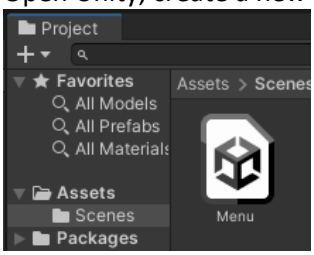
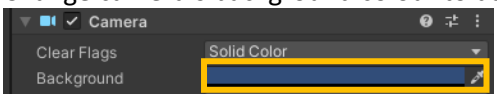
Unity Dynamic objects

3 Lane Game

Open Unity, create a new 2D project, and save the scene as **"Menu".** Make sure your scene is in 2D view.



Change camera's background colour to dark purple.



You can change colour to a precise value by entering Hex Color 130011

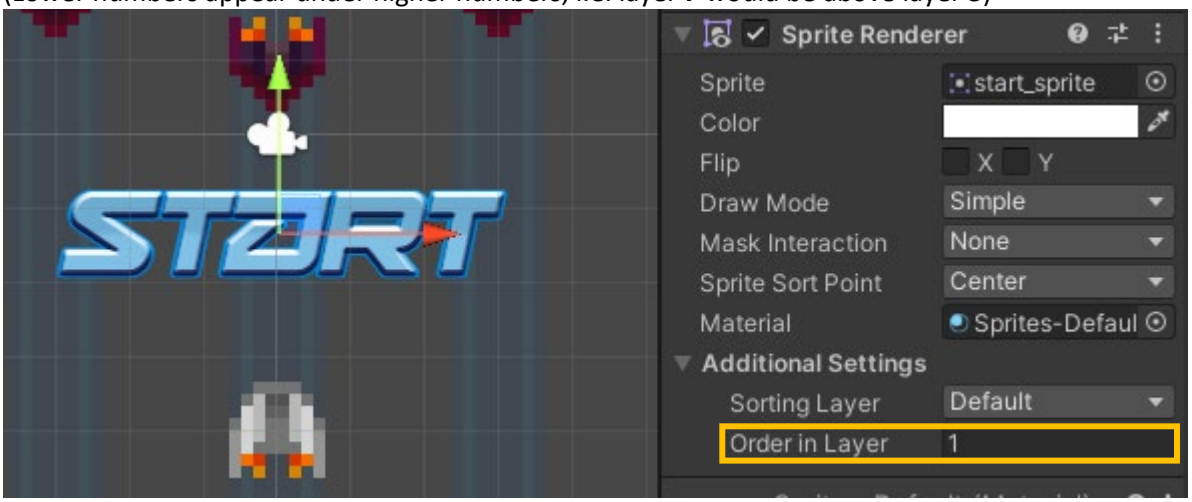
Hex Color consists of RED GREEN BLUE

Import Assets from the "**Tap Tap Escape Assets"** Folder. Make sure they are converted to **Sprite (2D and UI)**.
(If not – select all of them, and in the **Inspector**, click on **Texture Type/ Sprite (2d and UI)**.)

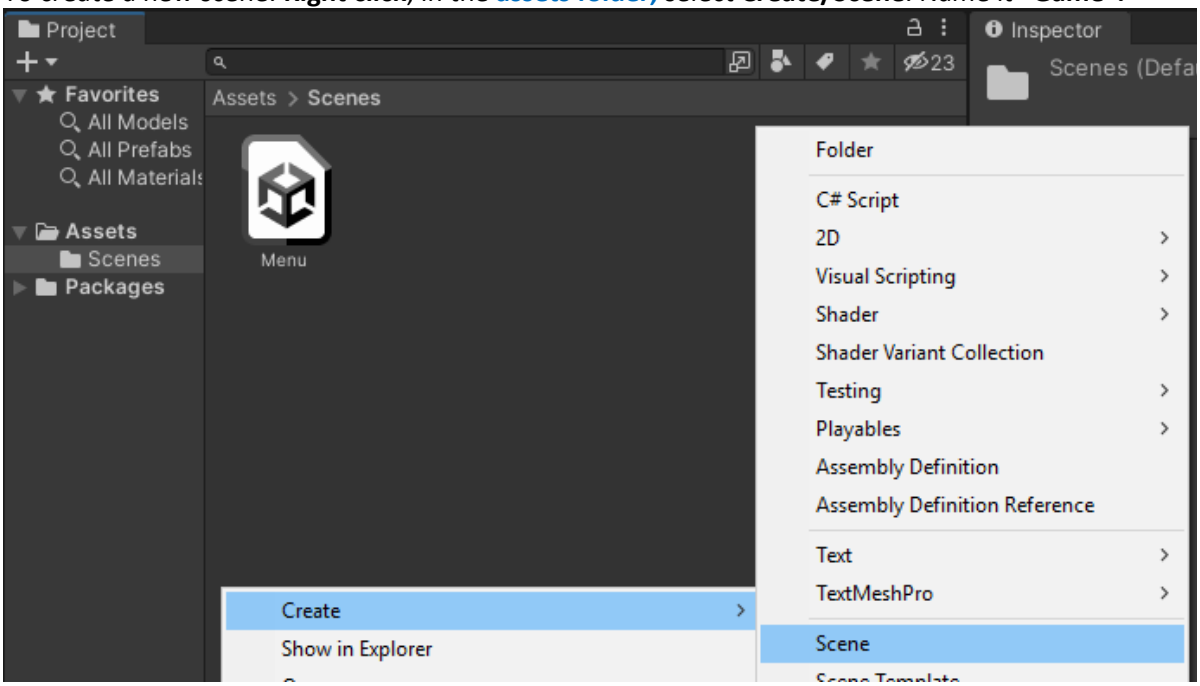
Create a simple menu that looks something like this.

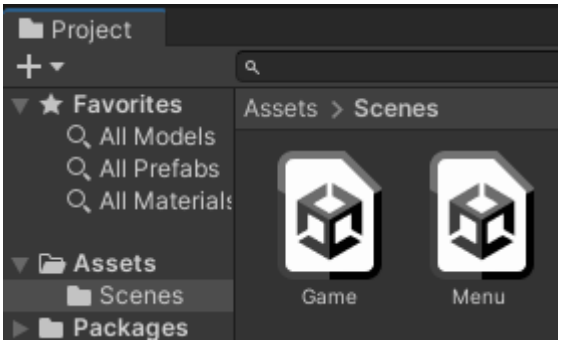
To do this use object manipulation tools.

You can change **Order of Layer –** by changing a number.
(Lower numbers appear under higher numbers, i.e. layer **7** would be above layer **3**)



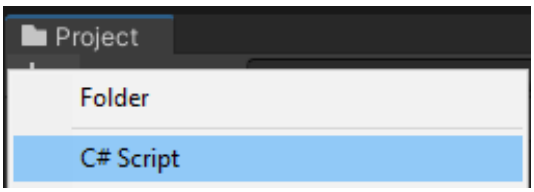
When you're finished with the menu, create a new scene. (Keep all of your scenes in the scene folder!)
To create a new scene: **Right click**, in the **assets folder,** select **Create/Scene**. Name it **"Game".**



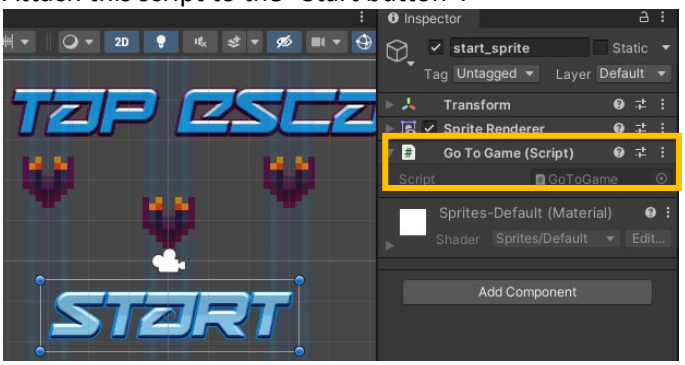


Save your progress in the menu by pressing [CTRL + S].
You can double Click on the **Game** Scene- it should be empty. Then Double Click back to the **Menu** Scene.

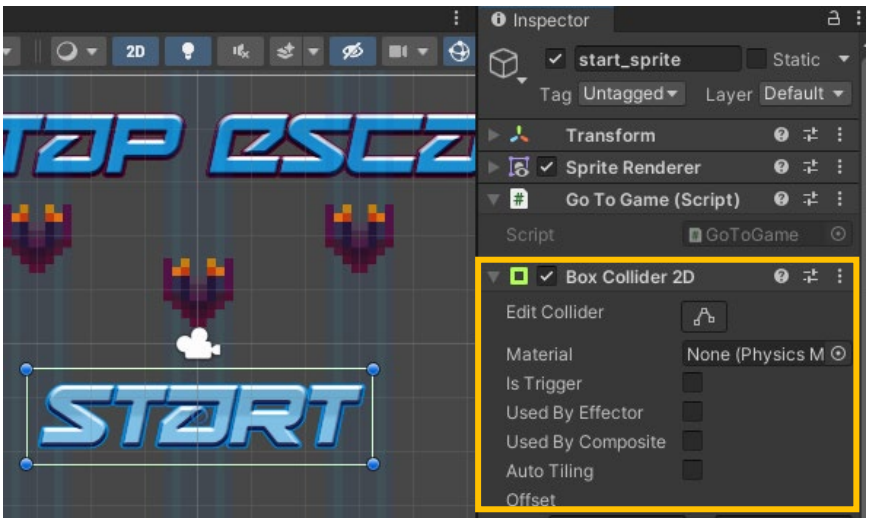
Create a C# Script – name it "**GoToGame**".
This script will enable you to click on "start button" and load the game scene.



Attach this script to the "Start button".



Additionally add a “**Box collider 2D”** to the button – so that it can be interactive with mouse functions.



Open your script, and add the following code.

Notice – I removed **Start()** and **Update()** functions – because I will not be using them.

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

public class GoToGame : MonoBehaviour
{

    private void OnMouseDown()
    {
        SceneManager.LoadScene("Game");
    }

}
```

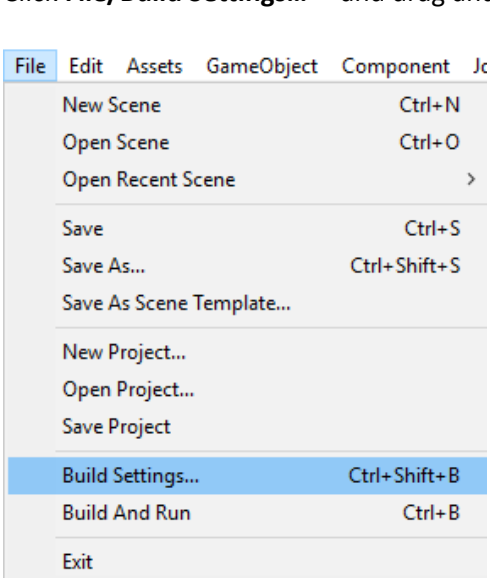
The first line adds **Scene Management** capabilities to the script.

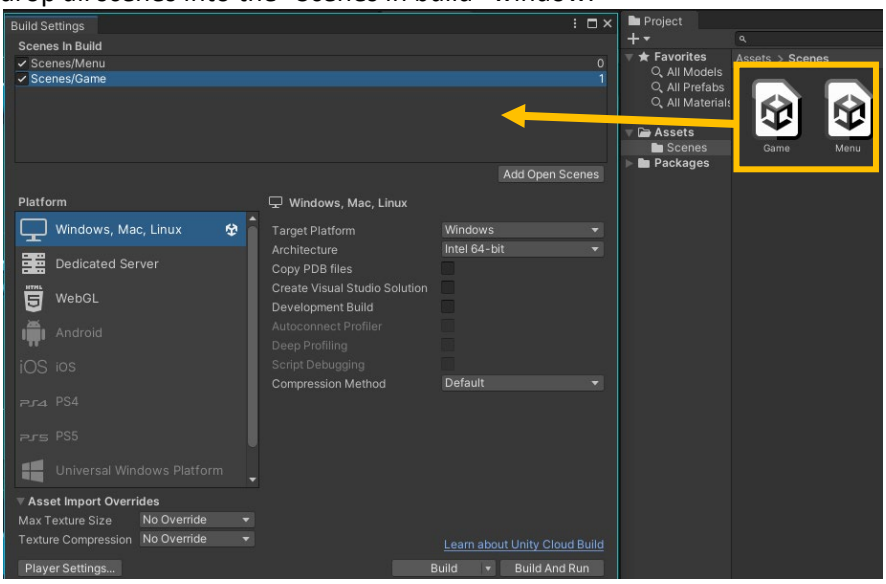
OnMouseDown function will load the “Game” scene, once the button is pressed.

The scene is called by a string name- make sure that it corresponds to its name in the project folder.

If you try to run the game now, it will give you an error- when you try to click on the start button. That’s because scenes must be added to the build, before they can be accessed.

Click **File/Build Settings…** and drag and drop all scenes into the “Scenes in build” window.

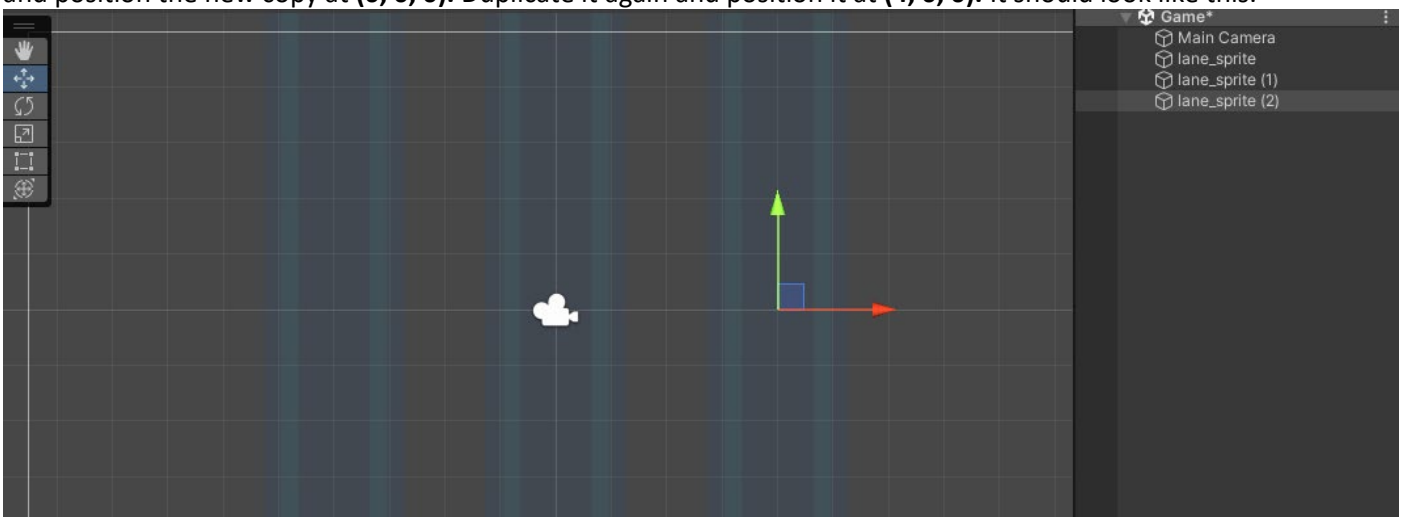




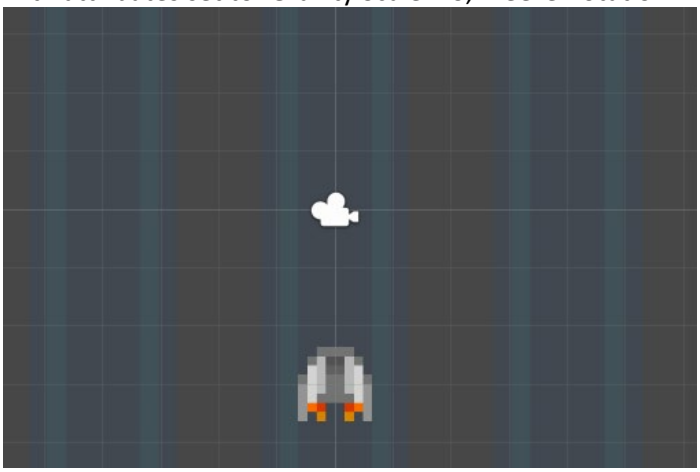
Save your project and press play. When you press on the “Start Button” Unity should load the “Game” Scene. Double click “Game” scene to access it. Change camera’s background colour to the same colour as the menu.

The main mechanic of the game is to dodge oncoming waves of enemies. Player’s movement is restricted to 3 lanes.

Create 3 lanes- by moving **lane_sprite** to the scene. Position it at **(-4, 0, 0)** and Scale it to **(2, 15, 1).** Then duplicate it and position the new copy at **(0, 0, 0).** Duplicate it again and position it at **(4, 0, 0).** It should look like this.



Now add the player object. Drag the **player_srite** to the scene. Add components **Box Collider 2D** and **Rigidbody2D** with attributes set to: Gravity Scale = **0**, Freeze Rotation = **Z**. Position it at – **(0, -3, 0).** Set Rotation to **(0, 0, 180).**



Add a **C# Script** for player’s movement – name it “**PlayerControl**”.
The player needs to move within 3 fixed positions. Create 3 Vectors, that will be used for positions.
And set player’s position to the middle point on the start function.

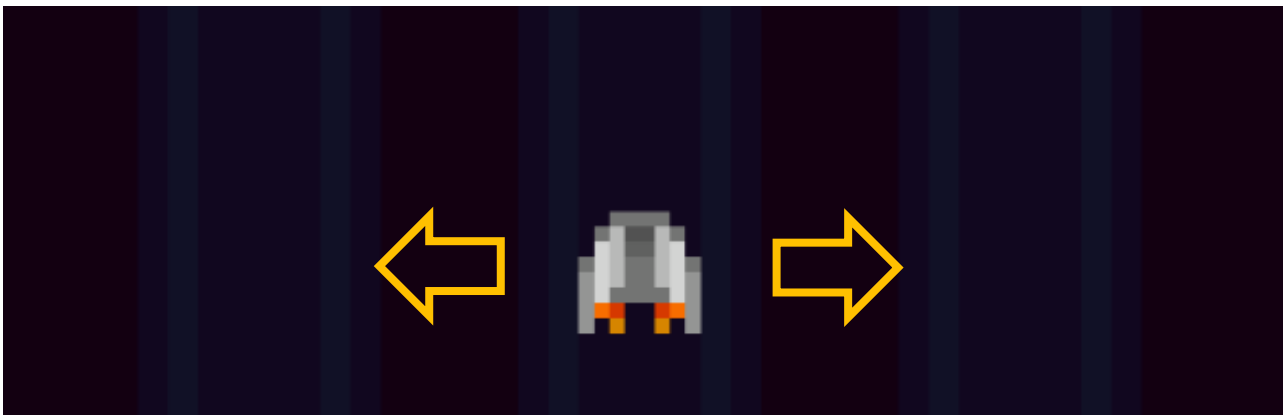
```
Vector3 leftPos = new Vector3(-4,-3,0);
Vector3 midPos = new Vector3(0, -3, 0);
Vector3 rightPos = new Vector3(4, -3, 0);

void Start()
{
    transform.position = midPos;
}
```

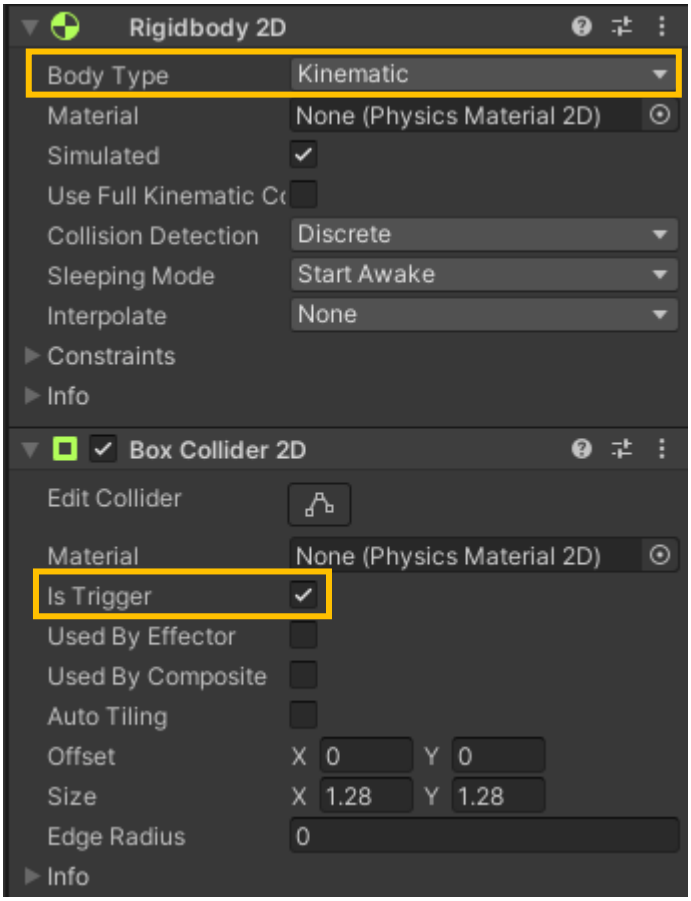
Now we need to add input and logic for the movement.

Add the following code in the Update function.
The code checks for current position and moves the player to the next possible position – based on input.

```
private void Update()
{
    //left input
    if (Input.GetKeyDown(KeyCode.LeftArrow))
    {
        if(transform.position == midPos)
        {
            transform.position = leftPos;
        }
        else if(transform.position == rightPos)
        {
            transform.position = midPos;
        }
    }
    //right input
    if (Input.GetKeyDown(KeyCode.RightArrow))
    {
        if (transform.position == midPos)
        {
            transform.position = rightPos;
        }
        else if (transform.position == leftPos)
        {
            transform.position = midPos;
        }
    }
}
```

Save this script and run your game. You should be able to control your player using **Left and Right** arrow keys.



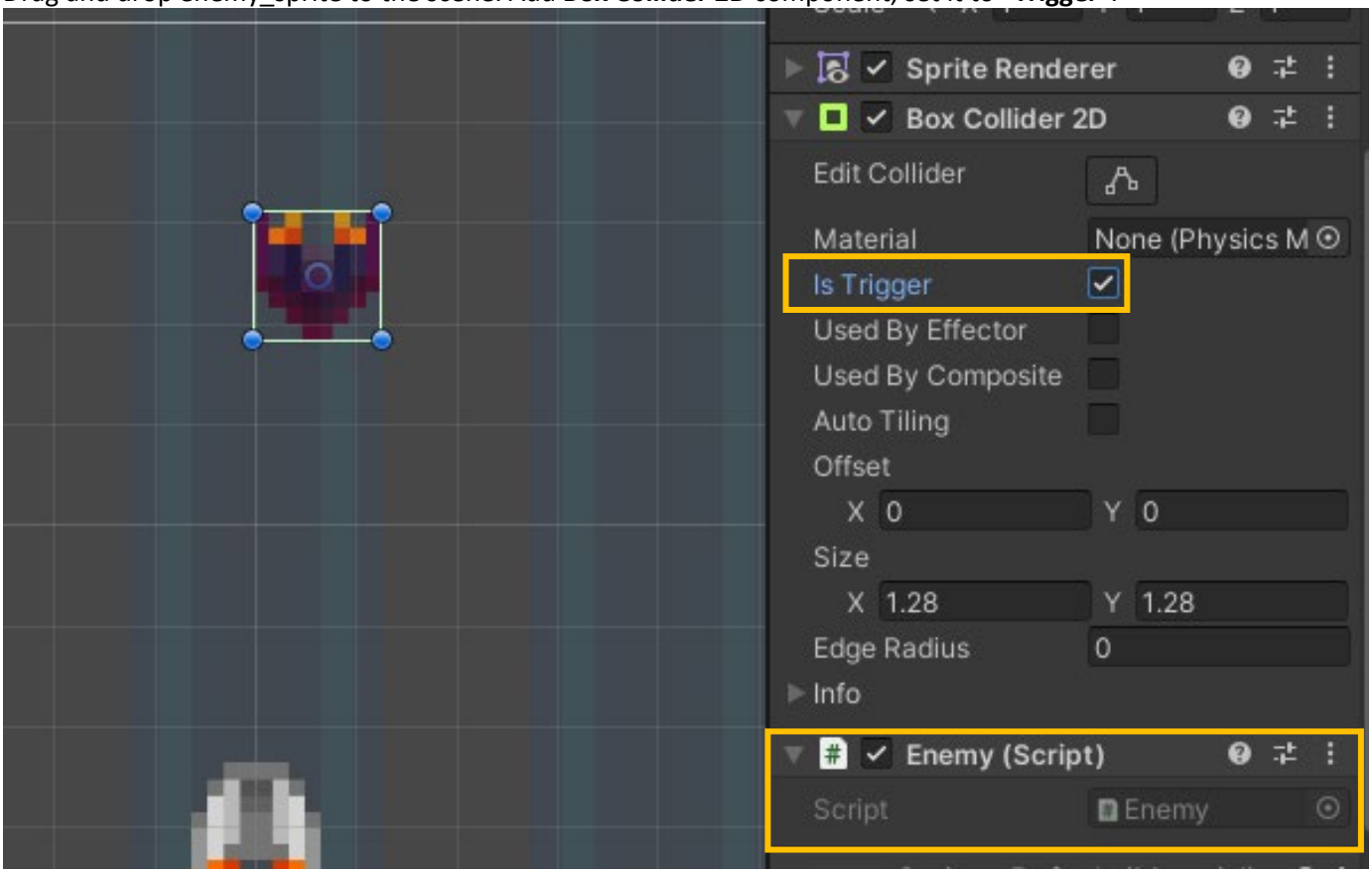


Now add a **Rigidbody2D component** to the player. This is needed to detect collision.

Change body type to "**Kinematic**".
Kinematic body can detect collision but is not affected by physics.

Add **Box Collider 2D** component – and set it to "**Trigger**", this will change it from collider to a trigger – that way- objects can freely pass through each other, yet they can still detect the interaction.

Now let's create **enemy** object. The enemy behaviour will be very simple- it will move down and destroy the player. Drag and drop enemy_sprite to the scene. Add **Box Collider 2D** component, set it to "**Trigger**".



Create and add **Enemy Script** to the enemy.

Add the following.

```
public class Enemy : MonoBehaviour
{

    float speed = 0.1f;

    private void Start()
    {
        Destroy(gameObject, 5f);
    }

    void FixedUpdate()
    {
        transform.Translate(0, -speed, 0);
    }
}
```

Speed variable will control the speed of movement.

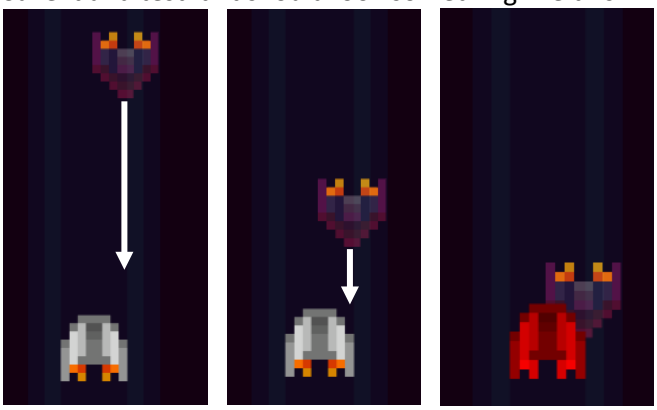
On Start function – we activate a Destroy function which will destroy the enemy after 5 seconds.

On **FixedUpdate** the object moves down using Translate function.

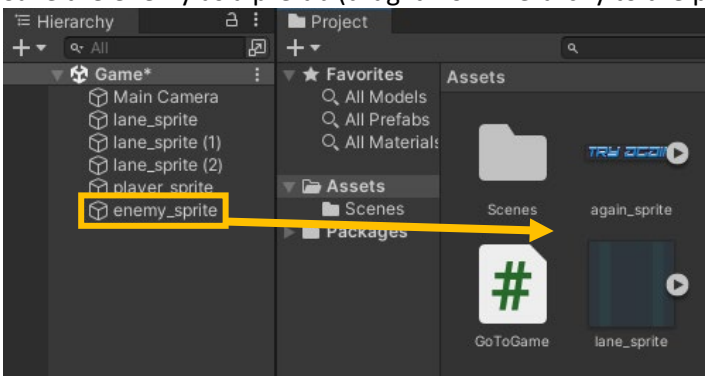
Save it, test it, check if enemy's speed needs an adjustment.
Then add an additional function which will change player's colour to red once it collided with the enemy object.

```
private void OnTriggerEnter2D(Collider2D collision)
{
    collision.GetComponent<SpriteRenderer>().color = Color.red;
}
```

Save it and test it. It should look something like this.



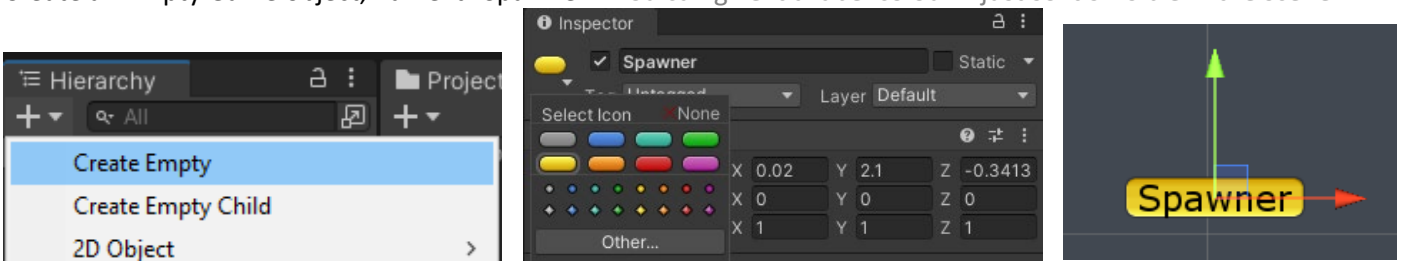
This will signify when the player is hit.
Also check that the enemy can reach the bottom of the screen within 5 seconds before destroying itself.
Save the enemy as a prefab (drag it from hierarchy to the project window).



Delete enemy prefab in the hierarchy view. (So, it doesn't get on the way)

Now let's create a spawner for our enemies. Enemies will be spawning in 3 lanes at certain time interval.
Let's create a spawner object.

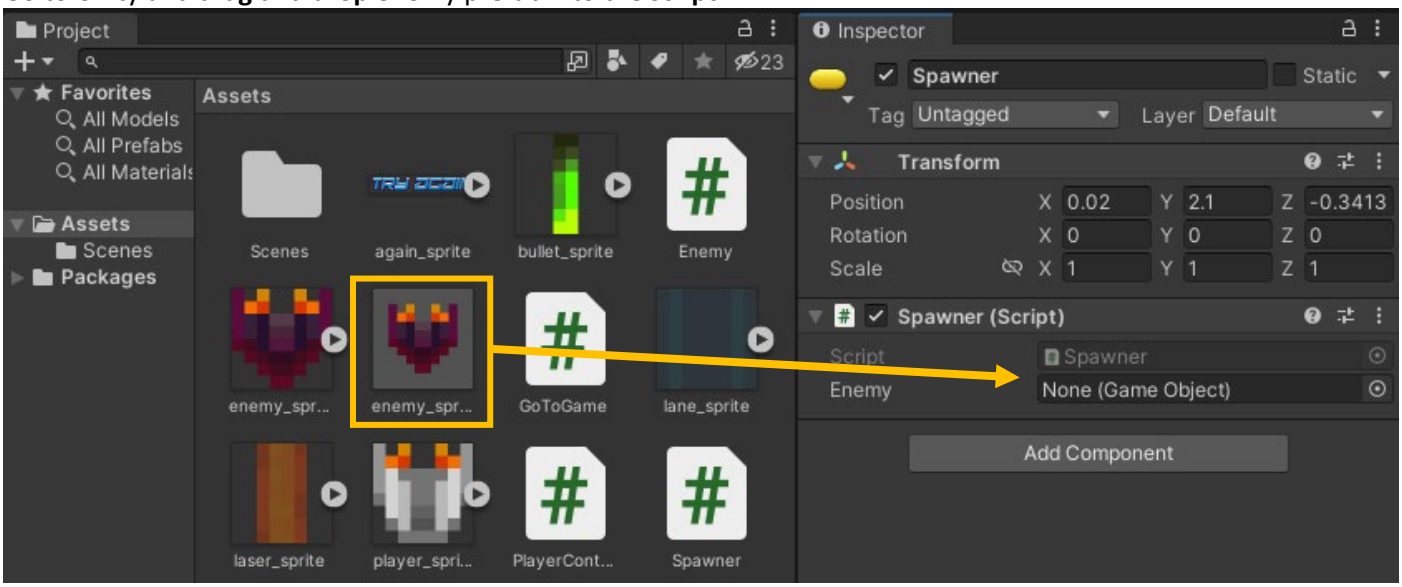
Create an Empty Game object, name it "Spawner". You can give it a label colour – just so it's visible in the scene.



Create and attach a Spawner Script to this object. For now just add a public Game object "enemy".

```
public GameObject enemy;

void Update()
{

}
```

Go to Unity and **drag and drop** enemy prefab into the **script**.



Add the following to your script.

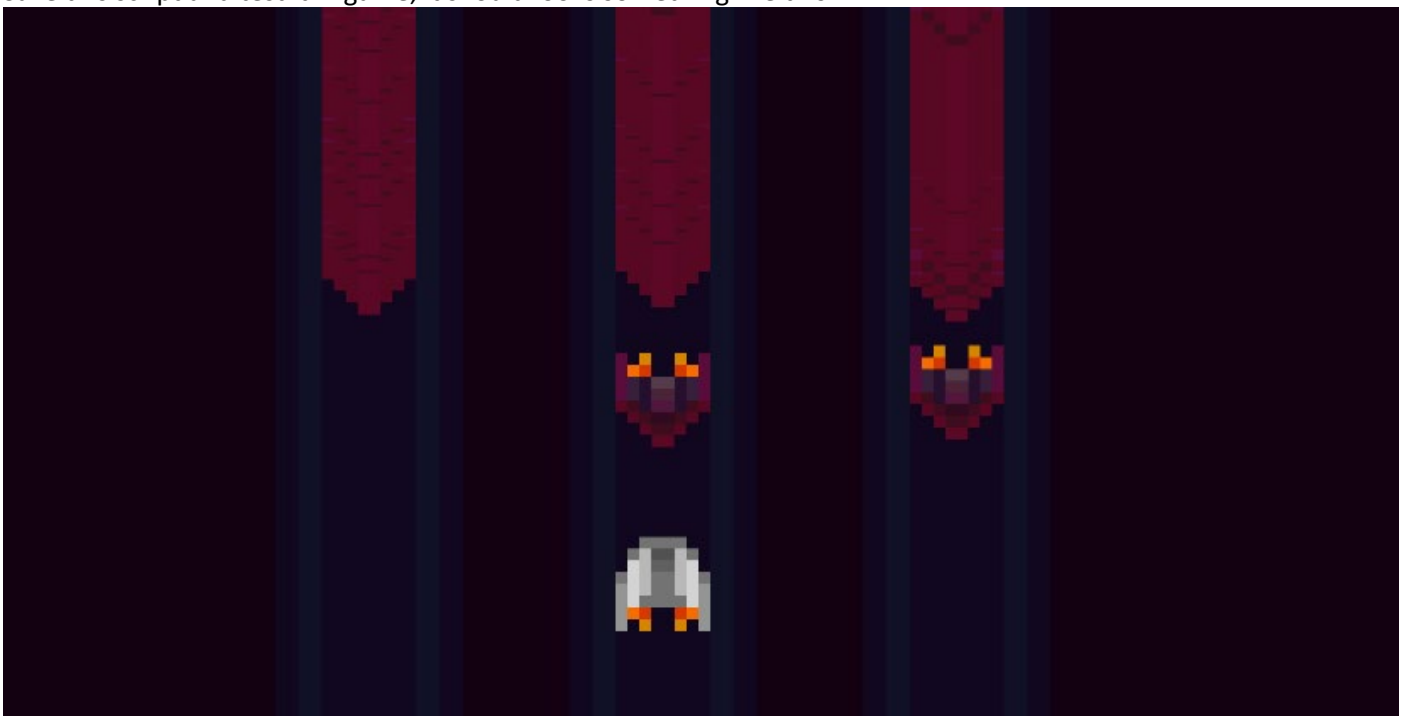
```
public class Spawner : MonoBehaviour
{

    public GameObject enemy;
    Vector3 pos1 = new Vector3(4, 6, 0);
    Vector3 pos2 = new Vector3(0, 6, 0);
    Vector3 pos3 = new Vector3(-4, 6, 0);
    int randNum;

    void Update()
    {
        randNum = Random.Range(0, 3);

        if(randNum == 0)Instantiate(enemy,pos1,Quaternion.identity);
        if (randNum == 1) Instantiate(enemy, pos2, Quaternion.identity);
        if (randNum == 2) Instantiate(enemy, pos3, Quaternion.identity);
    }
}
```

pos1,pos2 and pos3 are 3 different points where the enemies will spawn.

randNum integer will be used to store random numbers.

Enemies will spawn at different positions based on the random number.

Save this script and test it in game, it should looks something like this.



The enemies are spawning in 3 different lanes; however, they are spawning way too often!
The next step is to add a timer and move spawning of enemies into a function so it looks this.

```
public GameObject enemy;
Vector3 pos1 = new Vector3(4, 6, 0);
Vector3 pos2 = new Vector3(0, 6, 0);
Vector3 pos3 = new Vector3(-4, 6, 0);
int randNum;
float timer;
float interval = 0.8f;//seconds

void Update()
{
    timer += Time.deltaTime;
    if(timer > interval)
    {
        SpawnEnemies();
        timer = 0;
    }
}

void SpawnEnemies()
{
    randNum = Random.Range(0, 3);

    if(randNum == 0)Instantiate(enemy,pos1,Quaternion.identity);
    if (randNum == 1) Instantiate(enemy, pos2, Quaternion.identity);
    if (randNum == 2) Instantiate(enemy, pos3, Quaternion.identity);
}
```

The timer will increase (in seconds) and once it passes the interval (0.8 seconds)

It will spawn enemies and **reset to 0**

If you test it, it should look something like this.

Tasks

Task 1: Add a “Try again” button once the enemy is hit.

Task 2: If you’re in game and you press “Esc” it will go to the menu scene.

Task 3: Implement a mechanic where the player automatically goes to the next stage after dodging 10 enemies. In Level 2 enemies shoot green bullets. (This is open ended task – there is no “right or wrong” solution).