Practical 1

Aim: Setup DirectX 11, Window Framework and Initialize Direct3D Device. **Code:**

```
using System;
using
System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using
Microsoft.DirectX.Direct3D;

namespace practical1{
   public partial class Form1 : Form{
      Microsoft.DirectX.Direct3D.Device  devices;
      public Form1(){
         InitializeComponent();
         InitDevice();}
      private void InitDevice(){
         PresentParameters pp = new PresentParameters();
         pp.Windowed = true;
         pp.SwapEffect = SwapEffect.Discard;
         devices = new Device(0, DeviceType.Hardware, this,CreateFlags.HardwareVertexProcessing, pp);}
      private void Render() {
      devices.Clear(ClearFlags.Target, Color.CornflowerBlue, 0, 1);
         devices.Present();}
      private void Form1_Load(object sender, EventArgs e){
      private void Form1_Paint(object sender, PaintEventAr{
         Render();
      }
   }
}
```

Practical 2

Aim: Buffers, shaders and HLSL (Draw a triangle using Direct3D 11)

Code:

```
using System;
using
System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;

namespace prac2{
   public partial class Form1 : Form{
      Microsoft.DirectX.Direct3D.Device
      device; public Form1(){
         InitializeComponent();
         InitDevice();}
      private void InitDevice(){
         PresentParameters pp = new PresentParameters();
         pp.Windowed = true;
         pp.SwapEffect = SwapEffect.Discard;
         device = new Device(0, DeviceType.Hardware, this, CreateFlags.SoftwareVertexProcessing, pp);}
      private void Render(){
         CustomVertex.TransformedColored[] v = new
         CustomVertex.TransformedColored[3]; v[0].Position = new Vector4(100, 100, 0,
         1.4f);
         v[1].Position = new Vector4(150, 300, 0, 1.0f);
         v[2].Position = new Vector4(80, 300, 0, 0);
         v[0].Color = System.Drawing.Color.ForestGreen.ToArgb();
         v[1].Color = System.Drawing.Color.FromArgb(255, 0,
         0).ToArgb(); device.Clear(ClearFlags.Target, Color.Blue, 0, 1);
         device.BeginScene();
         device.VertexFormat = CustomVertex.TransformedColored.Format;
         device.DrawUserPrimitives(PrimitiveType.TriangleList, 1, v);
         device.EndScene();
         device.Present();}
      private void Form1_Load(object sender, EventArgs e){}
      private void Form1_Paint(object sender, PaintEventArgs e){
      Render();    } }}
```

Practical 3

Aim: Texturing (Texture the Triangle using Direct 3D 11).

Code:

```
using System;
using
System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using
Microsoft.DirectX.Direct3D;
using Microsoft.DirectX;

namespace prac3{
public partial class Form1 : Form{
Microsoft.DirectX.Direct3D.Device device;
public Form1(){
InitializeComponent();
InitDevice(); }
public void InitDevice() {
PresentParameters pp = new PresentParameters();
pp.Windowed = true;
pp.SwapEffect = SwapEffect.Discard;
device = new Device(0, DeviceType.Hardware, this,
CreateFlags.SoftwareVertexProcessing,pp);}
public void Render()
{ CustomVertex.PositionTextured[] v = new CustomVertex.PositionTextured[3];Texture t;
device.Transform.Projection = Matrix.PerspectiveFovLH(3.14f / 4, device.Viewport.Width
/device.Viewport.Height, 1f,1000f);device.Transform.View = Matrix.LookAtLH(new
Vector3(0,0, 20), new Vector3(), new Vector3(0, 1, 0));
device.RenderState.Lighting = false;
v[0] = new CustomVertex.PositionTextured(new Vector3(0, 4, 4), 0, 0);
v[1] = new CustomVertex.PositionTextured(new Vector3(-1, -4, 4), -1, 0);
v[2] = new CustomVertex.PositionTextured(new Vector3(1, -4, 4), 0, -1);
t = new Texture(device, new
Bitmap(&amp;quot;C:\\Users\\Pradeep\\Pictures\\IMG_20220729_193656_517.png&amp;quot;),0,Po
ol.
managed);device.Clear(ClearFlags.Target, Color.Black, 1, 0);
device.BeginScene();
device.SetTexture(0, t);
device.VertexFormat = CustomVertex.PositionTextured.Format;
device.DrawUserPrimitives(PrimitiveType.TriangleList, v.Length / 3, v);
device.EndScene();device.Present();} private void Form1_Load(object sender, EventArgs
e){ }
        Private void Form1_Paint(object sender, PaintEventArgs e) { Render() } }}
```

Practical 4

Aim: Lightning (Programmable Diffuse Lightning using Direct3D 11).

Code:

```
using System;
using
System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;

namespace prac4{
public partial class Form1 : Form{
Device device;
public Form1(){
InitializeComponent();
InitDevice();}
public void InitDevice(){
PresentParameters pp = new PresentParameters();
pp.Windowed = true;
pp.SwapEffect = SwapEffect.Discard;
device = new Device(0, DeviceType.Hardware, this, CreateFlags.SoftwareVertexProcessing, pp);
} public void Render(){1000f);
CustomVertex.PositionNormalColored[] v = new CustomVertex.PositionNormalColored[3];
device.Transform.Projection = Matrix.PerspectiveFovLH(3.14f / 4, device.Viewport.Width / device.Viewport.Height, 1f,
device.Transform.View = Matrix.LookAtLH(new Vector3(2, 0, 10), new Vector3(), new Vector3(9, 50, 50));
device.RenderState.Lighting = false;
v[0] = new CustomVertex.PositionNormalColored(new Vector3(0, 1, 1), new Vector3(1, 0, 1), Color.Red.ToArgb());
v[1] = new CustomVertex.PositionNormalColored(new Vector3(-1, -1, 1), new Vector3(1, 0, 1), Color.Red.ToArgb());
v[2] = new CustomVertex.PositionNormalColored(new Vector3(0, -1, 1), new Vector3(-1, 0, 1), Color.Red.ToArgb());
device.RenderState.Lighting = true;
device.Lights[0].Type = LightType.Directional;
device.Lights[0].Diffuse = Color.Red;
device.Lights[0].Direction = new Vector3(0.8f, 0, -1);
device.Lights[0].Enabled = true;
device.Clear(ClearFlags.Target, Color.RoyalBlue, 1, 0);
device.BeginScene();
device.VertexFormat = CustomVertex.PositionNormalColored.Format;
device.DrawUserPrimitives(PrimitiveType.TriangleList, v.Length / 3, v);
device.EndScene();
device.Present(); }
private void Form1_Load(object sender, EventArgs e)
{ }
private void Form1_Paint(object sender, PaintEventArgs e) {
Render(); } }}
```

Practical 5

Aim: Specular Lightning (Programmable Spot Lightning using Direct3D 11). **Code:**

```
using System;
using
System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Microsoft.DirectX;
using
Microsoft.DirectX.Direct3D;

namespace Prac5{
public partial class Form1 : Form {
Microsoft.DirectX.Direct3D.Device device;
private CustomVertex.PositionNormalColored[] v = new CustomVertex.PositionNormalColored[3];
public Form1() {
InitializeComponent();
InitDevice(); }
public void InitDevice() {
PresentParameters pp = new PresentParameters();
pp.Windowed = true;
pp.SwapEffect = SwapEffect.Discard;
device = new Device(0, DeviceType.Hardware, this, CreateFlags.HardwareVertexProcessing, pp);
device.Transform.Projection = Matrix.PerspectiveFovLH(3.14f / 4, device.Viewport.Width / device.Viewport.Height, 1f,
1000f);
device.Transform.View = Matrix.LookAtLH(new Vector3(0, 0, 10), new Vector3(), new Vector3(0, 1, 0));
device.RenderState.Lighting = false;
v[0] = new CustomVertex.PositionNormalColored(new Vector3(0, 1, 1), new Vector3(1, 0, 1), Color.Red.ToArgb());
v[1] = new CustomVertex.PositionNormalColored(new Vector3(-1, -1, 1), new Vector3(1, 0, 1), Color.Blue.ToArgb());
v[2] = new CustomVertex.PositionNormalColored(new Vector3(1, -1, 1), new Vector3(-1, 0, 1), Color.Green.ToArgb());
device.RenderState.Lighting = true;
device.Lights[0].Type = LightType.Directional;
device.Lights[0].Diffuse = Color.Plum;
device.Lights[0].Direction = new Vector3(0.8f, 0, -1);
device.Lights[0].Enabled = true;}
private void Form1_Load(object sender, EventArgs e){ }
private void Form1_Paint(object sender, PaintEventArgs e)
{ device.Clear(ClearFlags.Target, Color.BlueViolet, 1, 0);
device.BeginScene();
device.VertexFormat = CustomVertex.PositionNormalColored.Format;
device.DrawUserPrimitives(PrimitiveType.TriangleList, v.Length / 3, v);
device.EndScene();
device.Present();
} }}
```

Practical 6

Aim: Loading models into DirectX 11 and rendering.

Code:

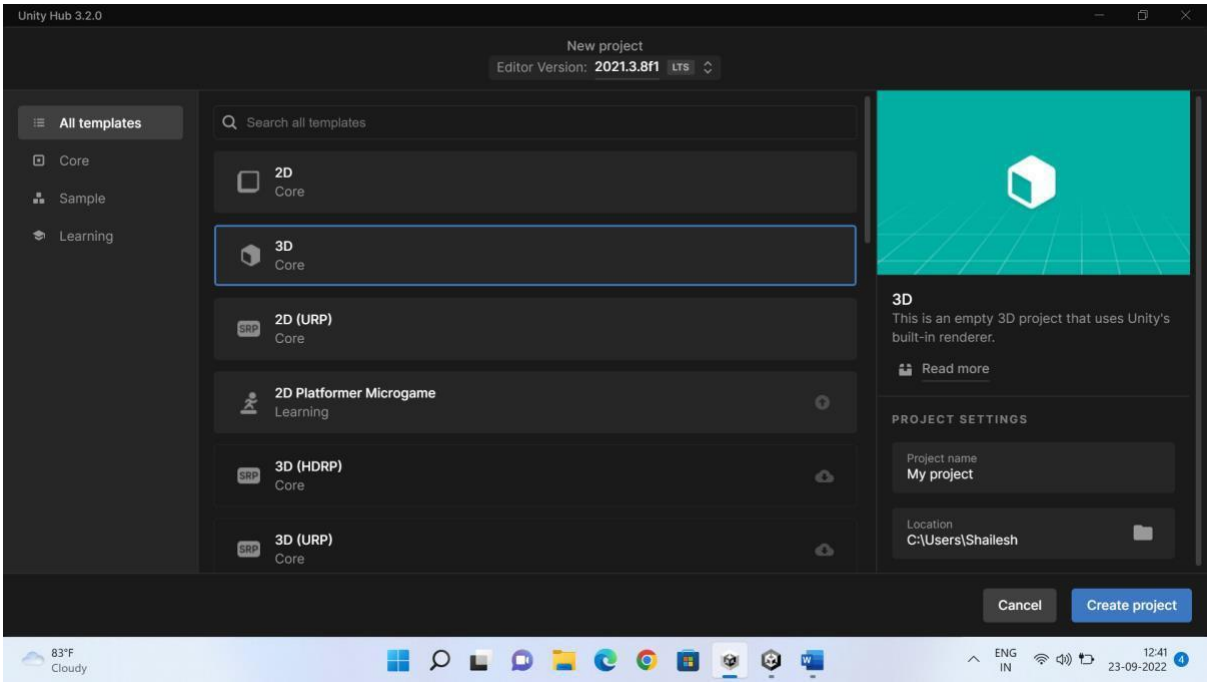
```
using System;
using
System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using
Microsoft.DirectX.Direct3D;
using Microsoft.DirectX;

namespace prac6
{
   public partial class Form1 : Form
   {
      Microsoft.DirectX.Direct3D.Device
      device;
      Microsoft.DirectX.Direct3D.Texture
      texture; Microsoft.DirectX.Direct3D.Font
      font; public Form1()
      {
         InitializeCompone
         nt(); InitDevice();
         InitFont();
         LoadTexture();
      }
      private void InitFont()
      {
         System.Drawing.Font f = new System.Drawing.Font("Arial", 16f,
         FontStyle.Regular); font = new Microsoft.DirectX.Direct3D.Font(device, f);
      }

      private void LoadTexture()
      {
         texture = TextureLoader.FromFile(device, "C:\\Users\\Pradeep\\Pictures\\salman.png",400, 400, 1, 0,
Format.A8B8G8R8, Pool.Managed, Filter.Point, Filter.Point, Color.Transparent.ToArgb());
      }

      private void InitDevice()
```

```
        {
            PresentParameters pp = new PresentParameters();
            pp.Windowed = true;
            pp.SwapEffect = SwapEffect.Discard;
            device = new Device(0, DeviceType.Hardware, this, CreateFlags.HardwareVertexProcessing, pp);
        }

        public void Render()
        {
            device.Clear(ClearFlags.Target, Color.CornflowerBlue, 0, 1);
            device.BeginScene();
            using (Sprite s = new Sprite(device))
            {
                s.Begin(SpriteFlags.AlphaBlend);
                s.Draw2D(texture, new Rectangle(0, 0, 0, 0), new Rectangle(0, 0, device.Viewport.Width,
device.Viewport.Height), new Point(0, 0), 0f, new Point(0, 0), Color.White);
                font.DrawText(s, "Salman Khan", new Point(0, 0), Color.White);
                s.End();
            }
            device.EndScene();
            device.Present();

        }


        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void Form1_Paint(object sender, PaintEventArgs e)
        {
            Render();
        }
    }
```

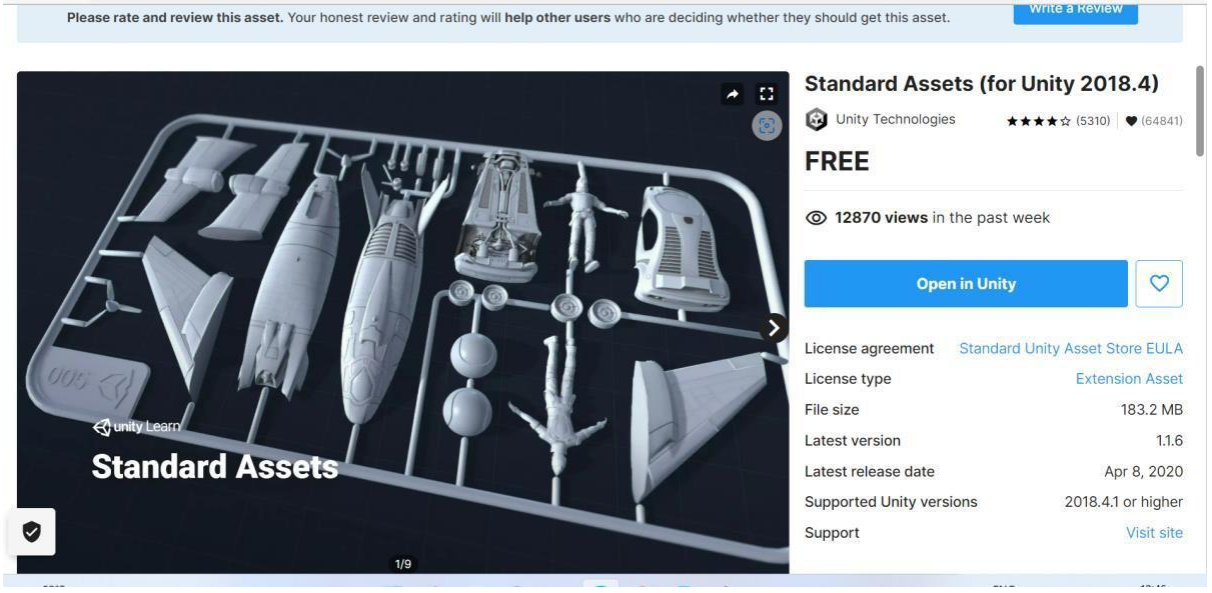
Practical 7

Aim- Car Game 3d in unity

Step 1-click on 3d project name it and choose the save location



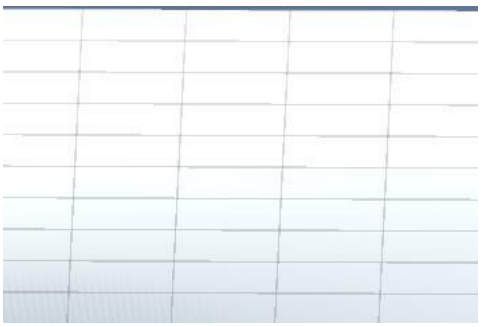
Step 2 go to asset store and asset from the following website of unity store which is standard assets for vehicle



Step 3

it will add asset automatically in the explorer by going into package manger and import them

Step 5 – right click go to 3d bject and click on terrain Terrain will be added you can choose to move it or even expand



Step 6 – place the prefabs like roads & vehicles from the asset

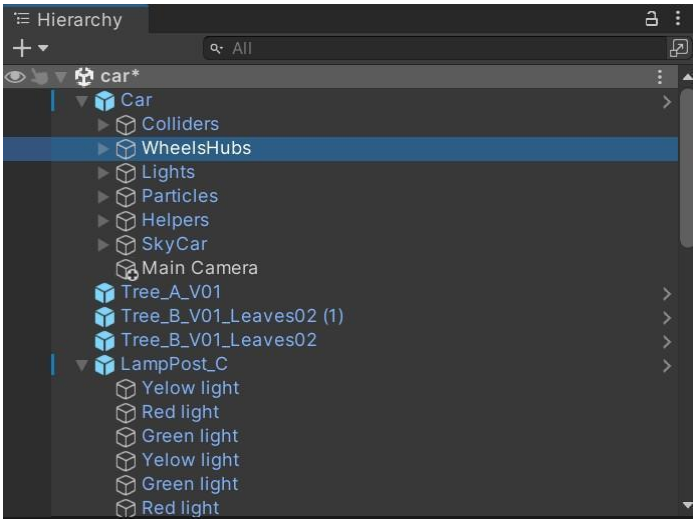
Step 7 – Code for the vehicle

CarUserControl.cs

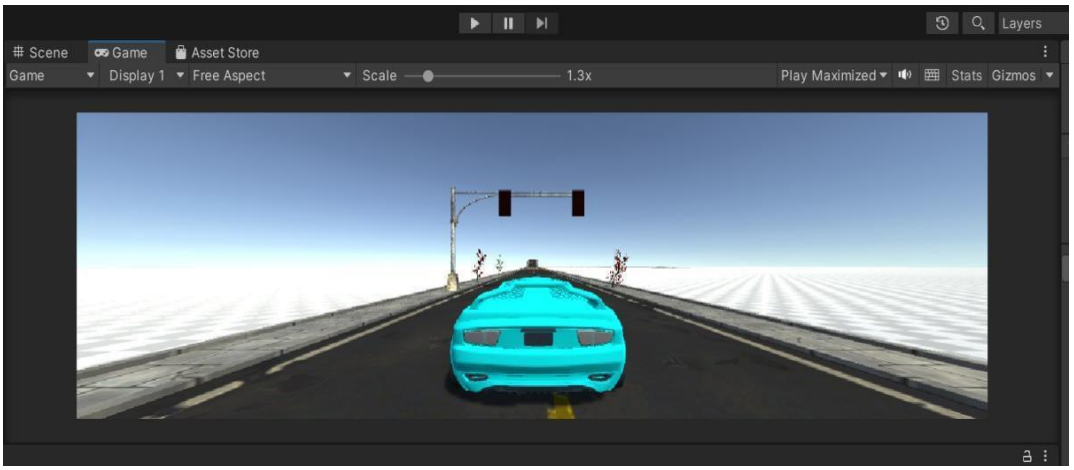
Miscellaneous Files — UnityStandardAssets.Vehicles.Car.CarUserControl

```
using System;
using UnityEngine;
using UnityStandardAssets.CrossPlatformInput;

namespace UnityStandardAssets.Vehicles.Car
{
    [RequireComponent(typeof (CarController))]
    public class CarUserControl : MonoBehaviour
    {
        private CarController m_Car; // the car controller we want to use


        private void Awake()
        {
            // get the car controller
            m_Car = GetComponent<CarController>();
        }


        private void FixedUpdate()
        {
            // pass the input to the car!
            float h = CrossPlatformInputManager.GetAxis("Horizontal");
            float v = CrossPlatformInputManager.GetAxis("Vertical");
#if !MOBILE_INPUT
            float handbrake = CrossPlatformInputManager.GetAxis("Jump");
            m_Car.Move(h, v, v, handbrake);
#else
            m_Car.Move(h, v, v, 0f);
#endif
        }
    }
}
```

Step 8 – drag main camera to the car so that when the movement of car happens the camera follows the vehicle



Step 9- press play button to run the project

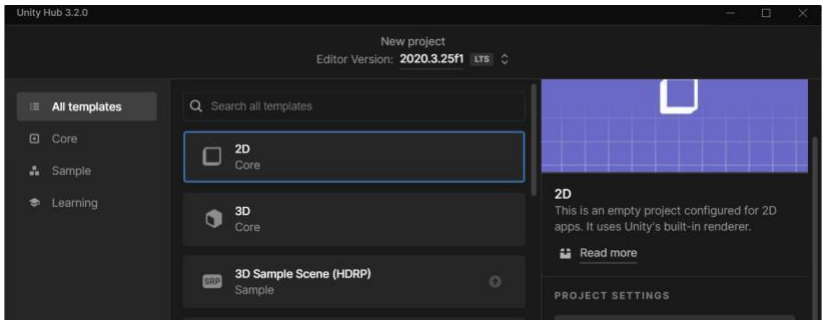


Step 10- control the car using arrow keys or W-A-S-D.

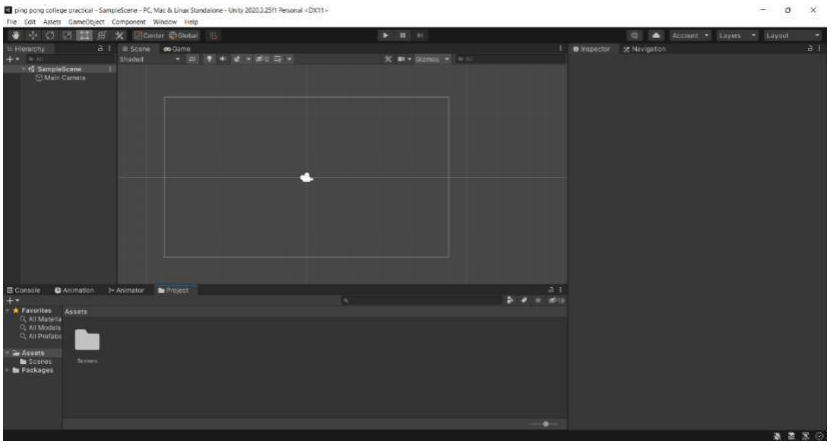
Practical 6

Aim: **:** Creating a 2D ping pong game in unity

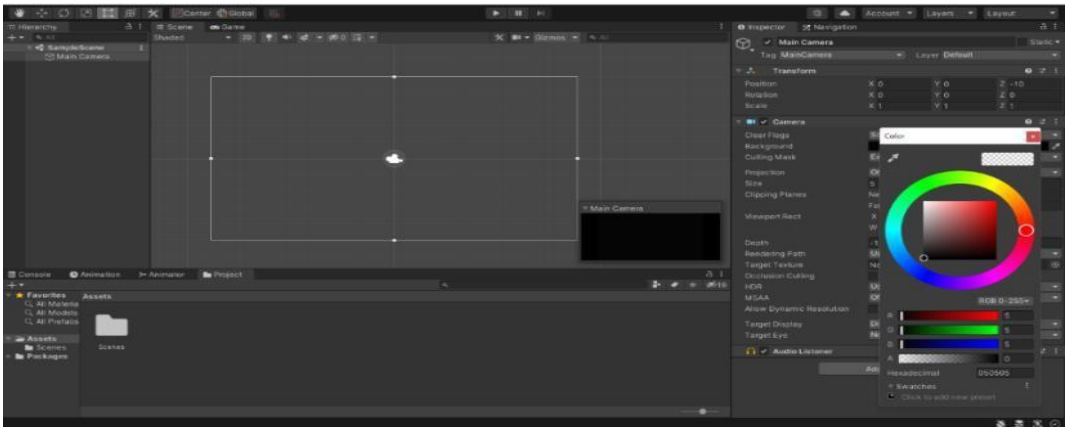
Create a new 2D project in unity.



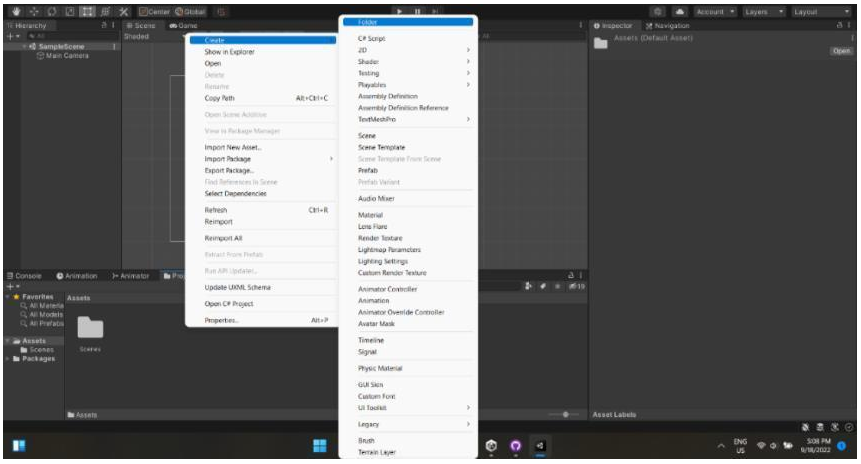
1. A new unity project will be created. Now click on Main Camera in Hierarchy.



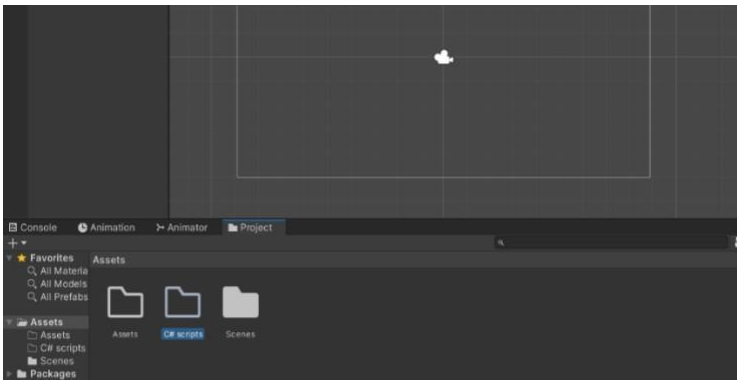
2. In inspector tab, check for background in camera component and change the background color to black.

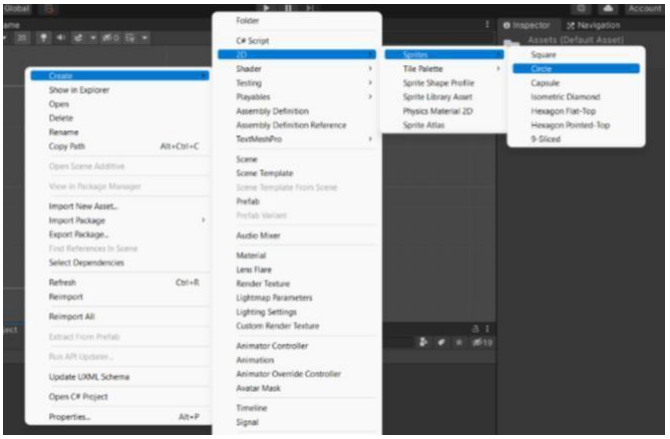
3. 

4. Now, click on the assets folder in project tab. Right click and click on create -> folder. Create 2 Folder



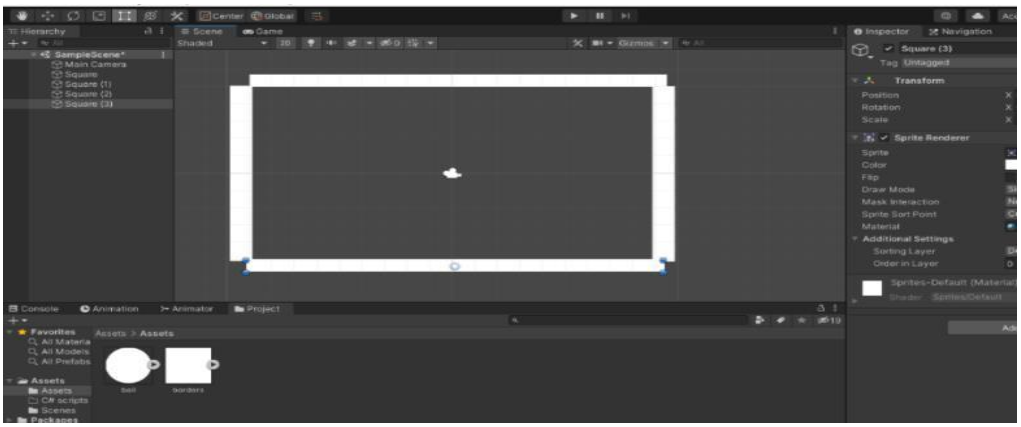
5. Create 2 folder with name “Assets” and “C# scripts”.



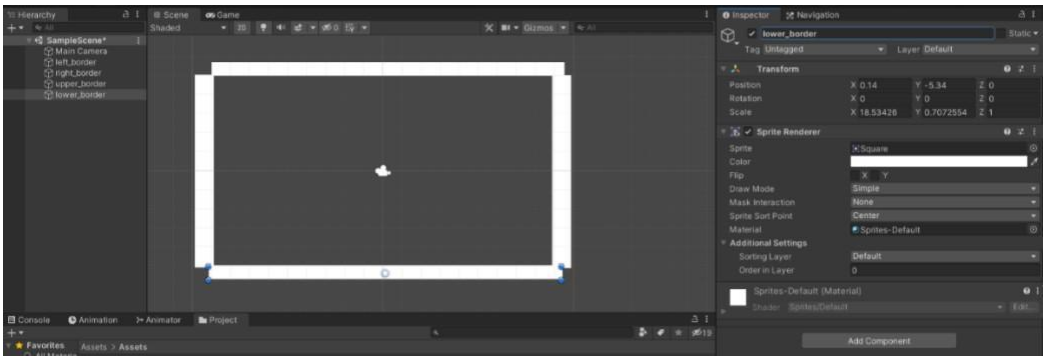


6. Now go into the assets folder. Right click -> create -> 2D -> Sprites -> Circle. And Right click -> create -> 2D -> Sprites -> Square.

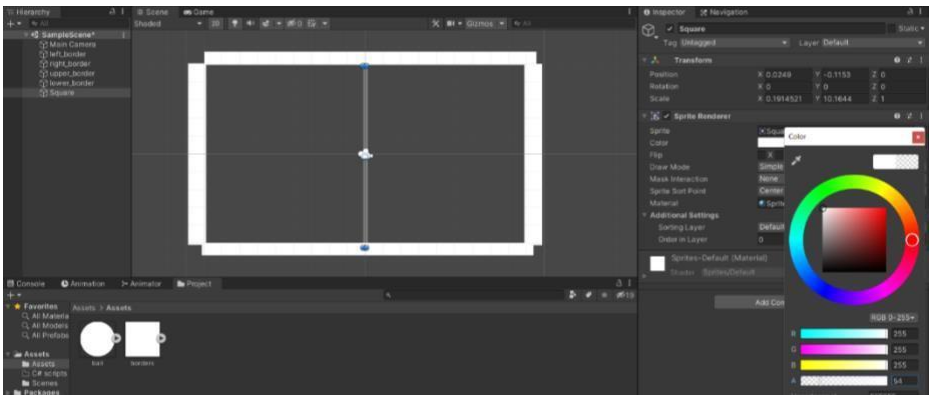
7. Now drag and drop the square in Scene view. And adjust the size and arrange the 4 Squares as shown in the screenshot below.



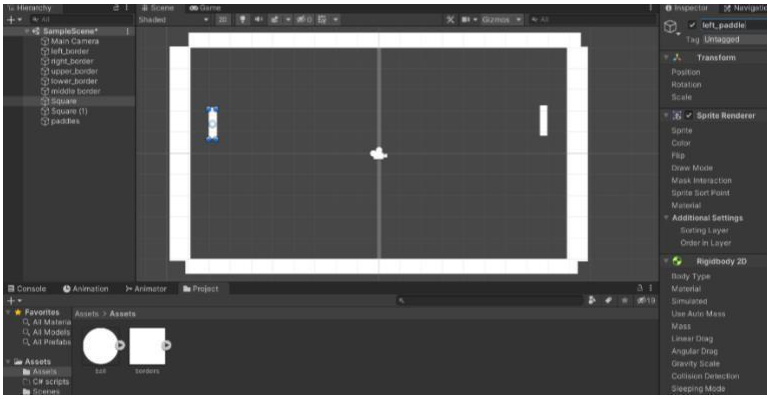
8. And rename them accordingly



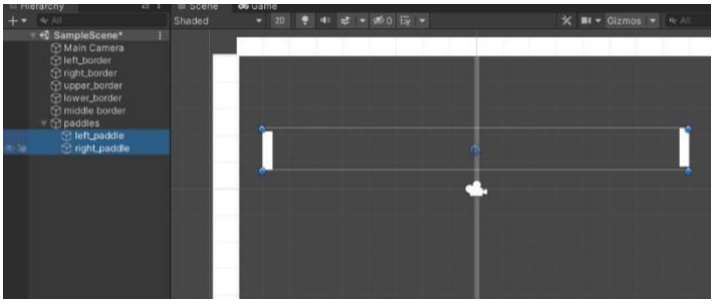
9. Drag one more square in the center of the scene view as shown in the screenshot below. In the inspector tab -> In sprite rendered component -> select color -> and set the Alpha value to 54.



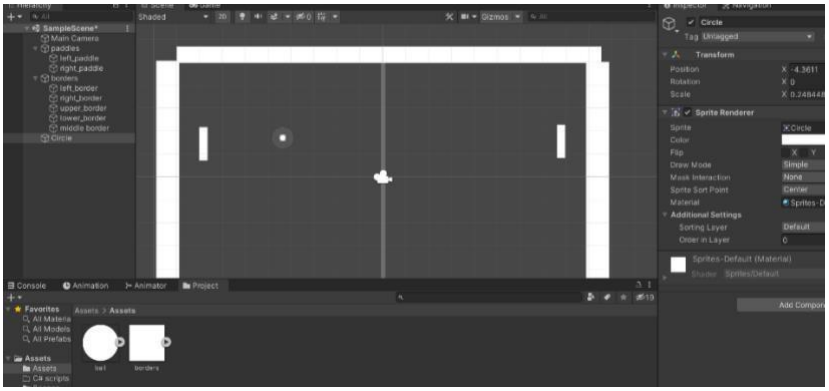
10. Drag two more square for paddle and adjust as shown in the screenshot.



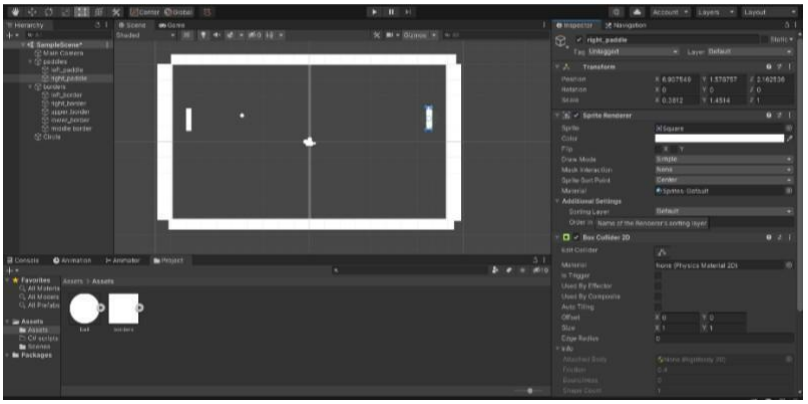
11. Create an empty game object by right clicking in the hierarchy and selecting Create empty. Name it paddle. And drag the two paddle objects in the paddle game object.



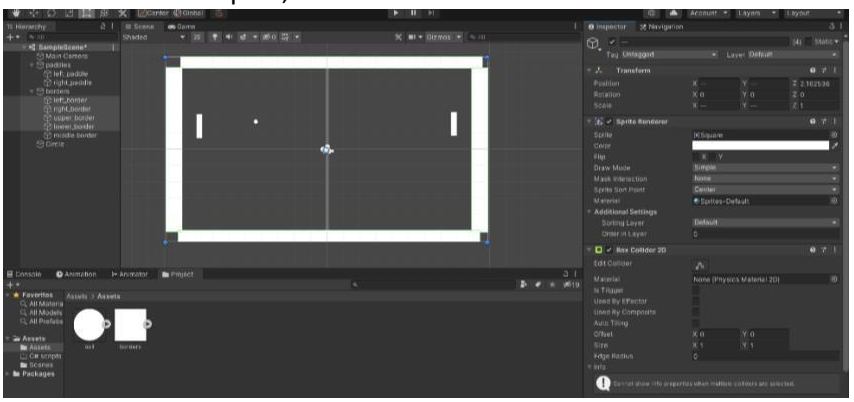
12. Do the same as step 11. But Name it as Borders and drag the border objects in the Border game object.
13. Drag a circle sprite from assets to scene view.



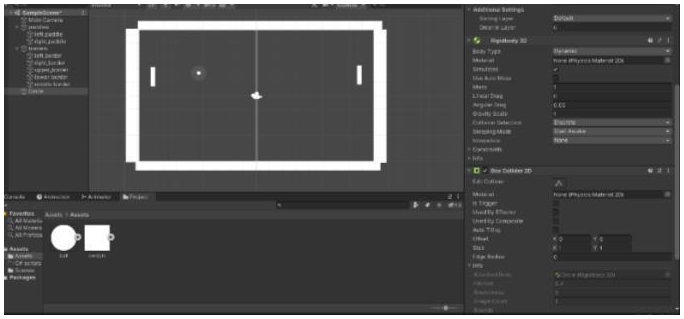
14. Add the box collider 2d and rigidbody 2d component to the paddles. Freeze the x and z constraint in the rigidbody2d.



15. Do the same as Step 14, but for all the four borders.

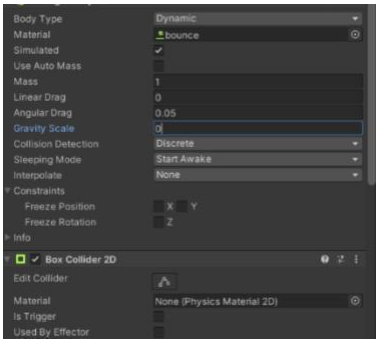


16. Do the same as Step 14, but for the ball (circle) game object.



17. In the assets folder. Right click -> 2D -> physics material 2d. And give it friction of 0 and bounciness of 1.

18. Add the bounce physics material to the rigid body 2d -> material of the ball.



19. In the C# scripts folder. Right click-> C# script. And name it player. And do it one more time. And name it ball.
20. Add the following code for player script.

```
using System.Collections;
using
System.Collections.Generic;
using UnityEngine;
public class Player : MonoBehaviour{
  // Start is called before the first frame
  update public float speed;
  public Rigidbody2D rigidbody2D;
  public Vector3 startposition;
  private float movement;
  public bool
  isPlayer1; void
  Start(){
    startposition = transform.position;}
  // Update is called once per frame
  void Update(){
    if (isPlayer1){
      movement = Input.GetAxisRaw("Vertical");}
    else{
      movement = Input.GetAxisRaw("Vertical2");}
    rigidbody2D.velocity = new Vector2(0, movement * speed);}
  public void Reset(){
    rigidbody2D.velocity = Vector2.zero;
    transform.position = startposition;}}
```

21. Add the following code for ball script.

```
using System.Collections;
using
System.Collections.Generic;
using UnityEngine;
public class ball : MonoBehaviour{
```

```
public Rigidbody2D
rigidbody; public float speed;
public Vector3 startPosition;
// Start is called before the first frame
update void Start(){
   Play();}
public void Reset(){
   rigidbody.velocity =
   Vector2.zero;
   transform.position =
   startPosition; Play();}
void Play(){
   float x = Random.Range(0, 2) == 0 ? -1 : 1;
   float y = Random.Range(0, 2) == 0 ? -1 : 1;
   rigidbody.velocity = new Vector2(speed * x, speed *
   y);}}
```

Add three more scripts in C# scripts folder and name it AddScore1, AddScore2 and Score. Add the following code for Score script.

```
using System.Collections;
using
System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class Score : MonoBehaviour{
  // Start is called before the first frame
  update public Text player1Score;
  public Text player2Score;
  private int player1Points =0; private int player2Points =0; int addScore1 = 0;
  int addScore2 = 0;
  void Start(){
     Debug.Log("Score Added");
     player1Score.text = "Player 1 :" + 0;
     player2Score.text = "Player 2 :" + 0;}
  public void updateScore1(int playerScore1){
     player1Score.text = "Player 1 : " + playerScore1.ToString();}
  public void updateScore2(int playerScore2){
     player2Score.text = "Player 2: " + playerScore2.ToString();}
  public void AddScore1(int points){
     addScore1 += points;
     updateScore1(addScore1);
     ResetPosition();}
  public void AddScore2(int points){
     addScore2 += points;
     updateScore2(addScore2);
     ResetPosition();}
  private void ResetPosition(){
     GameObject.Find("Ball").GetComponent<ball>().Reset();
     GameObject.Find("left_paddle").GetComponent<Player>().Reset();
     GameObject.Find("right_paddle").GetComponent<Player>().Reset();}}
```

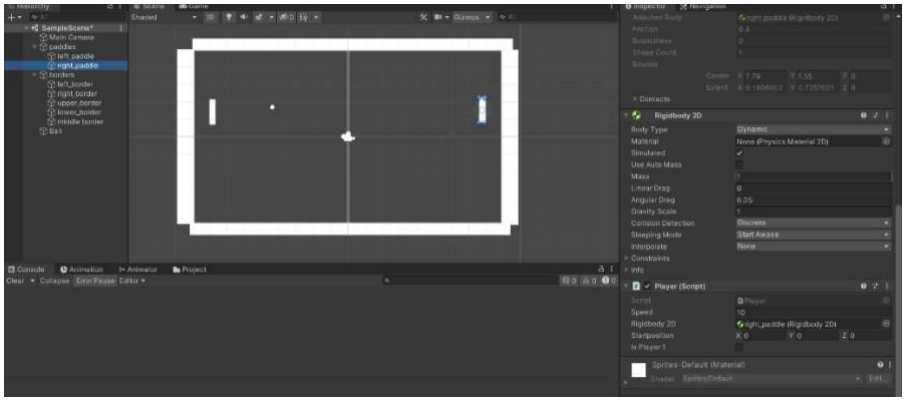
22. Add the following code for AddScore1 script.

```
using System.Collections;
using
System.Collections.Generic;
using UnityEngine;
public class addScore1 : MonoBehaviour{
  private void OnCollisionEnter2D(Collision2D collision){
    GameObject.Find("Player1").GetComponent<Score>().AddScore1(1);}}
```

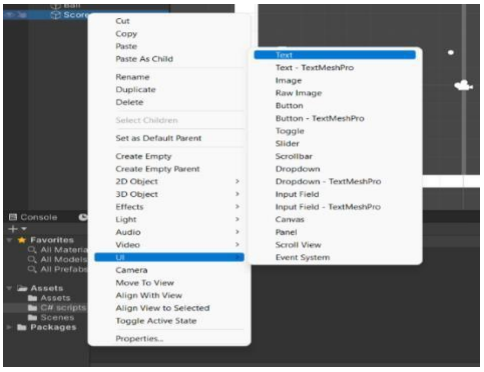
23. Add the following code for AddScore2 script.

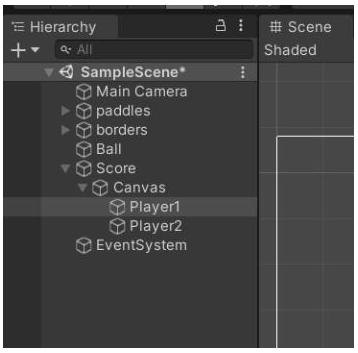
```
using System.Collections;
using System.Collections.Generic; using UnityEngine;
public class addScore2 : MonoBehaviour{
  private void OnCollisionEnter2D(Collision2D collision){
    GameObject.Find("Player2").GetComponent<Score>().AddScore2(1);}}
```

24. Drag and drop the player scripts to the left and right paddle. And give it a speed of 10. And check the checkbox of is player 1 of the left paddle.

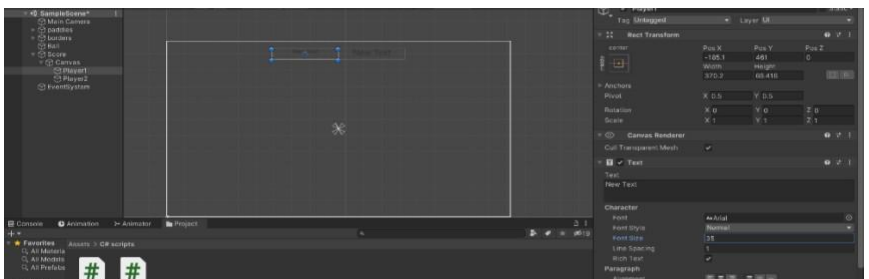


25. Drag and drop the ball script to the ball(circle) game object.
    And give it a speed of 5. And select the rigidbody2d of the ball in the rigid body 2d of the ball script.
26. Create a empty game object in the game object hierarchy. And name it score. Right click -> UI -> Text. Do it twice. And name it player1 and player2.

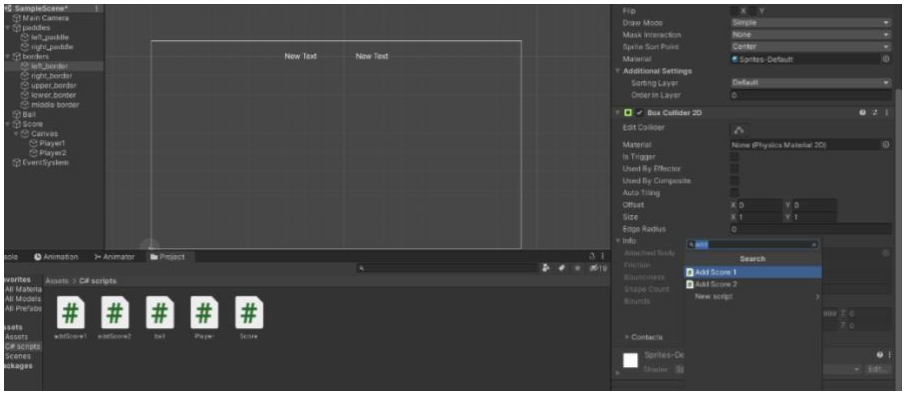




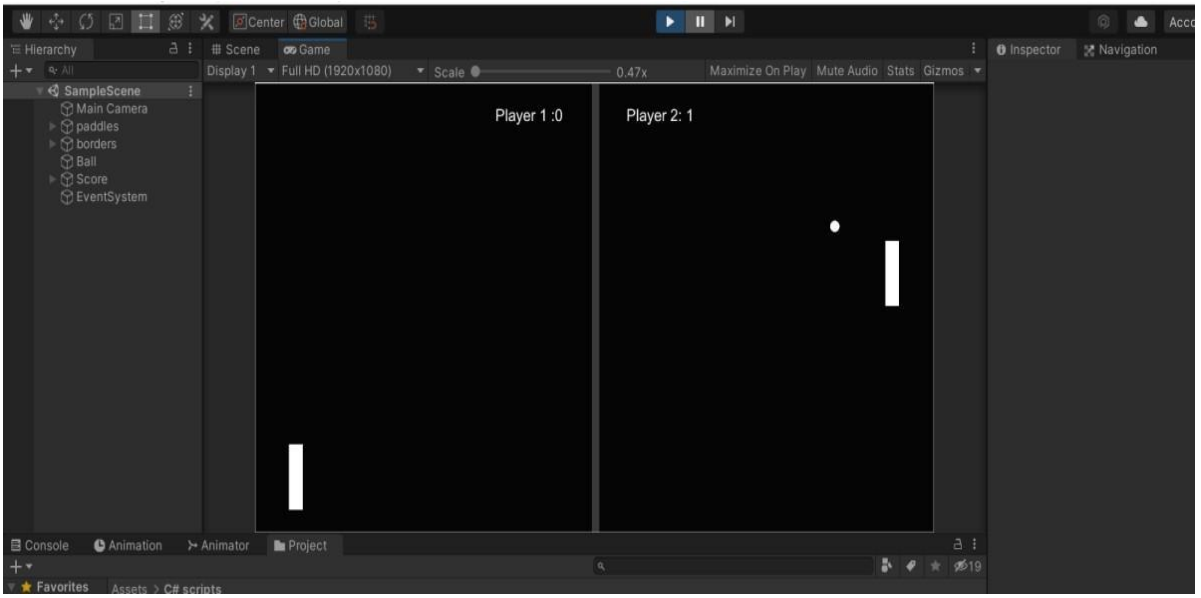
27. Arrange the score text as shown in the screenshot. Increase the font size and center the text.



28. Drag and drop the Score script to Player 1 in player 1 Score. And player 2 in player 2 Score.



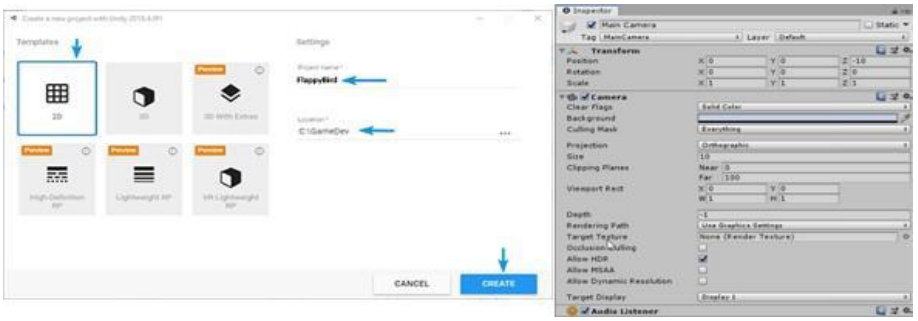
29. Drag and drop the AddScore1 to left border. And AddScore2 to right border.
30. Click on Play button And Play the game :)



Practical 8

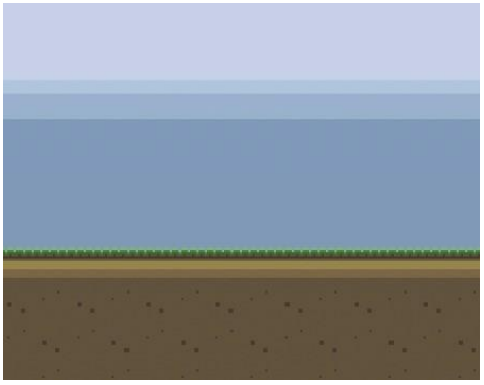
Aim: Unity 2D Flappy Bird Tutorial

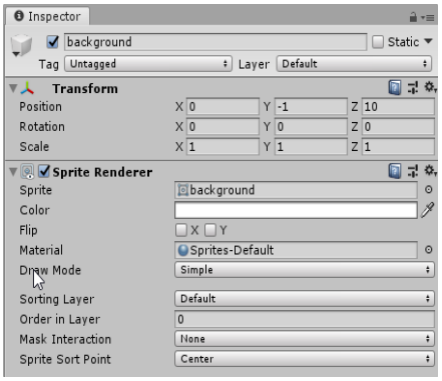
We will name it Flappy Bird, select any location like C:\GameDev, select 2D and click Create Project:



Creating the sky and ground setting x,y= 1

Creating the Bird: code and to towards right all the time and obstacale and reverse invoke repeating






```
using UnityEngine;
using System.Collections;
public class Bird : MonoBehaviour {
    void Start () {
    }
    void Update () {
    }
}
```

```
using UnityEngine;
using System.Co
public class Bird : MonoBehaviour {
   public float speed = 2;
void Start () {
getcomponent<rigidbody2d>().velocity=velocity=vector2.right=speed;
}
void update(){
}}
```

```
using UnityEngine;
using System.Collections;
public class Obstacle : MonoBehaviour {
    void Start () {
    }
    void Update () {
    }
}
```

```
using UnityEngine;
using System.Collections;
public class Obstacle : MonoBehaviour {
   public float speed = 0;
   public float switchTime = 2;
   void Start() {
      GetComponent<Rigidbody2D>().velocity = Vector2.up * speed;
      InvokeRepeating("Switch", 0, switchTime);
   }
   void Switch() {
      GetComponent<Rigidbody2D>().velocity *= -1;
   }
}
```

If we press Play then we can see our obstacle moving up- and downwards: set the Scale.Y property to 1 We can add as many obstacles with as many different speed and switchTime properties as we want:

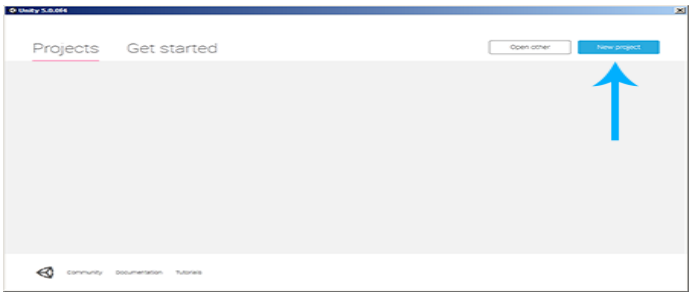
Conclusion:

Finally, our game is successfully generated

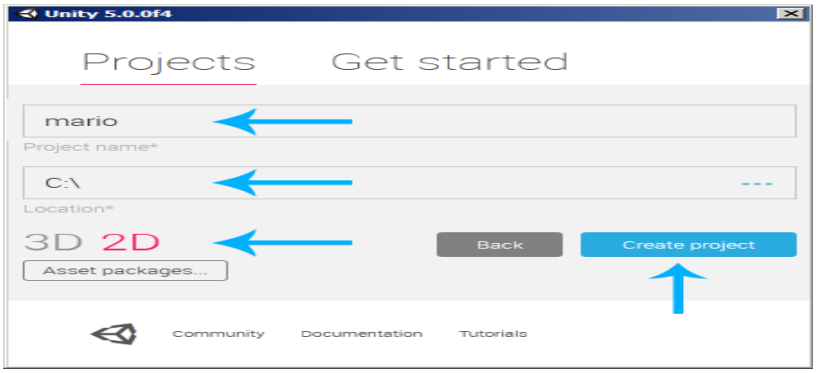
Practical 9

Aim: Mario Game

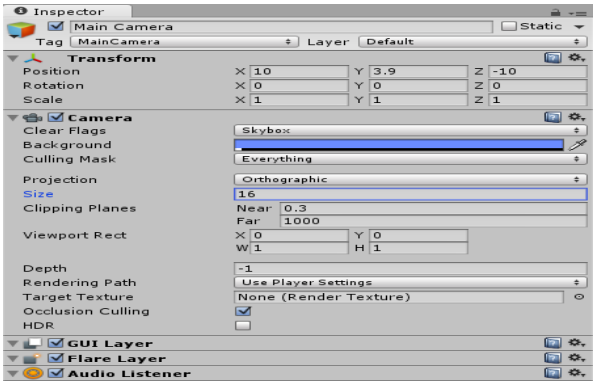
Let's get to it. We will start Unity and select **New Project**:



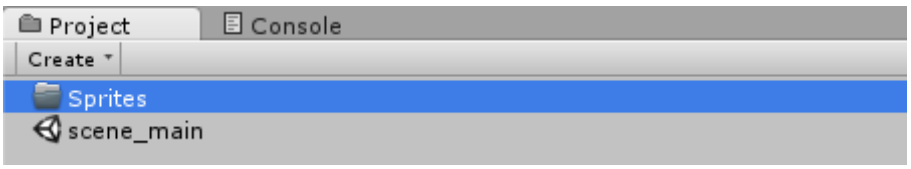
We will name it **mario**, select any location like **C:**, select **2D** and click **Create Project**:



Now we can select the **Main Camera** in the **Hierarchy** and use the typical blue **Background Color** ***(red=107, green=140, blue=255)***, adjust the **Size** and the **Position** so the game looks right later on

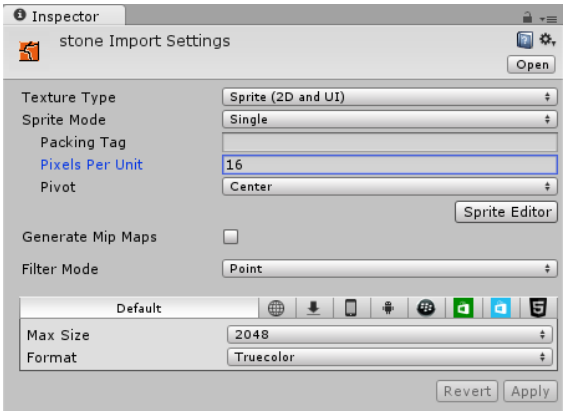


Before we start, let's create a new **Sprites** folder in the **Project Area**:

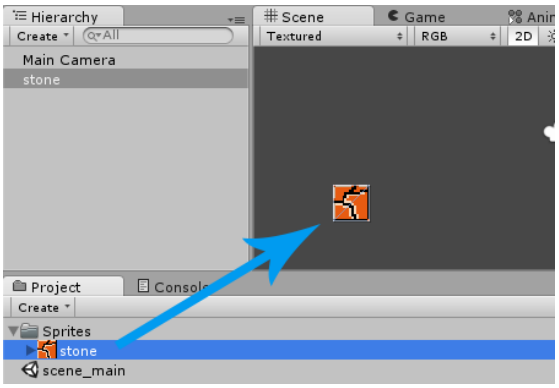


This is where we will put all our Sprites
The Stone Image We will begin by drawing a 16 x 16 pixels **Stone** image in a drawing tool like Paint.NET:

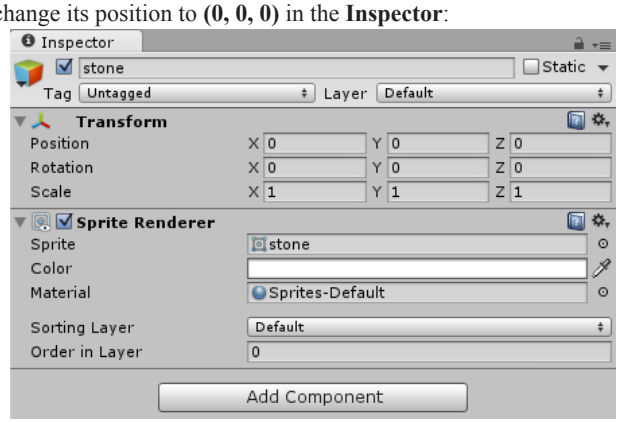
*Note: right click on the image, select **Save As...** and save it in the project's **Assets/Sprites** folder.*
Now we can select the image in the **Project Area** and then modify the **Import Settings** in the **Inspector**



Now we can drag the image from the **Project Area** into the **Scene** in order to add it to the game world:

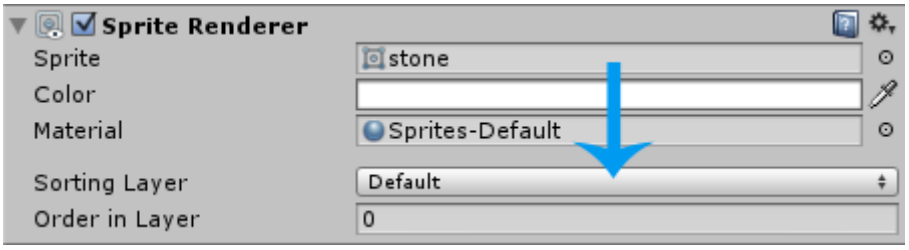


Let's select the stone in the **Hierarchy** and change its position to **(0, 0, 0)** in the **Inspector**:

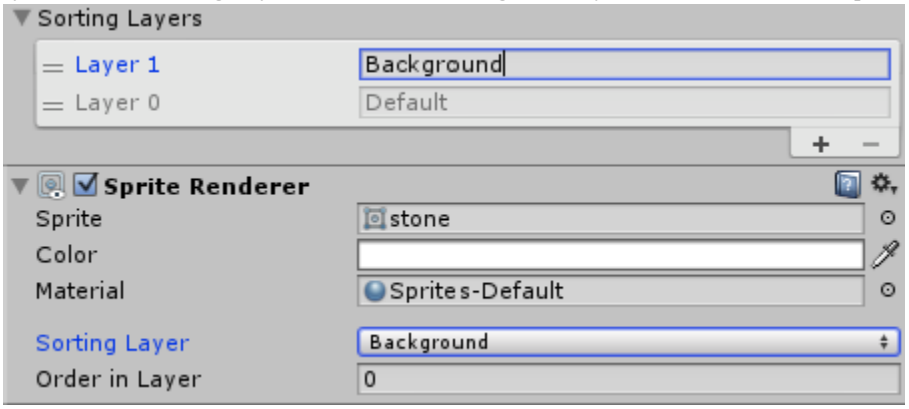


The Sorting Layer

Let's tell Unity that the stones are always supposed to be in the background. This is what **Sorting Layer**'s are used for. We can change the stone's Sorting Layer if we take a look at the Sprite Renderer component in the Inspector:

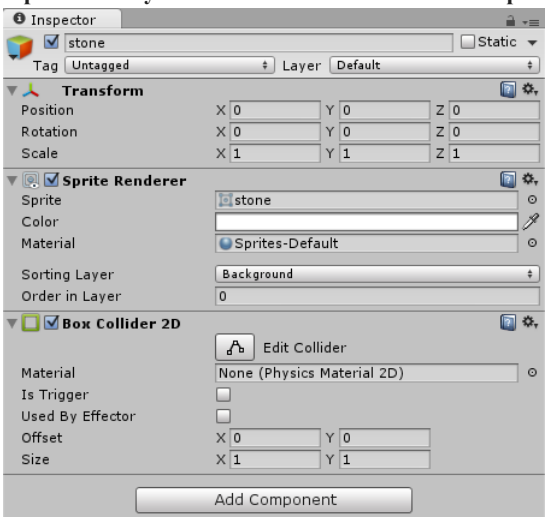


Let's select **Add Sorting Layer..** from the **Sorting Layer** list, then add a **Background** layer and move it to the first position like shown below:



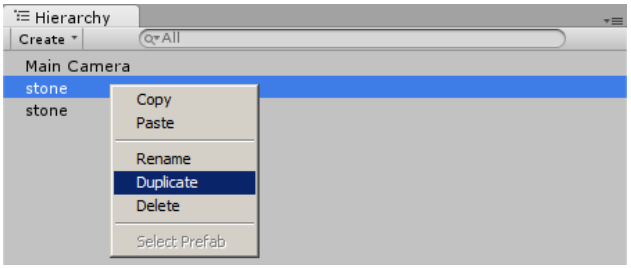
Right now the stone is only an image, nothing more. It's not part of the physics world, Mario won't be able to walk on top of it or anything. We will need to add a **Collider2D** to make it part of the physics world, which means that things will collide with the stone instead of falling or walking right through it.

We can add a **Collider2D** by selecting **Add Component->Physics2D->Box Collider 2D** in the **Inspector**:



Now the stones are part of the physics world, it's that easy.

Now we can add more stones to our Scene. We will take a look over to the **Hierarchy** where we right click the current stone and then select **Duplicate**:

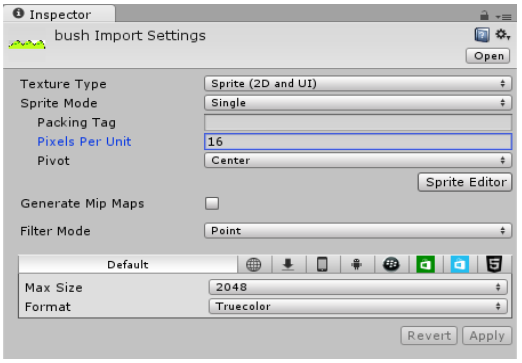


Let's repeat this work flow until we have two rows of 21 stones *(one row at y=0 and one below at y=-1)*. The important part is to always position them at rounded coordinates like **(2, 0)** and never at **(2.003, 0.005)**. Here is how it looks if we press **Play**:

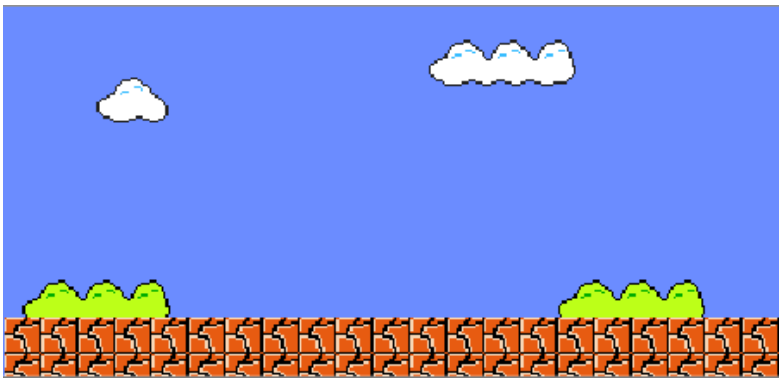
Bushes and Clouds

Just like in the original Super Mario Bros. game we will also add a few bushes and clouds to the background. As usual, we will begin by drawing the images:

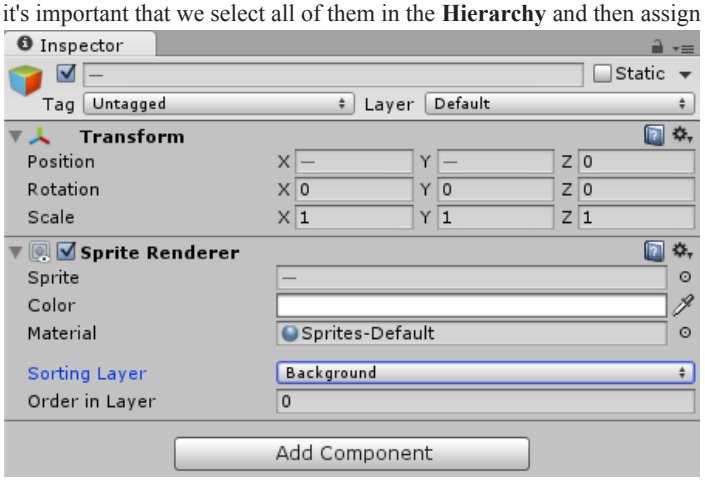
We will use the following **Import Settings** for all the images:



Now we can drag the Sprites from the **Project Area** into the **Scene** and position them where we like them to be:



Since they are part of the background, it's important that we select all of them in the **Hierarchy** and then assign the **Background** Sorting Layer again:

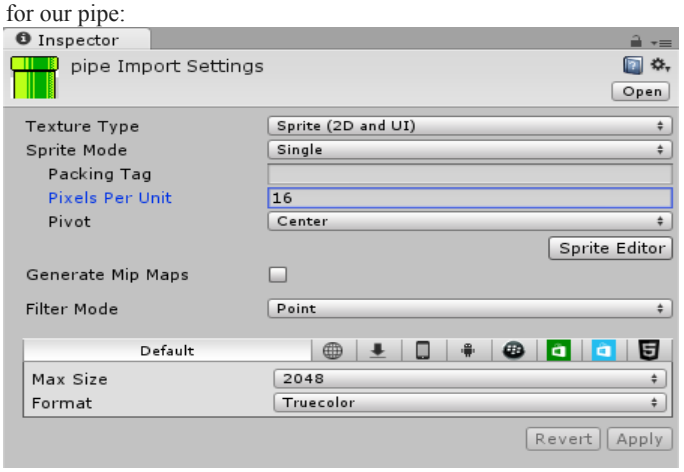


.

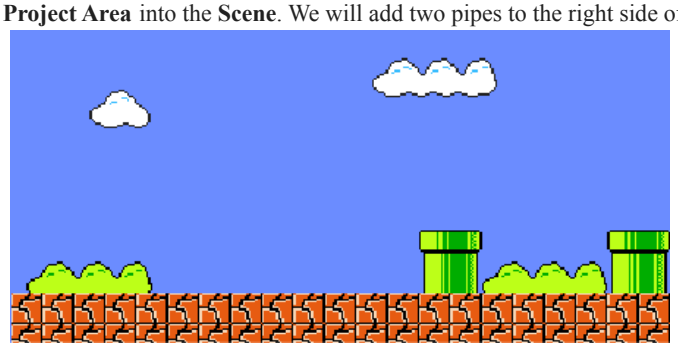
Pipes

Those green pipes are a big part of Super Mario Bros., so let's draw one:

And here are the **Import Settings** for our pipe:

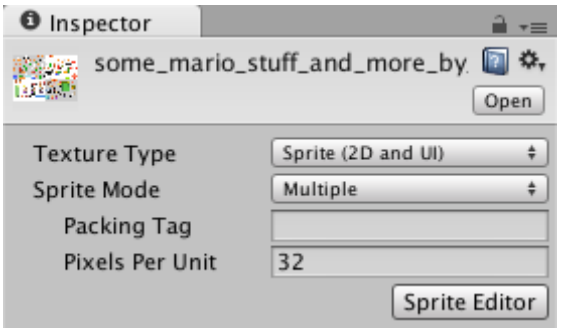


Now we can drag the Sprite from our **Project Area** into the **Scene**. We will add two pipes to the right side of the world:

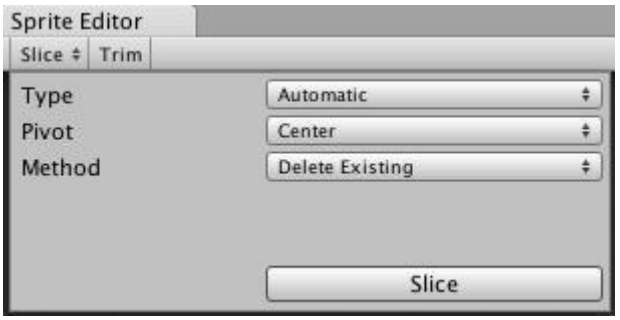


Now lets start with Mario Character.

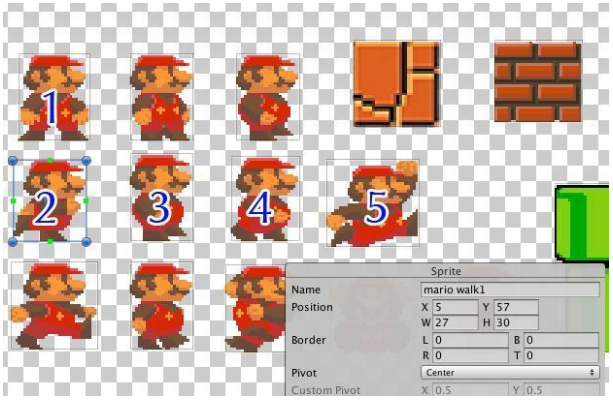
Create a sprites folder and import sprite sheet here. Select Sprite and make the Sprite Mode multiple. If you get another sprite, Pixels Per Unit value may be different bits it's 32 for my sprite.



Open sprite editor.



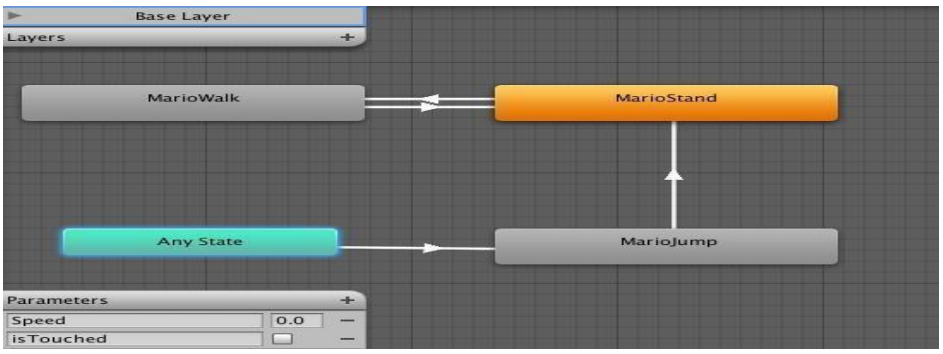
We will use these 5 frames to create animation for Mario.



Click on frames and rename them.

- Mario_Stand
- Mario_Walk_1
- Mario_Walk_2
- Mario_Walk_3
- Mario_jump

Apply changes and back to scene. Expand sprite file and you will see your 5 reanamed sprites.
Select 3 walk sprites and drag to scene.Give a name of your walk animation. Press play and test Mario's movement. Now open animation window and create new clip. Drag Mario stand and change Samples value with one because we have only one frame. Do it for Mario jump to.
Open the Animator Window:



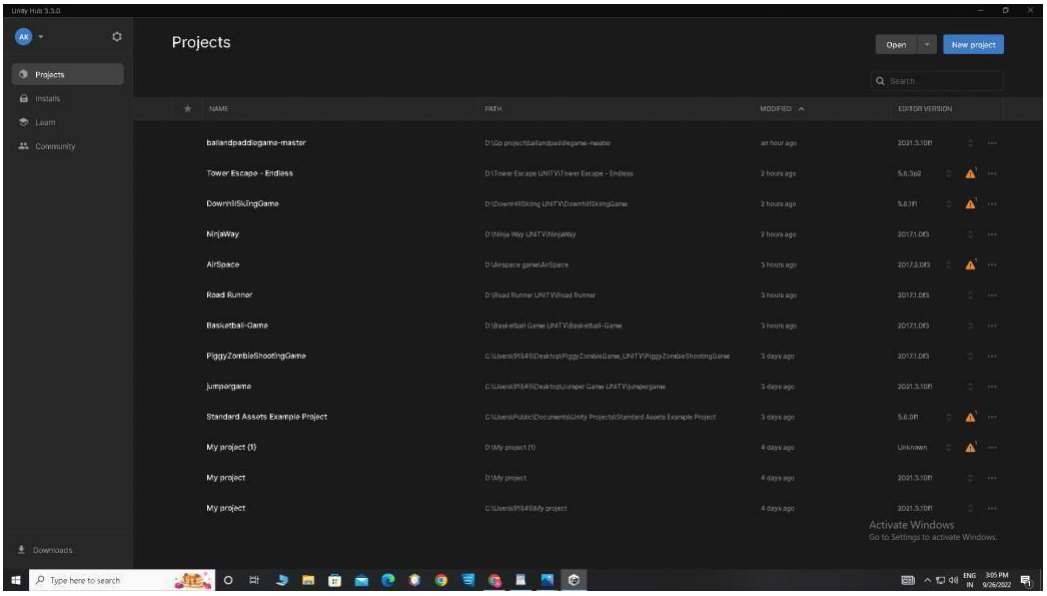
You see animation names not sprites here. Animate the charcter actions here.
Add Rigidbody2D and a BoxCollider2D to your Mario Object. You can create a ground with sprite textures. Add Ground Layer to ground.
Now create a C# Script for mario.

```
public float speed = 1.0f;
        public float jumpSpeed = 0.5f; public
        LayerMask groundLayer; private Animator
        marioAnimator; private Transform gCheck;
        private float scaleX = 1.0f; private float scaleY
        = 1.0f;
        void Start () {
        marioAnimator = GetComponent();
                gCheck = transform.FindChild("GCheck"); } void FixedUpdate ()
        {float mSpeed = Input.GetAxis("Horizontal");
        marioAnimator.SetFloat("Speed", Mathf.Abs(mSpeed));
                bool isTouched = Physics2D.OverlapPoint(gCheck.position, groundLayer); if
                (Input.GetKey(KeyCode.Space)){if (isTouched){
rigidbody2D.AddForce(Vector2.up * jumpSpeed, ForceMode2D.Force); isTouched = false;        }}
                marioAnimator.SetBool("isTouched", isTouched); if (mSpeed &gt;
                0){transform.localScale = new Vector2(scaleX, sc} else if (mSpeed
                &lt; 0){
                        transform.localScale = new Vector2(-scaleX, scaleY);                    } this.rigidbody2D.velocity
                = new Vector2(mSpeed * speed, this.rigidbody2D.velocity.y)
```

Practical 10

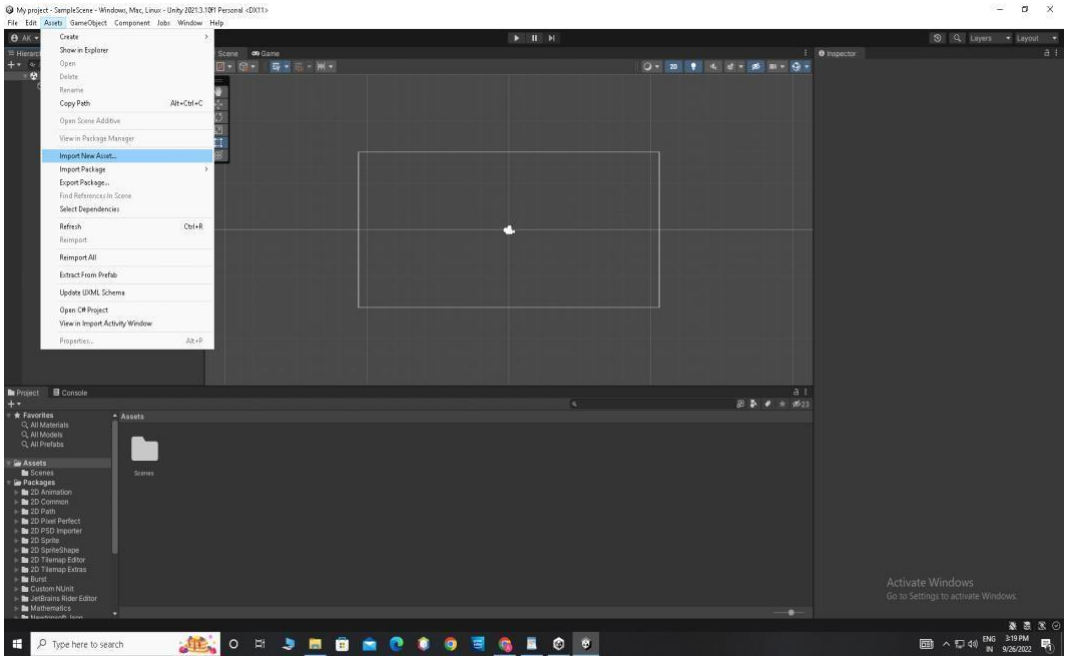
Aim- Ball and Paddle Game 2d in Unity

Step 1- Click on new project



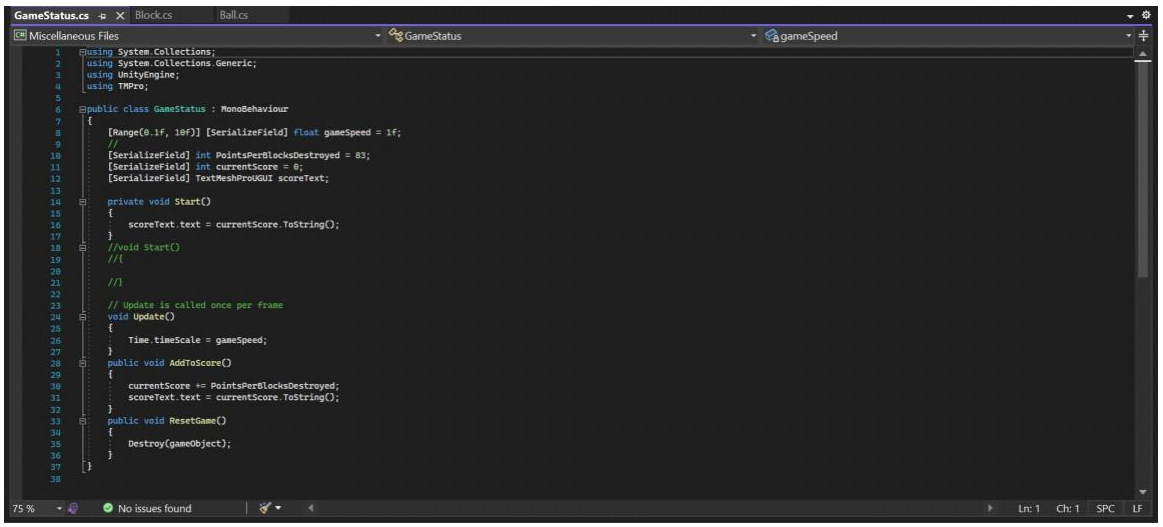
Step 2- Click on 2d project name it and choose the save

tep 3- Go to asset and import new asset

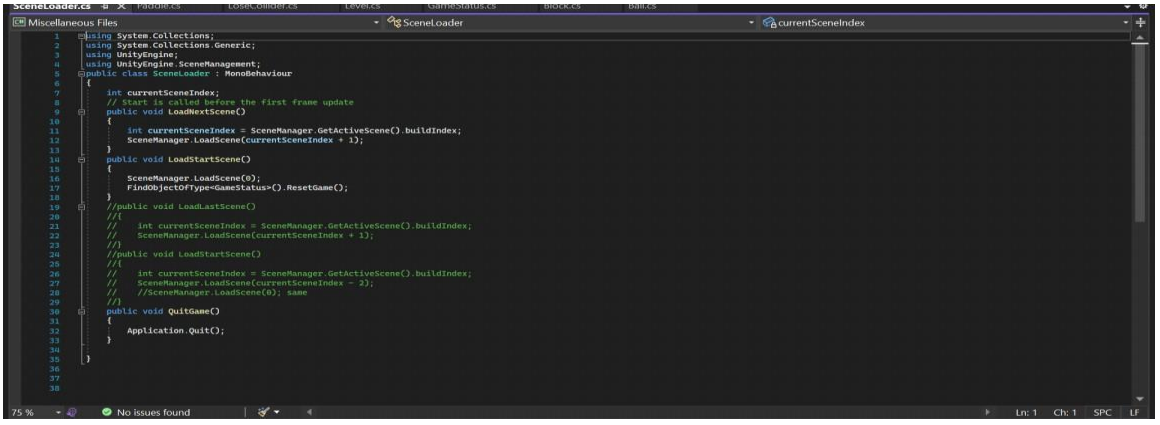


Step 4- Select asset folder and click on Import

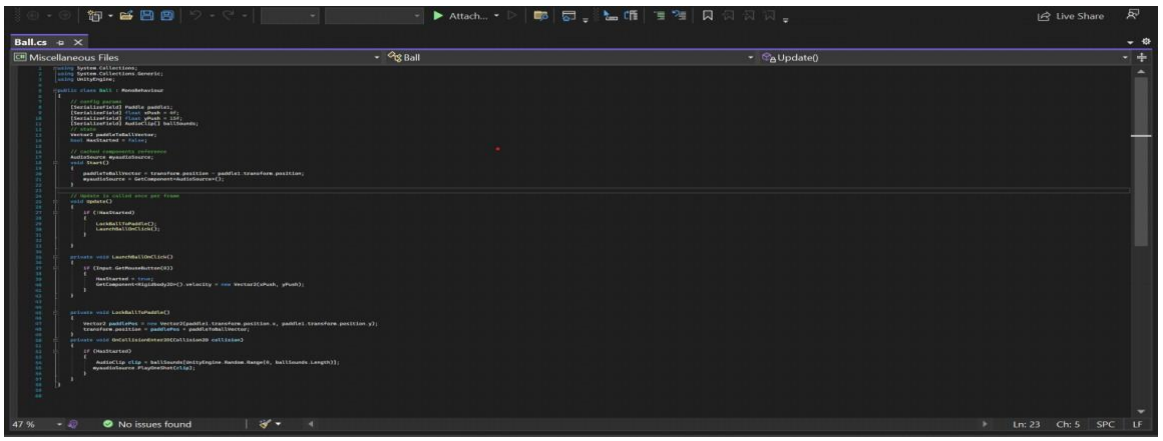
tep 5- Code for Game-start



Step 6- Code for Screen Loader

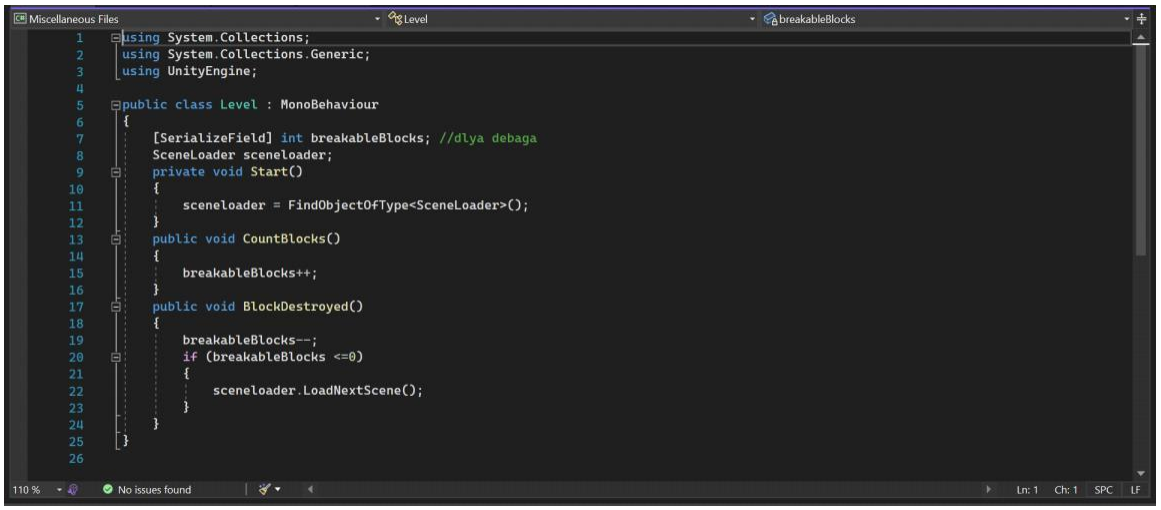


7- Code for the Ball



Step 8- Code for the Block

Step 9- Code for Level



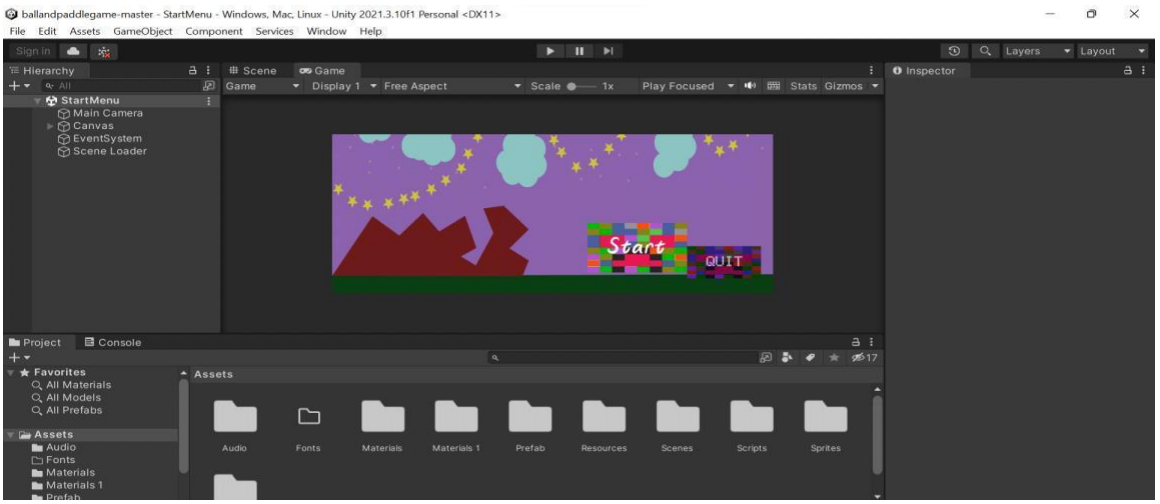
Step 10- Code for Loose collider

```
LoseCollider.cs | Level.cs | GameStatus.cs | Block.cs | Ball.cs
Miscellaneous Files | LoseCollider | OnTriggerEnter2D(Collider2D collision)

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
6  public class LoseCollider : MonoBehaviour
7  {
8      // Start is called before the first frame update
9      public void OnTriggerEnter2D(Collider2D collision)
10     {
11         int currentSceneIndex = SceneManager.GetActiveScene().buildIndex;
12         SceneManager.LoadScene("Game Over");
13     }
14 }
15

133 % | No issues found | Ln: 1 | Ch: 1 | SPC | LF
```

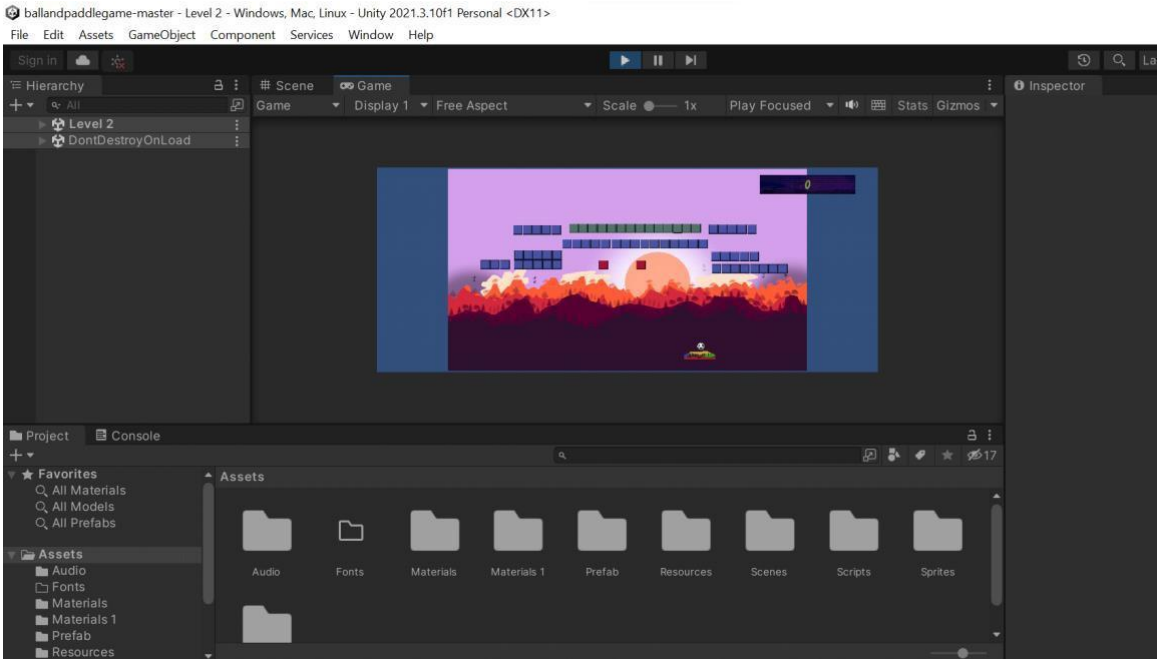
Step 11- Code for Paddle

```
Miscellaneous Files | Paddle | minX

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Paddle : MonoBehaviour
6  {
7
8      // configuration parameters
9      [SerializeField] float minX = 1f;
10     [SerializeField] float maxX = 15f;
11     [SerializeField] float screenWidthInUnits = 16f;
12
13     // Use this for initialization
14     void Start()
15     {
16
17     }
18
19     // Update is called once per frame
20     void Update()
21     {
22         Vector3 mousePos = Camera.main.ScreenToWorldPoint(Input.mousePosition);
23         //Debug.Log("ScreenToWorldPoint mousePosition.x: " + mousePos.x);
24         Vector2 paddlePos = new Vector2(mousePos.x, transform.position.y, transform.position.z);
25         paddlePos.x = Mathf.Clamp(paddlePos.x, minX, maxX);
26         transform.position = paddlePos;
27     }
28 }

91 % | No issues found | Ln: 1 | Ch: 1 | SPC | LF
```

Step 12- Press play button to run the project



Step 13- Control the paddle using arrow keys



Step 14- When you drop the ball from the paddle the game will be over