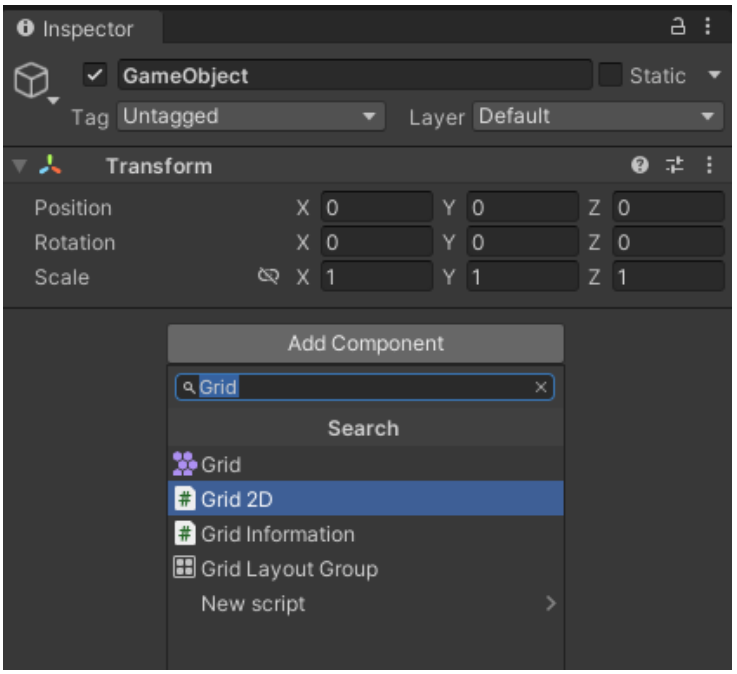
1. Create a new **2D Core** project
2. Create a new **square** sprite from the hierarchy window.
3. 

4. Create a new c# script called **Grid2D**. Add the following code to the script.

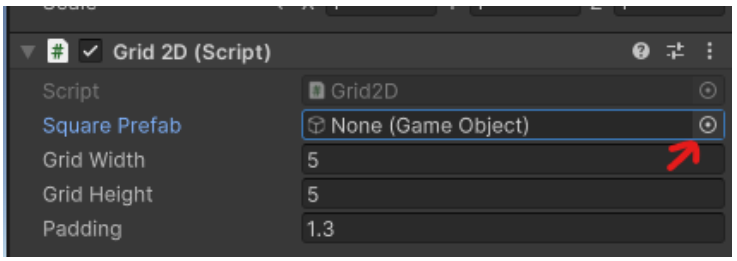
```C/C++
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Grid2D : MonoBehaviour
{
  public GameObject squarePrefab; // Assign your square sprite in the
inspector
  public int gridWidth = 5;
  public int gridHeight = 5;
  public float padding = 1.3f; // Space between sprites

  void Start()
  {
    SpawnGrid();
  }

  void SpawnGrid()
  {
    for (int x = 0; x < gridWidth; x++)
    {
```

```
      for (int y = 0; y < gridHeight; y++)
      {
        // Calculate position for each sprite in the grid
        Vector2 spawnPosition = new Vector2(x * padding, y * padding);
        // Instantiate the sprite at the calculated position
        Instantiate(squarePrefab, spawnPosition, Quaternion.identity,
transform);
      }
    }
  }
}
```

1.2 Spawning

1. Create an empty gameobject.
2. Set transform position to (0,0,0).
3. Attach the **Grid2D** script to this empty gameobject. You can use the add component button in the inspector to add the script by searching its name or you can directly drag the C# script from the Project window to the inspector of the empty gameobject.
4. 

5. Once script is added click on the bullseye icon in front of the Square Prefab field in inspector.

6. 

7. Choose Square from the list that appears.
8. Hit play. You will notice a grid of 5 x 5 squares appear.
9. Grid size can be adjusted by tweaking the Grid Width and Grid Height fields in the script or in the editor itself.

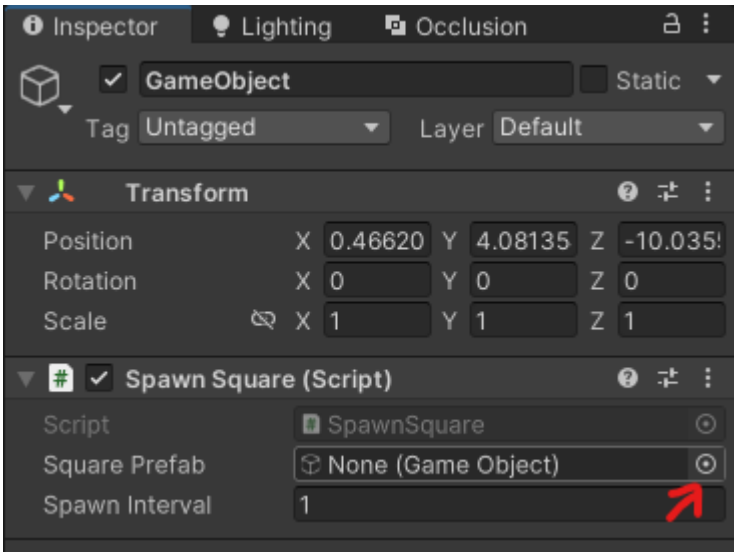
2. Randomly instantiate sprites and disappear after a specific amount of time

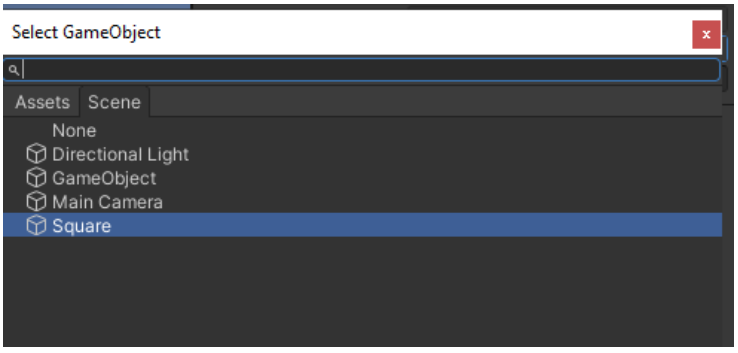
2.1 Project Setup and Script

1. Create a new 2D Core project
2. Create a new square sprite from the hierarchy window.
3. Create a new C# script called SpawnSquare. Add the following code to the script.

```C/C++
using System.Collections;
using UnityEngine;

public class SpawnSquare : MonoBehaviour
{
  public GameObject squarePrefab;
  public float spawnInterval = 2.0f;

  void Start()
  {
       squarePrefab.transform.position = new Vector2(100, 100);
    StartCoroutine(SpawnAndDestroy());
  }

  IEnumerator SpawnAndDestroy()
  {
    while (true)
    {
      // Spawn a square at a random position within the camera's view
      Vector2 spawnPosition = new Vector2(Random.Range(-8.0f, 8.0f),
Random.Range(-4.0f, 4.0f));
      GameObject square = Instantiate(squarePrefab, spawnPosition,
Quaternion.identity);
```

```
        // Wait for the specified interval, then destroy the square
        yield return new WaitForSeconds(spawnInterval);
        Destroy(square);
    }
  }
}
```

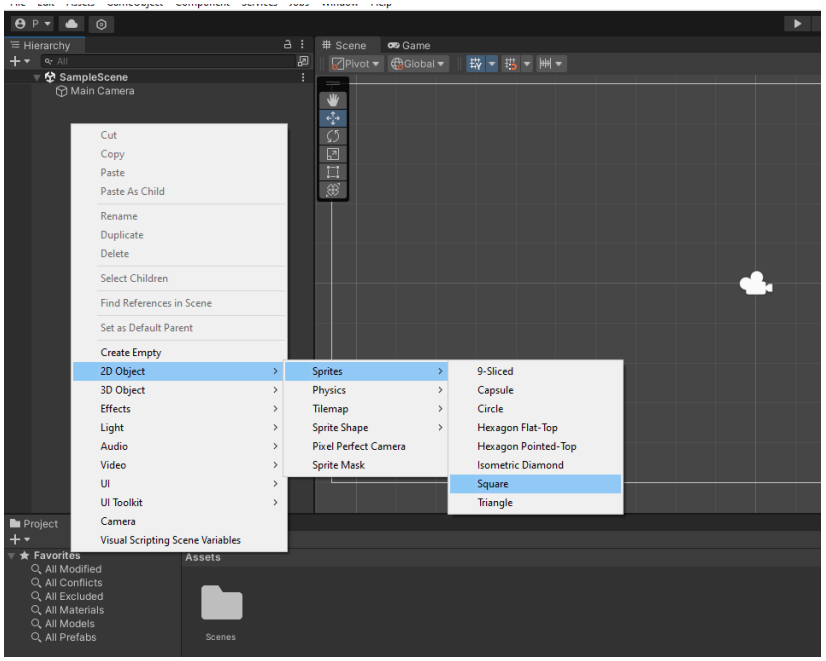
4. A Coroutine is used here compared to the standard Update loop. Read about it here.
5. Create a new empty gameobject.
6. Attach the SpawnSquare script to this empty gameobject.
7. 

8. Click this bullseye icon.
9. Choose the square gameobject from the list.
10. 

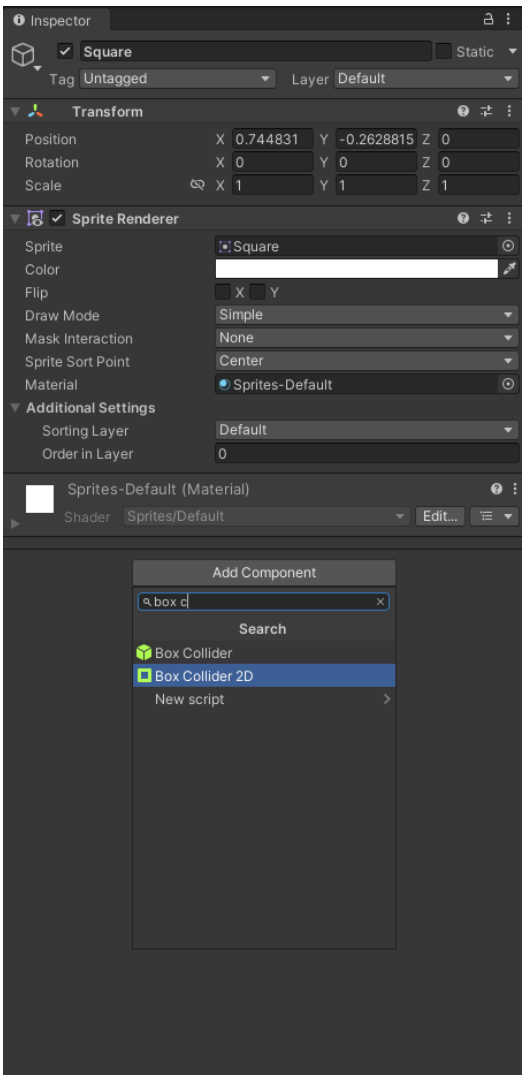
11. Hit play, you will notice that the square gets instantiated at random positions and then gets destroyed. The interval can be changed by changing the value in spawnInterval variable.

3. Square should change its color when clicked. Colors should cycle through the VIBGYOR colors.

3.1 Setup Square Sprite

1. Create a new 2D Core Project.
2. With the SampleScene open, right click on the hierarchy window > 2D Object > Sprites > Square. This will create a square sprite in your scene.
    a. 

3. Next, click on the square sprite, and then click on the add component button in the inspector window.
4. Search for “Box Collider 2D” and click on the result that appears.

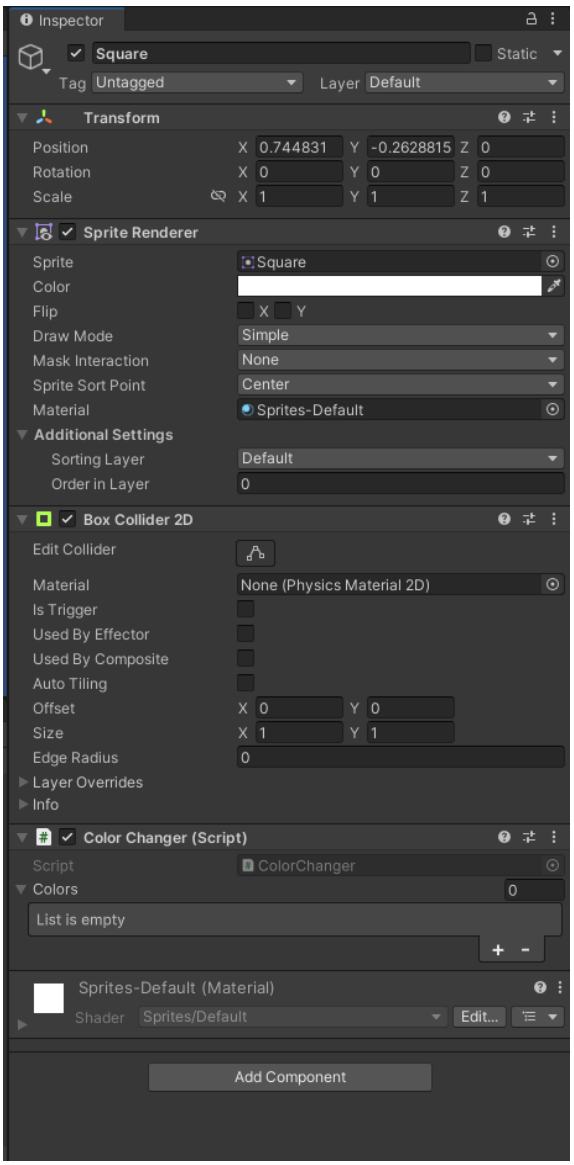
a. 

5. Next, Right click on the Project Window > Create > C# Script. Name this script as “ColorChanger”. Open the script.
6. Add the following code.

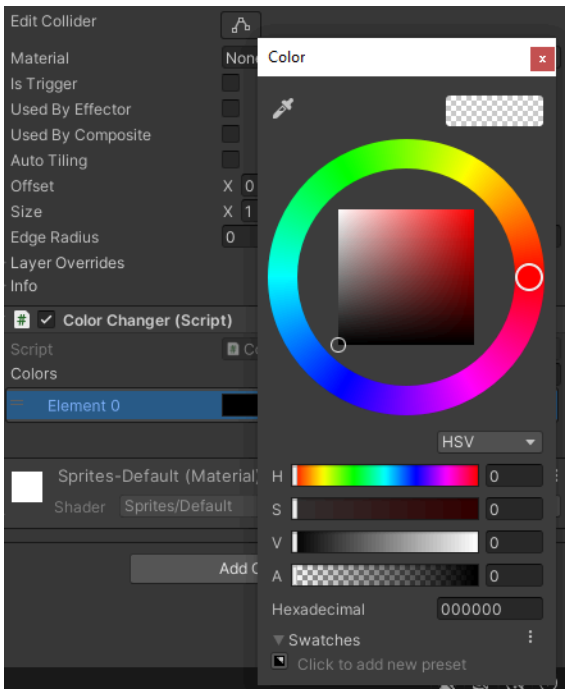
```C/C++
using System.Collections.Generic;
using UnityEngine;

public class ColorChanger : MonoBehaviour
{
  // List of colors
  public List<Color> colors;
  private int currentColorIndex = 0; //current color

  void Update()
  {
    if (Input.GetMouseButtonDown(0)) //check for mouse click
    {
```

```
      Vector2 mousePosition =
Camera.main.ScreenToWorldPoint(Input.mousePosition);//get mouse position
      RaycastHit2D hit = Physics2D.Raycast(mousePosition, Vector2.zero); //Cast
ray from mouse position

      if (hit.collider != null && hit.transform == this.transform) //check if ray
hits square
      {
        // Change to the next color in the list
        currentColorIndex = (currentColorIndex + 1) % colors.Count;
        GetComponent<SpriteRenderer>().color = colors[currentColorIndex];
      }
    }
  }
}
```

7. Save the script and return to Unity.
8. With the square selected, drag and drop the script into the inspector of the square to attach it.

a. 

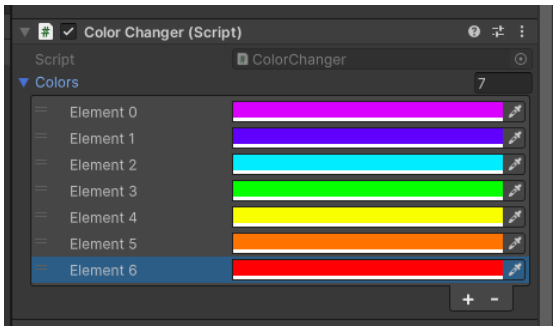
9. Click on the “+” symbol of the script.
10. Click on the black color bar to open the color selector menu.

a. 

11. Set the A value to 100.

a. 

12. Choose violet in the color picker tool.
13. Repeat steps 9 to 11 and then create colors for Indigo, Blue, Green, Yellow, Orange & Red. It should look similar to the following image.

a. 

14. Next, click on play.
15. When you click on the square in game mode, it should now change its color.