SIES College of Arts, Science & Commerce

RISE WITH EDUCATION

NAAC REACCREDITED - 'A' GRADE

ISO 9001 : 2008

**S.I.E.S College of Arts, Science and CommerceSion(W), Mumbai – 400 022.**

# CERTIFICATE

This is to certify that Miss. Dipali Gupta Roll No. **Tcs2222026** Has successfully completed the necessary course of experiments in the subject of **Game Programming** during the academic year **2022 – 2023** complying with the requirements of **University of Mumbai**,

for the course of **T.Y. BSc. Computer Science [Semester-3]**

Prof. In-Charge
Miss. Soni Yadav
**(Game Programming)**

Examination Date:
Examiner's Signature & Date:

Head of the Department
**Prof. Manoj Singh**

College Seal
And
Date

INDEX

# GAME PROGRAMMING

Dipali Gupta

026

## 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 Prac1_GP
{
 public partial class Form1 : Form
 {
 Microsoft.DirectX.Direct3D.Device device;

 public Form1()
 {
 InitializeComponent();
 InitDevice();

 }

 private void Form1_Load(object sender, EventArgs e)
 {
```

```
 }
 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.DarkOliveGreen, 0, 1);
device.Present();
 }

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

Output:

# PRACTICAL 2:

## Aim:

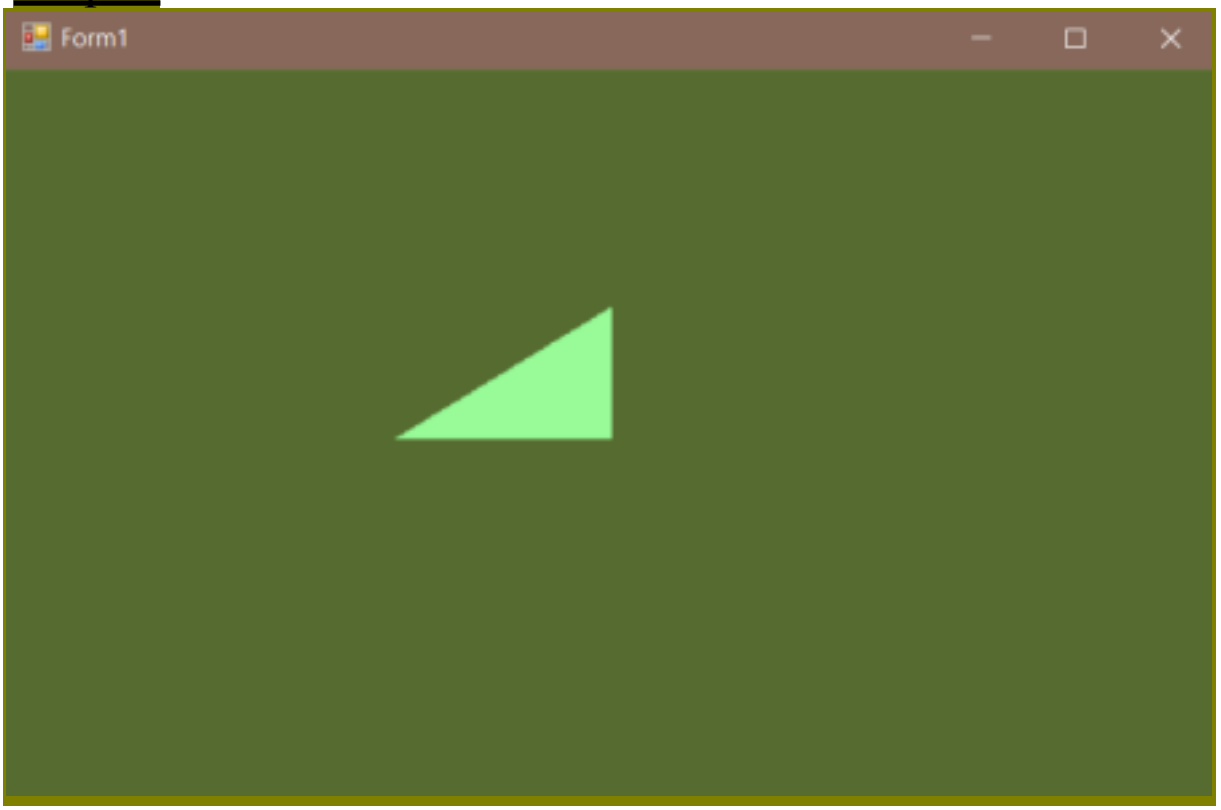
Buffers, Shaders and HLSL (Draw a triangle using Direct3D 11

## 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_GP
{
 public partial class Form1 : Form
 {
 private Device device;
 private CustomVertex.PositionColored[] vertex =
new CustomVertex.PositionColored[3];
 public Form1()
 {
 InitializeComponent();

 }


 private void Form1_Load(object sender,
EventArgs e)  {
 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, 20),  new Vector3(), new Vector3(0, 1, 0));
 device.RenderState.Lighting = false;
 vertex[0] = new CustomVertex.PositionColored(new
```

```
Vector3(),  Color.PaleGreen.ToArgb());
 vertex[1] = new CustomVertex.PositionColored(new Vector3(3, 0,
0),  Color.PaleGreen.ToArgb());
 vertex[2] = new CustomVertex.PositionColored(new Vector3(0, 3,
0),  Color.PaleGreen.ToArgb());
 }

 private void Form1_Paint_1(object sender,
PaintEventArgs e)  {
 device.Clear(ClearFlags.Target, Color.DarkOliveGreen, 1, 0);
device.BeginScene();
 device.VertexFormat = CustomVertex.PositionColored.Format;


device.DrawUserPrimitives(PrimitiveType.TriangleList,
vertex.Length / 3, vertex);
 device.EndScene();
 device.Present();
 }
 }
}
```

Output:



# PRACTICAL 3:

## Aim:

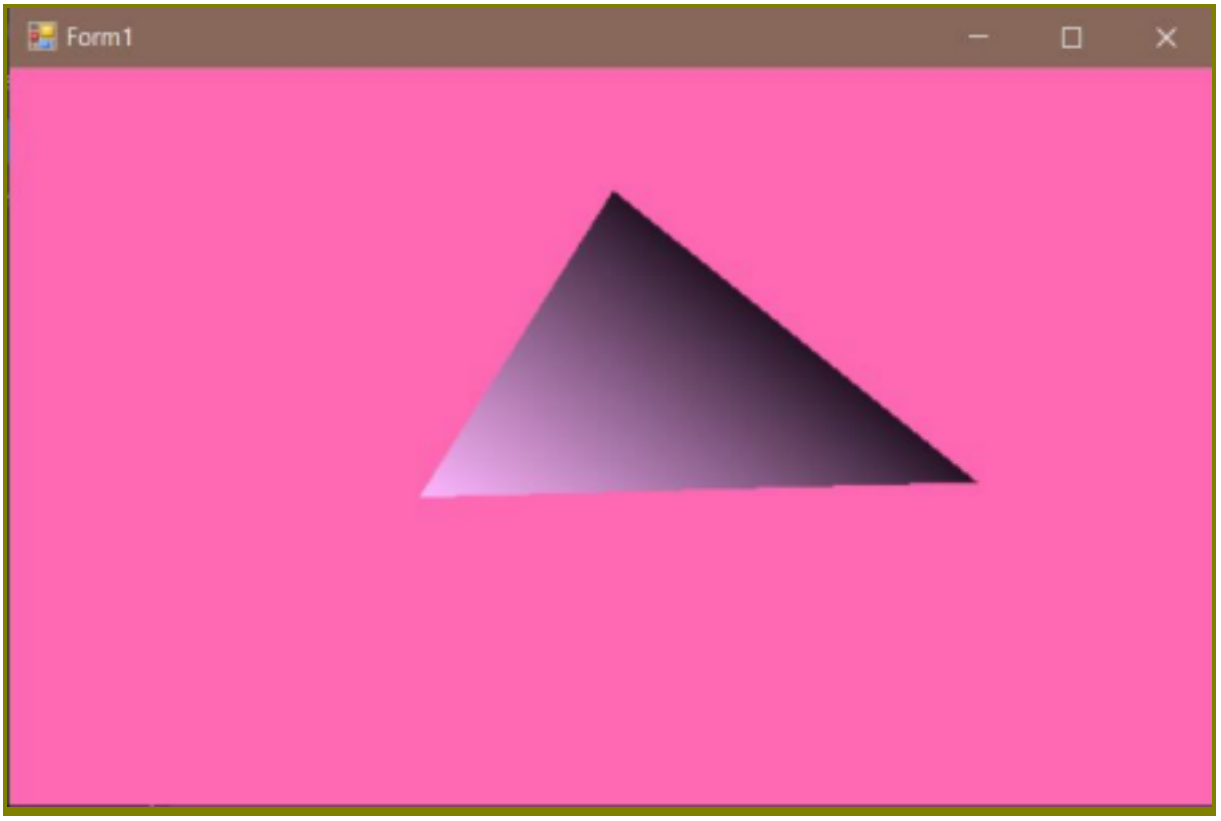
Buffers, Shaders and HLSL (Draw a triangle using Direct3D 11

## 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 Prac3_GP
{
 public partial class Form1 : Form
 {
 private Device device;
 private CustomVertex.PositionNormalColored[] vertex
= new CustomVertex.PositionNormalColored[3];

 public Form1()
 {
 InitializeComponent();
 }
 private void Form1_Load(object sender, EventArgs e)
 {
 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,
20), new Vector3(), new Vector3(0, 1, 0));
 device.RenderState.Lighting = false;
 vertex[0] = new CustomVertex.PositionNormalColored(new Vector3(0,
5, 2),  new Vector3(1, 0, 1), Color.Violet.ToArgb());
```

```
 vertex[1] = new CustomVertex.PositionNormalColored(new Vector3(-5,
-1, 0),  new Vector3(1, 0, 1), Color.Violet.ToArgb());
 vertex[2] = new CustomVertex.PositionNormalColored(new Vector3(2,
-1, 5),  new Vector3(-1, 0, 1), Color.Violet.ToArgb());

 device.RenderState.Lighting = true;
 device.Lights[0].Type = LightType.Directional;
 device.Lights[0].Diffuse = Color.White;
 device.Lights[0].Direction = new Vector3(0.8f, 0, -1);
device.Lights[0].Enabled = true;

 }

 private void Form1_Paint(object sender, PaintEventArgs e)
 {
 device.Clear(ClearFlags.Target, Color.HotPink, 1, 0);
 device.BeginScene();
 device.VertexFormat = CustomVertex.PositionNormalColored.Format;
device.DrawUserPrimitives(PrimitiveType.TriangleList, vertex.Length /
3,  vertex);
 device.EndScene();
 device.Present();

 }
 }
}
```

Output:




# PRACTICAL 4:

## Aim:

Lightning (Programmable Diffuse Lightning 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 Prac4_GP
{
 public partial class Form1 : Form
 {
 private Device device;
```

```
 private CustomVertex.PositionTextured[] vertex = new
CustomVertex.PositionTextured[3];
 private Texture texture;

 public Form1()
 {
 InitializeComponent();
 }

 private void Form1_Load(object sender, EventArgs e)
 {
 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,
20), new Vector3(), new Vector3(0, 1, 0));
 device.RenderState.Lighting = false;

  vertex[0] = new CustomVertex.PositionTextured(new Vector3(0, 1, 1), 0,
0);   vertex[1] = new CustomVertex.PositionTextured(new Vector3(-5, -5,
1), -1,  0);
 vertex[2] = new CustomVertex.PositionTextured(new Vector3(5, -5, 1),
0, - 1);
 texture = new Texture(device, new
Bitmap("C:\\Users\\SATYAM\\source\\repos\\Prac4_GP\\Prac4_
GP\\im.jpg"), 0,  Pool.Managed);
 }

 private void Form1_Paint(object sender,
PaintEventArgs e)  {
 device.Clear(ClearFlags.Target, Color.Maroon, 1, 0);
 device.BeginScene();
 device.SetTexture(0, texture);
 device.VertexFormat = CustomVertex.PositionTextured.Format;
device.DrawUserPrimitives(PrimitiveType.TriangleList, vertex.Length /
3,  vertex);
 device.EndScene();
 device.Present();
 }
 }
```

}

## Output:


## PRACTICAL 5:

## Aim:

Loading models(image or File) using DirectX 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;
using Microsoft.DirectX.Direct3D;

namespace Practical5_GP
{
 public partial class Form1 : Form
```

```
{
Microsoft.DirectX.Direct3D.Device device;
Microsoft.DirectX.Direct3D.Font font;
Texture texture;

public Form1()
{
InitializeComponent();
InitializeDevice();
InitFont();
LoadTexture();
}
private void InitializeDevice()
{
PresentParameters pp = new PresentParameters();
pp.Windowed = true;
pp.SwapEffect = SwapEffect.Discard;
device = new Device(0, DeviceType.Hardware, this,
CreateFlags.HardwareVertexProcessing, pp);
}

private void LoadTexture()
{
texture = TextureLoader.FromFile(device,
"C:\\Users\\SATYAM\\source\\repos\\Practical5_GP\\Practical5_GP\\user.
png", 400, 400,  1, 0, Format.A8R8G8B8, Pool.Managed, Filter.Point,
Filter.Point,  Color.Transparent.ToArgb());
}

private void InitFont()
{
System.Drawing.Font f = new
System.Drawing.Font("COMIC", 26f,  FontStyle.Bold);
font = new Microsoft.DirectX.Direct3D.Font(device, f);
}

private void Render()
{
device.Clear(ClearFlags.Target, Color.Orange, 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.LightCyan);
 font.DrawText(s, "Dipali Gupta", new Point(0, 0), Color.Beige);
s.End();
 }
 device.EndScene();
 device.Present();
 }

 private void Form1_Load(object sender, EventArgs e)
 {

 }

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

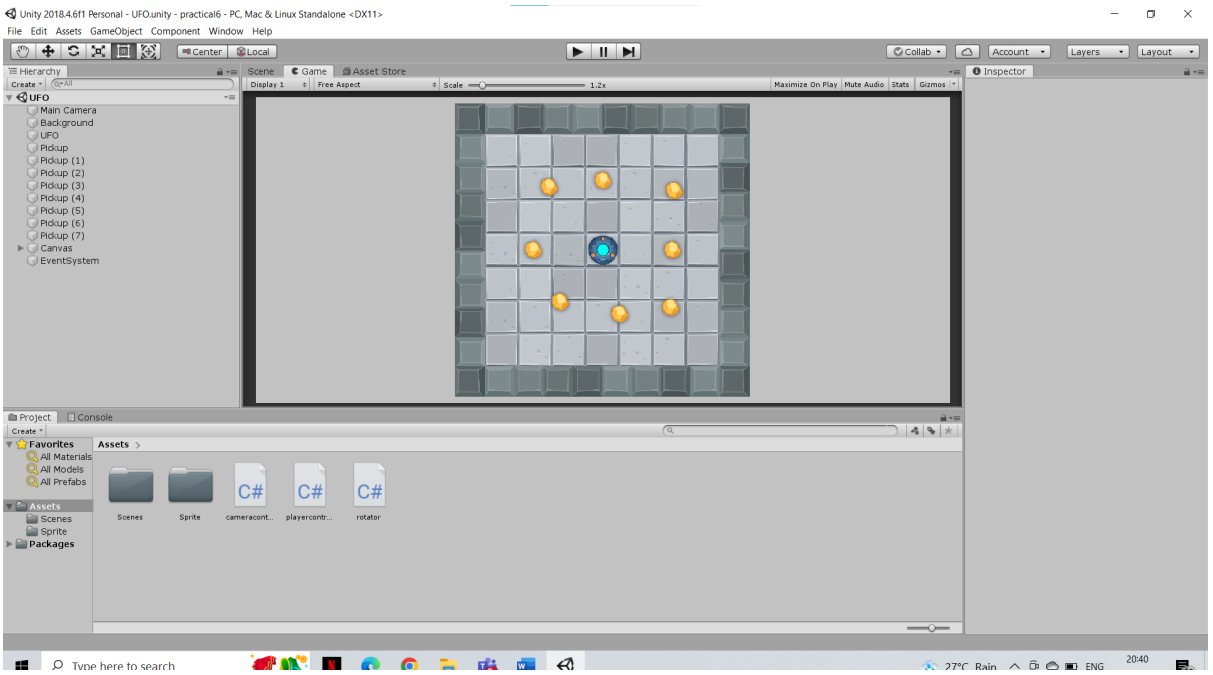
Output:


PRACTICAL 6:

## Aim:

Using unity Create 2D UFO game by downloading asset from asset store

## Code:




***playercontroller.cs***

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class playercontroller : MonoBehaviour
{
    public Text winText;
    public Text countText;
    public int count=0;
    private Rigidbody2D rbd;
    public float speed;
    // Start is called before the first frame update
     void Start()
    {
        rbd = GetComponent<Rigidbody2D>();
    }
    // Update is called once per frame
    void FixedUpdate()
    {
```

```
        float moveHorizontal = Input.GetAxis("Horizontal");
        float moveVertical = Input.GetAxis("Vertical");
        Vector2 movement = new Vector2(moveHorizontal, moveVertical);
        rbd.AddForce(movement * speed);
    }
    void OnTriggerEnter2D(Collider2D other)
    {
        if (other.tag == "pickup")
        {
            other.gameObject.SetActive(false);
            count++; SetCountText();
        }
    }
    void SetCountText()
    {
        countText.text = "Count" + count.ToString();
        if (count == 8)
        {
            winText.text = "You Win";
        }
    }
}
```

***cameracontroller.cs***

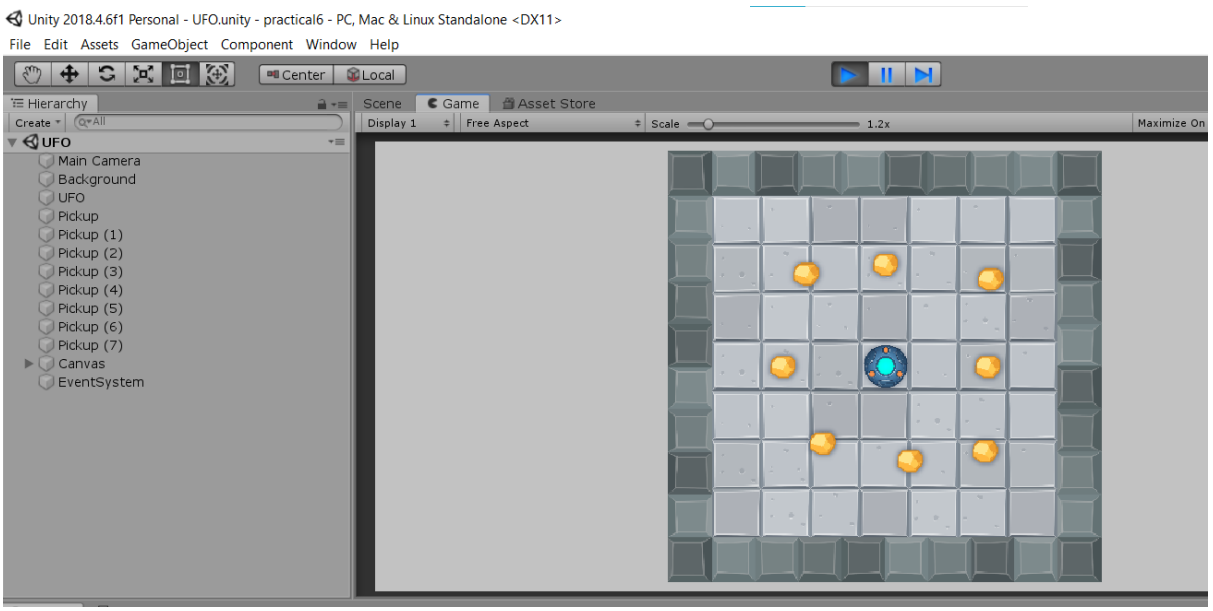
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class cameracontroller : MonoBehaviour
{
    public GameObject player; private Vector3 offset;
    // Start is called before the first frame update
    void Start(){
        offset = transform.position - player.transform.position;
    }
    // Update is called once per frame
    void LateUpdate()
    {
        transform.position = player.transform.position + offset;
```

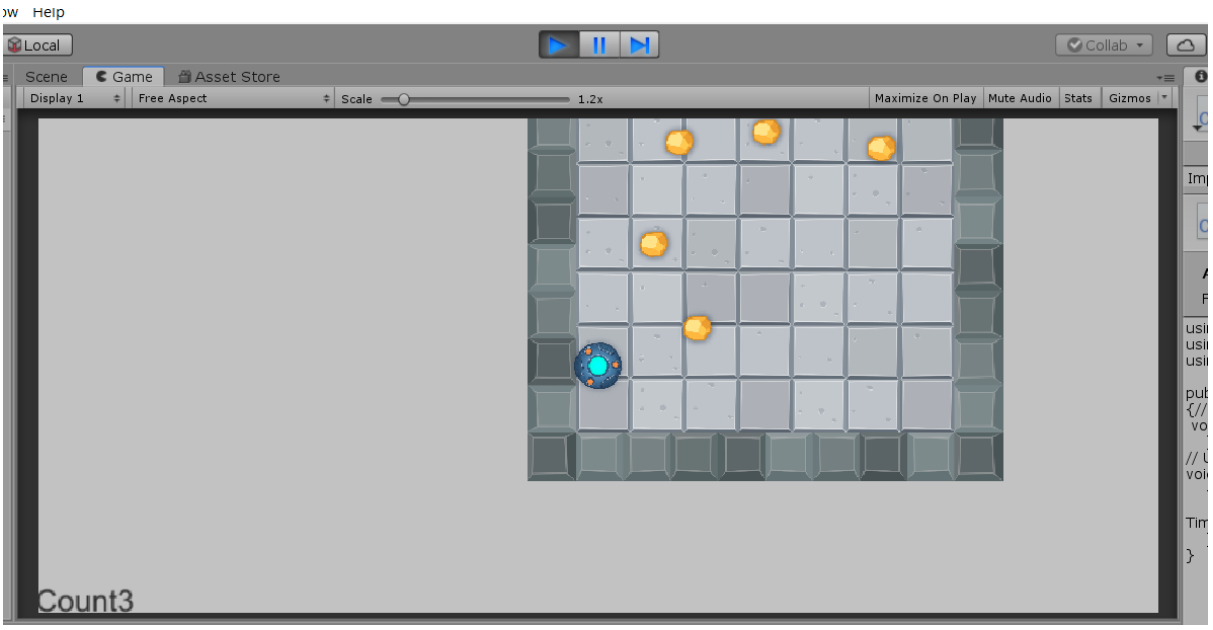
```
        }
}
```

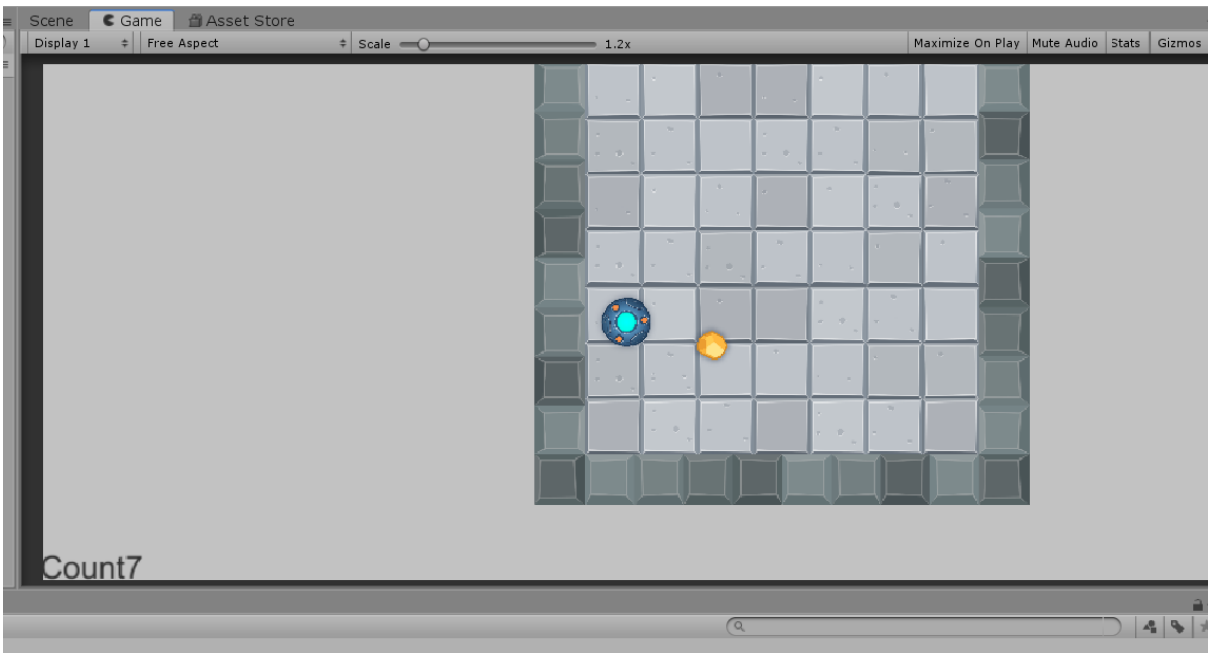
***rotator.cs***

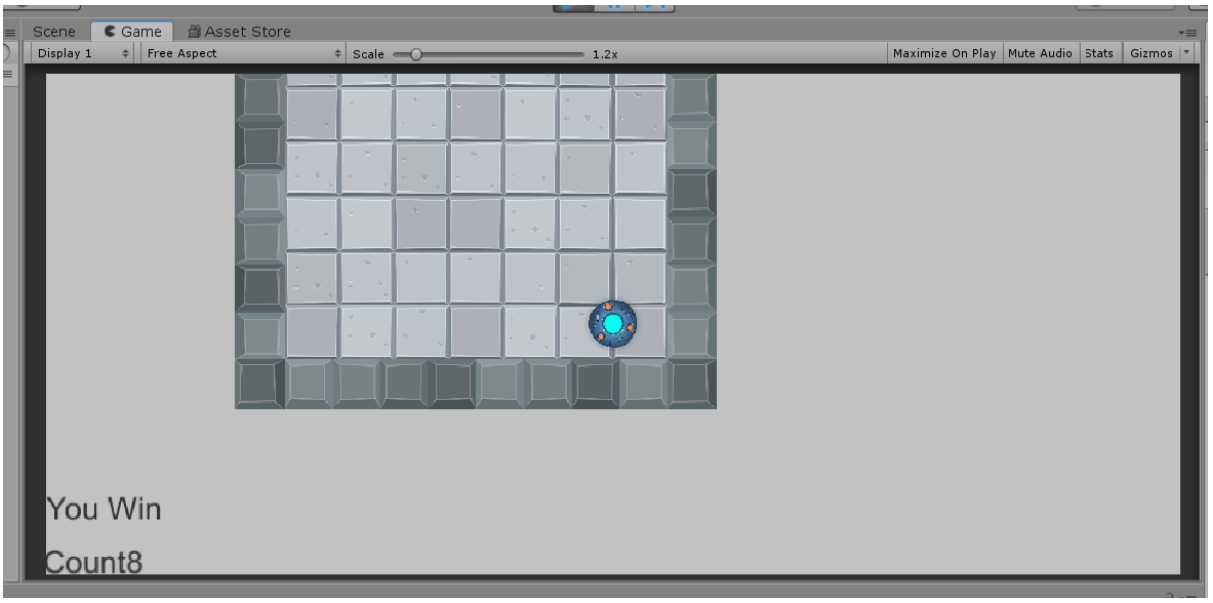
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class rotator : MonoBehaviour
{// Start is called before the first frame update
 void Start(){
    }
// Update is called once per frame
void Update()
    {
        transform.Rotate(new Vector3(0, 0, 45) * Time.deltaTime);
    }
}
```

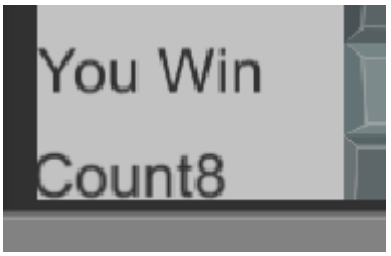
Output:




Help
Local
Collab
Scene
Game
Asset Store
Display 1
Free Aspect
Scale
1.2x
Maximize On Play
Mute Audio
Stats
Gizmos
Count3


Scene
Game
Asset Store
Display 1
Free Aspect
Scale
1.2x
Maximize On Play
Mute Audio
Stats
Gizmos
Count7


Scene
Game
Asset Store
Display 1
Free Aspect
Scale
1.2x
Maximize On Play
Mute Audio
Stats
Gizmos
You Win
Count8




# PRACTICAL 7

## Aim:

Using Unity create 3D rolling Ball game.

## Code:

***PlayerController.cs***

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

    public Text winText;
    public Text countText;
    public int count;

    private Rigidbody rb;
    public float speed;
    // Start is called before the first frame update
    void Start()
    {
        count = 0;
        winText.text = "";
        rb = GetComponent<Rigidbody>();
        SetCountText();
    }

    // Update is called once per frame
    void FixedUpdate()
    {
        float moveHorizontal = Input.GetAxis("Horizontal");
        float moveVertical = Input.GetAxis("Vertical");
```

```
        Vector3 movement = new Vector3(moveHorizontal, 0.0f,
moveVertical);
        rb.AddForce(movement * speed);

    }
    void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.CompareTag("PickUp"))
        {
            other.gameObject.SetActive(false);
            count = count + 1;
            SetCountText();
        }
    }
    void SetCountText()
    {
        countText.text = "Count : " + count.ToString();
        if (count >= 8)
        {
            winText.text = "You win!";
        }

    }

}
```

***<u>Camera Controller.cs</u>***

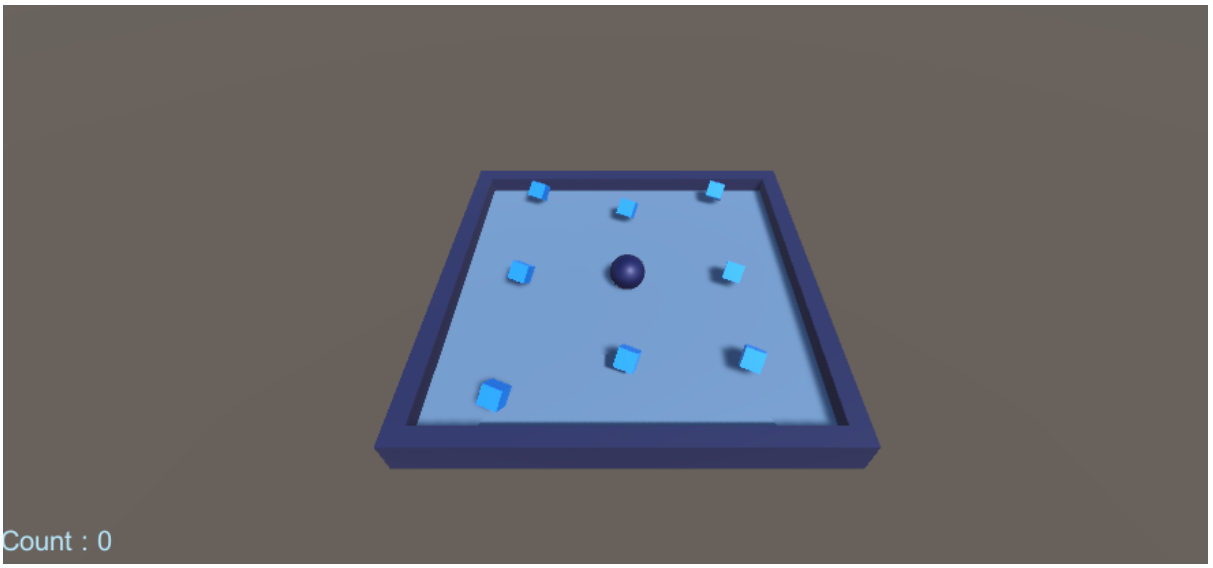
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CameraController : MonoBehaviour
{
    private Vector3 offset;
    public GameObject player;
    // Start is called before the first frame update
    void Start()
    {
        offset = transform.position - player.transform.position;
    }
```

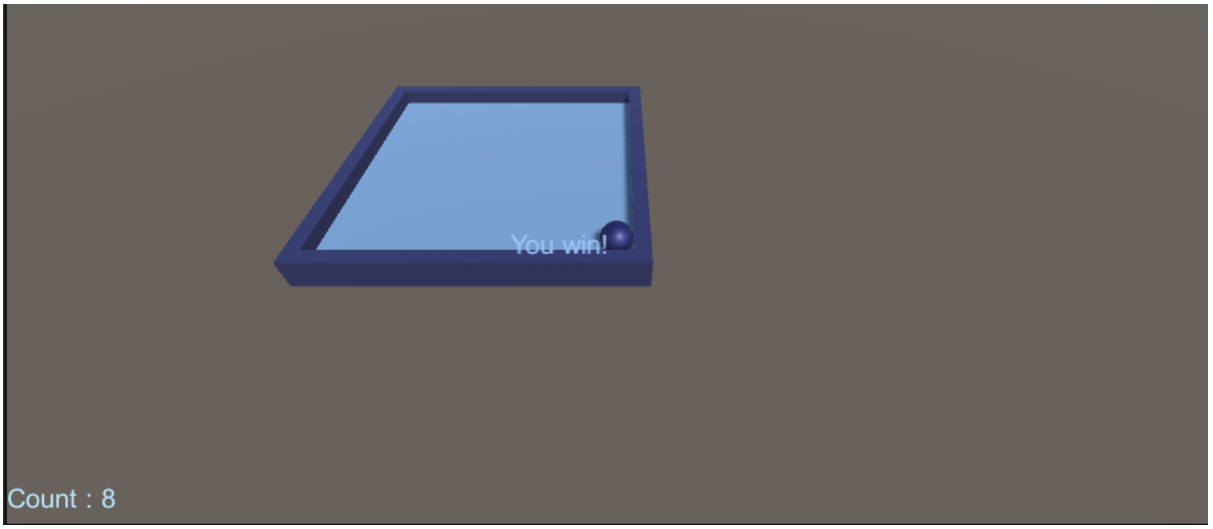
```
    // Update is called once per frame
    void LateUpdate()
    {
        transform.position = player.transform.position + offset;
    }
}
```

***Rotator.cs***

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

public class Rotator : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        transform.Rotate(new Vector3(15, 30, 45) * Time.deltaTime);
    }
}
```

Output:




# PRACTICAL 8

## Aim:

Create a 2D/3D Space Shooter Game

## Code:

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help

Debug | Any CPU | IIS Express (Microsoft Edge)

BGScroller.cs

Miscellaneous Files | BGScroller

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

public class BGScroller : MonoBehaviour
{
    public float scrollSpeed;
    public float tileSizeZ;

    private Vector3 startPosition;

    void Start()
    {
        startPosition = transform.position;
    }
    void Update()
    {
        float newPosition = Mathf.Repeat(Time.time * scrollSpeed, tileSizeZ);
        transform.position = startPosition + Vector3.forward * newPosition;
    }
}
```

File Edit View Git Project Build Debug Test Analyze Tools Ex

Debug Any CPU IIS Exp

Server Explorer Toolbox

DestroyByBoundary.cs BGScroller.cs

Miscellaneous Files DestroyByBo

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

public class DestroyByBoundary : MonoBehaviour
{
    void OnTriggerExit(Collider other)
    {
        Destroy(other.gameObject);
    }
}

```

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search

Debug | Any CPU | IIS Express (Microsoft Edge)

Server Explorer | Toolbox

DestroyByContact.cs | DestroyByBoundary.cs | BGScroller.cs

Miscellaneous Files | DestroyByContact

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

public class DestroyByContact : MonoBehaviour
{
    public GameObject explosion;
    public GameObject playerExplosion;

    public int scoreValue;
    private GameController gameController;

    void Start()
    {
        GameObject gameControllerObject = GameObject.FindWithTag("GameController");
        if(gameControllerObject != null)
        {
            gameController = gameControllerObject.GetComponent<GameController>();
        }
        if (gameControllerObject == null)
        {
            Debug.Log("Cannot find 'GameController' script");
        }

    }
    void OnTriggerEnter(Collider other)
    {
        if(other.CompareTag("Boundary") || other.CompareTag("Enemy"))
        {
            return;
        }

        if(explosion != null)
        {
            Instantiate(explosion, transform.position, transform.rotation);
        }
        if (other.CompareTag("Player"))
        {
            Instantiate(playerExplosion, other.transform.position, other.transform.rotation);
            gameController.GameOver();
        }

        gameController.AddScore(scoreValue);
        Destroy(other.gameObject);
        Destroy(gameObject);
    }
}
```

File Edit View Git Project Build Debug Test Ana

Debug Any CPU

Server Explorer Toolbox

DestroyByTime.cs DestroyByContact.cs DestroyByBound

Miscellaneous Files

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class DestroyByTime : MonoBehaviour
6   {
7       public float lifetime;
8       void Start()
9       {
10          Destroy(gameObject, lifetime);
11      }
12  }
```

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q)

Debug | Any CPU | IIS Express (Microsoft Edge)

EvasiveManeuver.cs | DestroyByTime.cs | DestroyByContact.cs | DestroyByBoundary.cs | BGScroller.cs

Miscellaneous Files | EvasiveManeuver

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

public class EvasiveManeuver : MonoBehaviour
{
    public float dodge;
    public float tilt;
    public float smoothing;
    public Vector2 startWait;
    public Vector2 maneuverTime;
    public Vector2 maneuverWait;
    private float targetManeuver;
    private float currentSpeed;

    public Boundary boundary;
    private Rigidbody rb;

    void Start()
    {
        rb = GetComponent<Rigidbody>();
        currentSpeed = rb.velocity.z;
        StartCoroutine(Evade());

    }

    IEnumerator Evade()
    {
        yield return new WaitForSeconds(Random.Range(startWait.x, startWait.y));

        while(true)
        {
            targetManeuver = Random.Range(1, dodge) * -Mathf.Sign(transform.position.x);
            yield return new WaitForSeconds(Random.Range(maneuverTime.x, maneuverTime.y));
            targetManeuver = 0;
            yield return new WaitForSeconds(Random.Range(maneuverWait.x, maneuverWait.y));
        }
    }
    void FixedUpdate()
    {
        float newManeuver = Mathf.MoveTowards(rb.velocity.x,targetManeuver,Time.deltaTime * smoothing);
        rb.velocity = new Vector3(newManeuver, 0.0f, currentSpeed);
        rb.position = new Vector3
        (
            Mathf.Clamp(rb.position.x, boundary.xMin, boundary.xMax),
            0.0f,
            Mathf.Clamp(rb.position.z, boundary.zMin, boundary.zMax)
        );
        rb.rotation = Quaternion.Euler(0.0f, 0.0f, rb.velocity.x * -tilt);
    }
```

100 % | No issues found

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q)

Debug | Any CPU | IIS Express (Microsoft Edge)

GameController.cs | EvasiveManeuver.cs | DestroyByTime.cs | DestroyByContact.cs | DestroyByBoundary.cs | BGScroller.cs

Miscellaneous Files | GameController | hazards

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

public class GameController : MonoBehaviour
{
    public GameObject[] hazards;
    public Vector3 spawnValues;
    public int harzardCount;
    public float spawnWait;
    public float startWait;
    public float waveWait;

    public Text scoreText;
    public Text restartText;
    public Text gameOverText;
    private int score;
    private bool gameOver;
    private bool restart;

    void Start()
    {
        gameOver = false;
        restart = false;
        score = 0;
        restartText.text = "";
        gameOverText.text = "";
        UpdateScore();
        StartCoroutine(SpawnWaves());
    }

    void Update()
    {
        if(restart)
        {
            if(Input.GetKeyDown(KeyCode.R))
            {
                Application.LoadLevel(Application.loadedLevel);
            }
        }
    }

    IEnumerator SpawnWaves()
    {
        yield return new WaitForSeconds(startWait);
        while (true)
        {
            for (int i = 0; i < harzardCount; i++)
            {
```

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q)

Debug | Any CPU | IIS Express (Microsoft Edge)

GameController.cs | EvasiveManeuver.cs | DestroyByTime.cs | DestroyByContact.cs | DestroyByBoundary.cs | BGScroller.cs

Miscellaneous Files | GameController | hazards

```
            }
        }
    }

    IEnumerator SpawnWaves()
    {
        yield return new WaitForSeconds(startWait);
        while (true)
        {
            for (int i = 0; i < harzardCount; i++)
            {
                GameObject hazard = hazards[Random.Range(0, hazards.Length)];
                Vector3 spawnPosition = new Vector3(Random.Range(-spawnValues.x, spawnValues.x), spawnValues.y, spawnValues.z);
                Quaternion spawnRotation = Quaternion.identity;
                Instantiate(hazard, spawnPosition, spawnRotation);
                yield return new WaitForSeconds(spawnWait);
            }
            yield return new WaitForSeconds(waveWait);

            if(gameOver)
            {
                restartText.text = "Press R to restart";
                restart = true;
                break;
            }
        }
    }

    public void AddScore(int newScoreValue)
    {
        score += newScoreValue;
        UpdateScore();
    }
    void UpdateScore()
    {
        scoreText.text = "Score:" + score;
    }

    public void GameOver()
    {
        gameOverText.text = "Game Over";
        gameOver = true;
    }
}

```

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window H

Debug | Any CPU | IIS Express (Microsoft Edge)

Mover.cs | GameController.cs | EvasiveManeuver.cs | DestroyByTime.cs | DestroyByCont

Miscellaneous Files | Mover

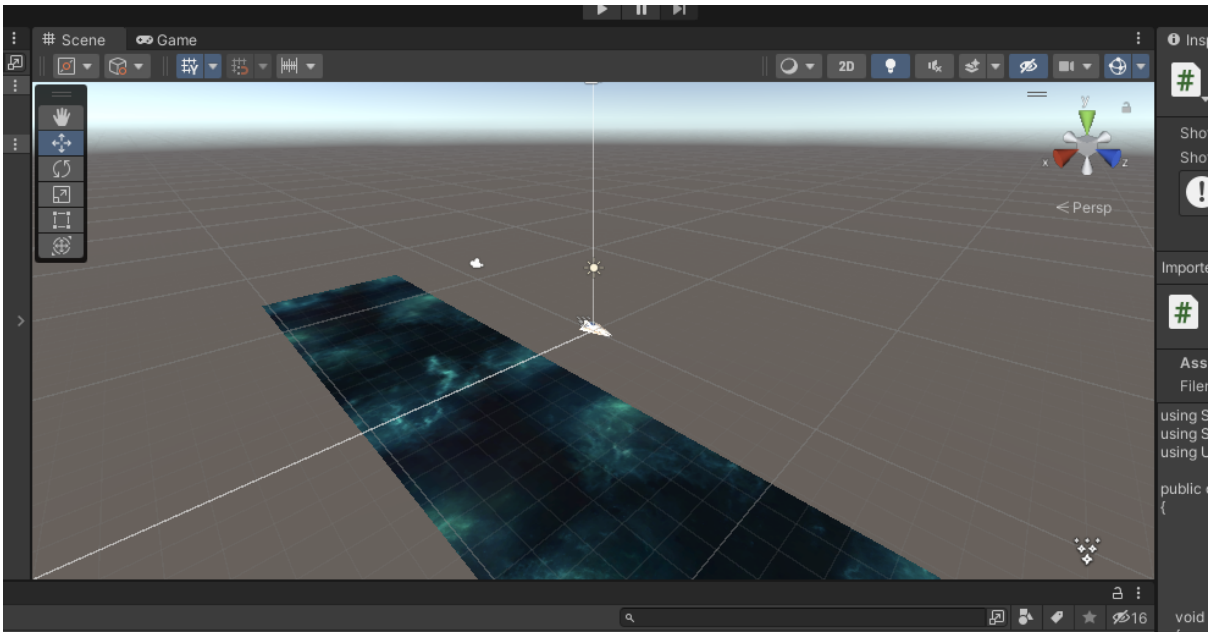
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Mover : MonoBehaviour
{
    public float speed;
    void Start()
    {
        GetComponent<Rigidbody>().velocity = transform.forward * speed;
    }
}

```

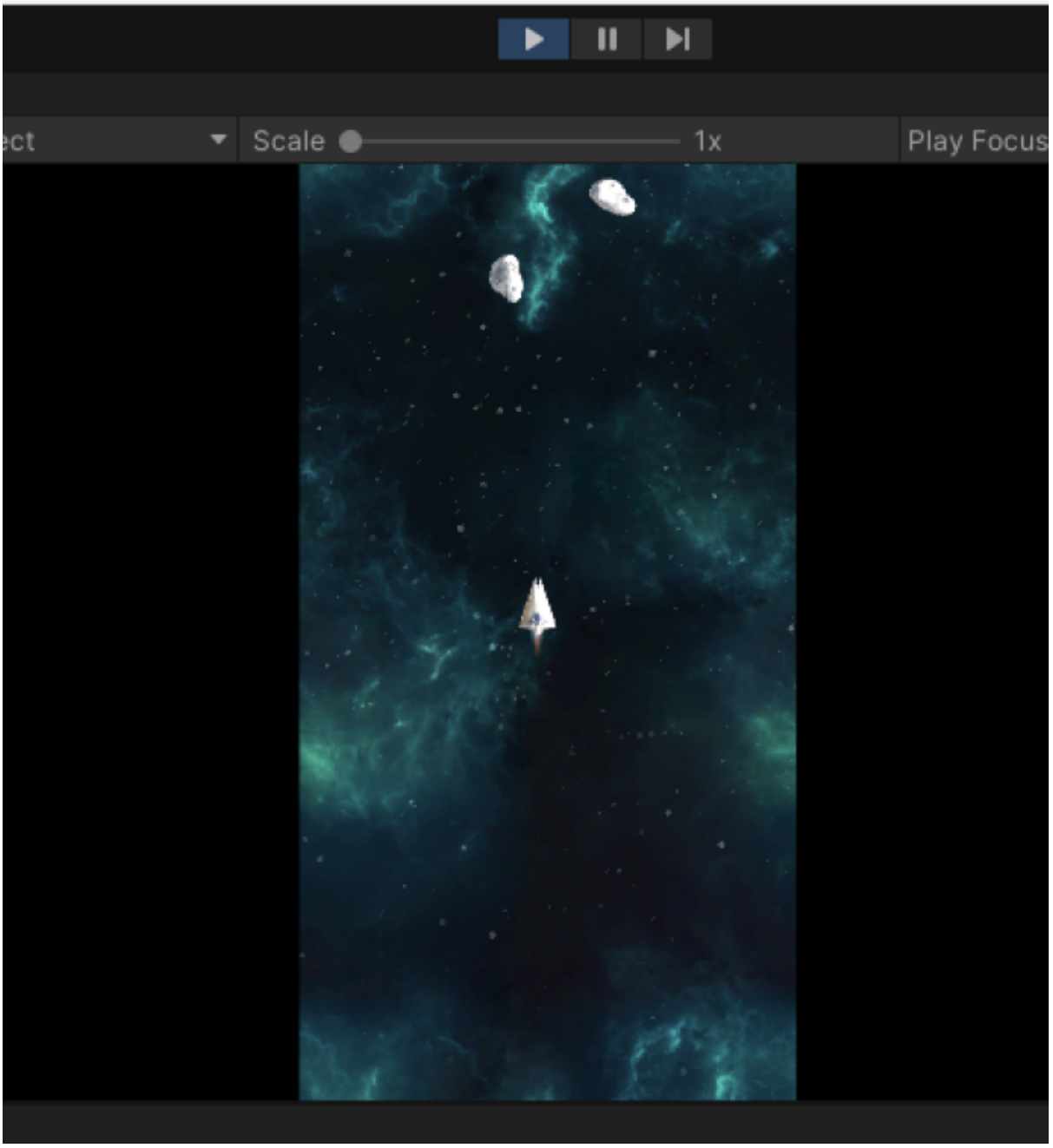
```
using System.Collections;
using UnityEngine;

[System.Serializable]
public class Boundary
{
    public float xMin, xMax, zMin, zMax;
}
public class PlayerController : MonoBehaviour
{
    public float speed;
    public float tilt;
    public Boundary boundary;

    public GameObject shot;
    public Transform shotSpawn;

    public float fireRate;
    private float nextFire;

    void Update()
    {
        if(Input.GetButton("Fire1") && Time.time > nextFire)
        {
            nextFire = Time.time + fireRate;
            Instantiate(shot, shotSpawn.position, shotSpawn.rotation);
        }
    }

    void FixedUpdate()
    {
        float moveHorizontal = Input.GetAxis("Horizontal");
        float moveVertical = Input.GetAxis("Vertical");

        Vector3 movement = new Vector3(moveHorizontal, 0.0f, moveVertical);
        GetComponent<Rigidbody>().velocity = movement * speed;

        GetComponent<Rigidbody>().position = new Vector3
        (
            Mathf.Clamp(GetComponent<Rigidbody>().position.x, boundary.xMin, boundary.xMax),
            0.0f,
            Mathf.Clamp(GetComponent<Rigidbody>().position.z, boundary.zMin, boundary.zMax)
        );
        GetComponent<Rigidbody>().rotation = Quaternion.Euler(0.0f, 0.0f, GetComponent<Rigidbody>().velocity.x * -tilt);
    }
}
```

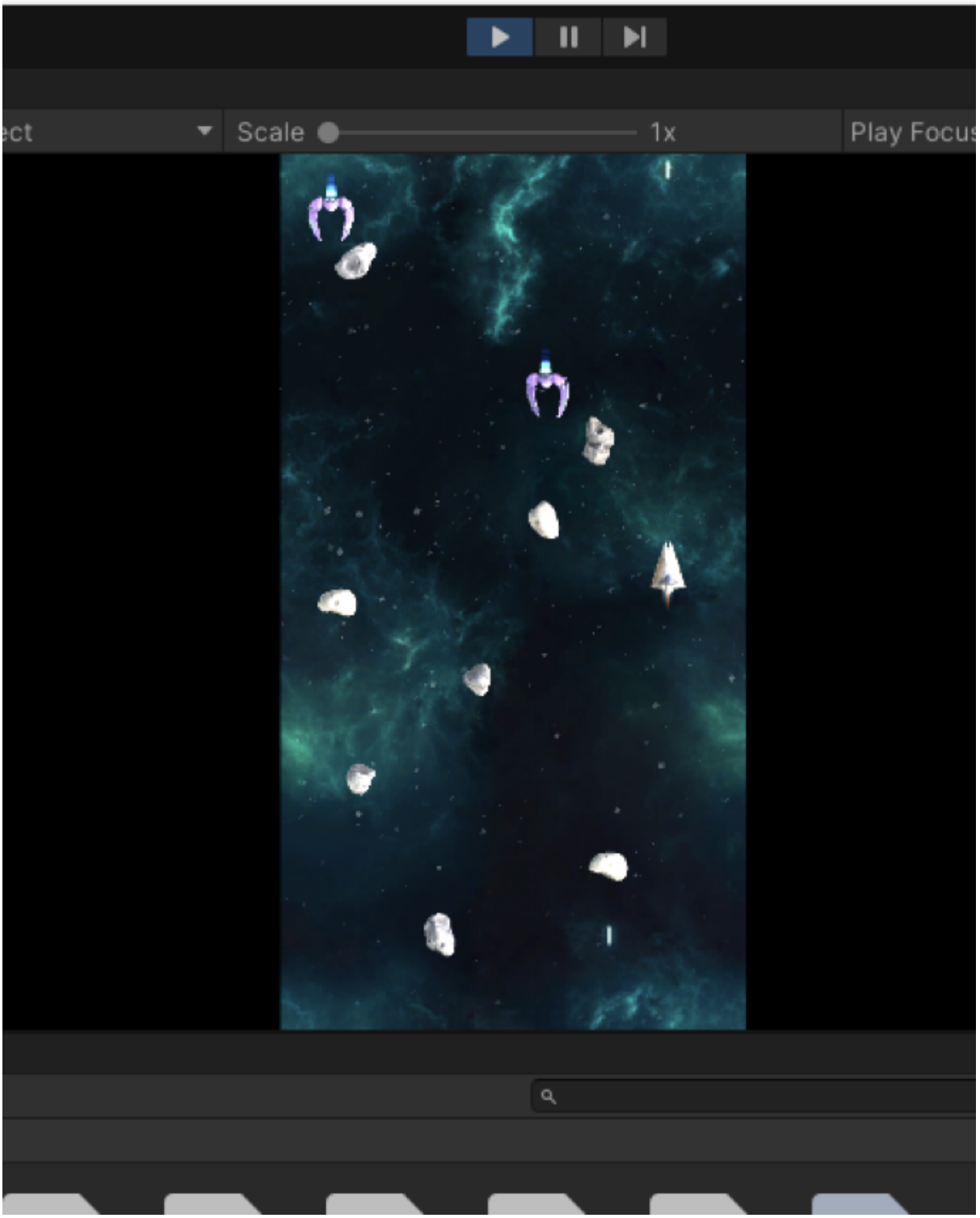
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class RandomRotator : MonoBehaviour
{
    public float tumble;

    void Start()
    {
        GetComponent<Rigidbody>().angularVelocity = Random.insideUnitSphere * tumble;
    }
}
```

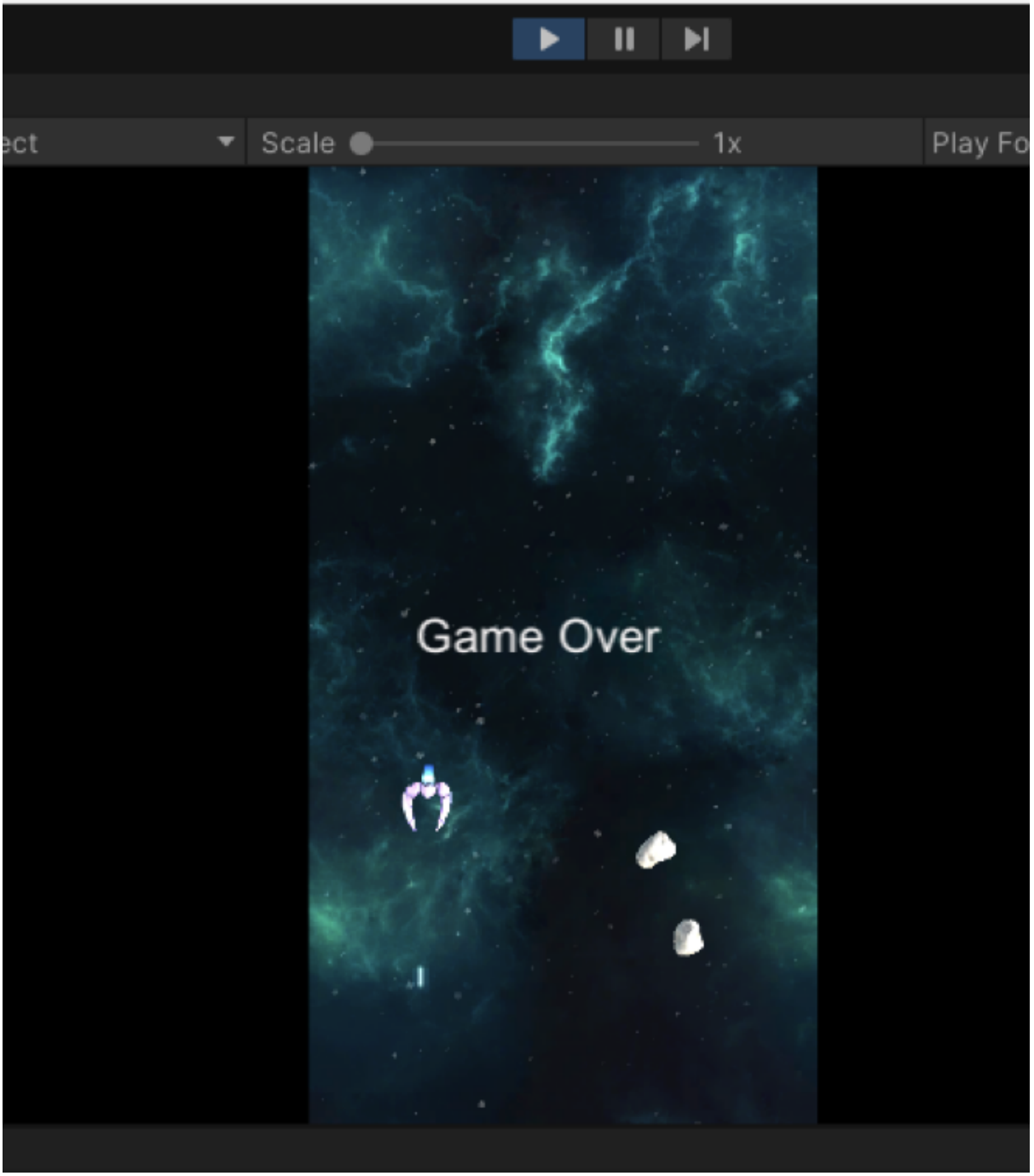
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class WeaponController : MonoBehaviour
{
    public GameObject shot;
    public Transform shotSpawn;
    public float fireRate;
    public float delay;
    private AudioSource audioSource;
    void Start()
    {
        audioSource = GetComponent<AudioSource>();
        InvokeRepeating("Fire",delay,fireRate);
    }
    void Fire()
    {
        Instantiate(shot, shotSpawn.position, shotSpawn.rotation);
        audioSource.Play();
    }
}
```

Output:


Scene
Game
2D
Persp
using S
using S
using U
public
void


Scale
1x
Play Focus


Scale
1x
Play Focus


ect
Scale
1x
Play Fo
Game Over