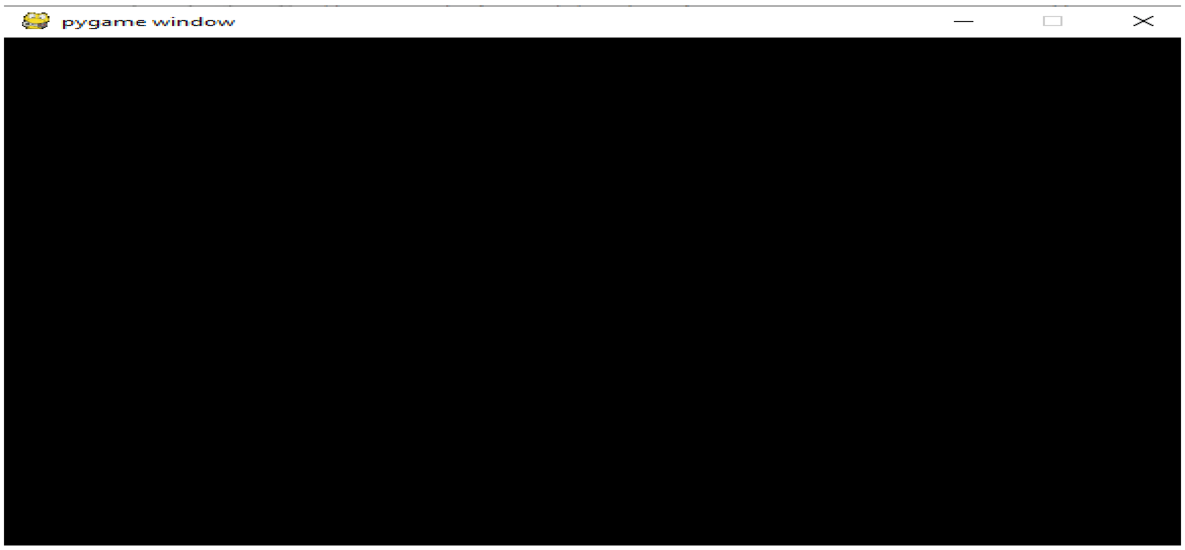
practical:-2

Aim : Setup DirectX 11, Window Framework and Initialize Direct3D Device, Loading models into DirectX 11 and rendering.

```
# display blank screen
import pygame
pygame.init()
screen = pygame.display.set_mode((800,800))
done = False
while not done:
   for event in pygame.event.get():
      if event.type == pygame.QUIT:
         done = True
   pygame.display.flip()
```

OUTPUT:



practical:-2

Aim:- Learn Basic Game Designing Techniques with pygame.

```
#add imag
import pygame
pygame.init()
white = (255, 255, 255)

# assigning values to height and width variable
height = 400
width = 400
# creating the display surface object
of specific dimension..e(X, Y).
display_surface = pygame.display.set_mode((height, width))
# set the pygame window name
pygame.display.set_caption('Image')
image = pygame.image.load(r'C:\Users\latab\OneDrive\Desktop\chess.png')

# infinite loop
while True:
   display_surface.fill(white)
   display_surface.blit(image, (0, 0))
   for event in pygame.event.get():
      if event.type == pygame.QUIT:
         pygame.quit()
         # quit the program.
         quit()
      # Draws the surface object to the screen.
      pygame.display.update()
#Key pressing and  key release
import pygame
pygame.init()
pygame.display.set_caption(u'Keyboard events')
# sets the window size
pygame.display.set_mode((800, 800))
```

```
while True:
    # gets a single event from the event queue
    event = pygame.event.wait()
    # if the 'close' button of the window is pressed
    if event.type == pygame.QUIT:
        # stops the application
        break

  # Detects the 'KEYDOWN' and 'KEYUP' events
    if event.type in (pygame.KEYDOWN, pygame.KEYUP):
        # gets the key name
        key_name = pygame.key.name(event.key)
        # converts to uppercase the key name
        key_name = key_name.upper()

     # if any key is pressed
        if event.type == pygame.KEYDOWN:
            # prints on the console the key pressed
            print(u'"{}" key pressed'.format(key_name))
  # if any key is released
        elif event.type == pygame.KEYUP:
            # prints on the console the released key
            print(u'"{}" key released'.format(key_name))
#to dispaly different shapes(rectangle, circle, triangle, elipse)
import pygame
from math import pi
pygame.init()

# size variable is using for set screen size
size = [800, 800]
screen = pygame.display.set_mode(size)
pygame.display.set_caption("Example program to draw geometry")
# done variable is using as flag
done = False
clock = pygame.time.Clock()
while not done:
    clock.tick(10)
    for event in pygame.event.get():

  if event.type == pygame.QUIT:  # If user clicked on close symbol
            done = True  # done variable that we are complete, so we exit this loop

 # All drawing code occurs after the for loop and but
    # inside the main while done==False loop.
    # Clear the default screen background and set the white screen background
    screen.fill((255, 255, 255))
# Draw on the screen a green line which is 5 pixels wide.
    pygame.draw.line(screen, (0, 255, 0), [0, 0], [50, 30], 5)

  # Draw on the screen a green line which is 5 pixels wide.
    pygame.draw.lines(screen, (0, 0, 0), False, [[0, 80], [50, 90], [200, 80], [220, 30]], 5)

    # Draw a rectangle outline
    pygame.draw.rect(screen, (0, 0, 0), [75, 10, 50, 20], 2)




# Draw a solid rectangle
    pygame.draw.rect(screen, (0, 0, 0), [150, 10, 50, 20])

    # This draw an ellipse outline, using a rectangle as the outside boundaries
    pygame.draw.ellipse(screen, (255, 0, 0), [225, 10, 50, 20], 2)

# This draw a solid ellipse, using a rectangle as the outside boundaries
    pygame.draw.ellipse(screen, (255, 0, 0), [300, 10, 50, 20])

    # Draw a triangle using the polygon function
    pygame.draw.polygon(screen, (0, 0, 0), [[100, 100], [0, 200], [200, 200]], 5)

    # This draw a circle
    pygame.draw.circle(screen, (0, 0, 255), [60, 250], 40)

    # This draw an arc
    pygame.draw.arc(screen, (0, 0, 0), [210, 75, 150, 125], 0, pi / 2, 2)

    # This function must write after all the other drawing commands.
    pygame.display.flip()

# Quite the execution when clicking on close
pygame.quit()
```

practical 3

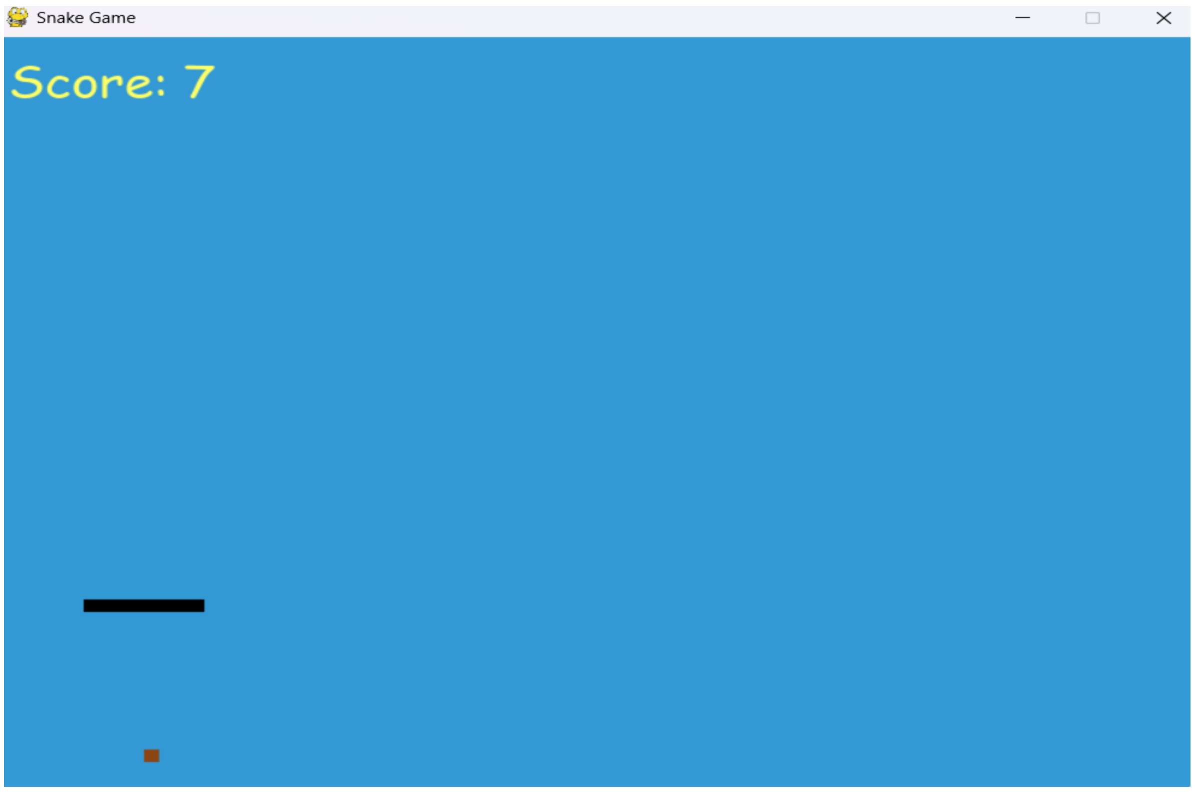
Aim:- Develop Snake Game using pygame

```
import pygame
import time
import random
pygame.init()
# Colors

white = (255, 255, 255)
yellow = (255, 255, 102)
black = (0, 0, 0)
red = (213, 50, 80)
brown = (0, 255, 0)
blue = (50, 153, 213)

# Display dimensions
dis_width = 800
dis_height = 600

# Set up display
dis = pygame.display.set_mode((dis_width, dis_height))
pygame.display.set_caption('Snake Game')

# Clock for controlling the game's frame rate
clock = pygame.time.Clock()
snake_block = 10
snake_speed = 15

# Fonts
font_style = pygame.font.SysFont("bahnschrift", 25)
score_font = pygame.font.SysFont("comicsansms", 35)
```

```
def our_snake(snake_block, snake_list):
              for x in snake_list:
        pygame.draw.rect(dis, black, [x[0], x[1], snake_block, snake_block])

def message(msg, color):
 mesg = font_style.render(msg, True, color)
    dis.blit(mesg, [dis_width / 6, dis_height / 3])

def gameLoop():
game_over = False
    game_close = False

    x1 = dis_width / 2
    y1 = dis_height / 2

    x1_change = 0
    y1_change = 0

    snake_List = []
    Length_of_snake = 1

    foodx = round(random.randrange(0, dis_width - snake_block) / 10.0) * 10.0
    foody = round(random.randrange(0, dis_height - snake_block) / 10.0) * 10.0

    score = 0
    while not game_over:
        while game_close:
            dis.fill(blue)
            message("You Lost! Press Q-Quit or C-Play Again", red)
            pygame.display.update()

            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    game_over = True
                    game_close = False
        if event.type == pygame.KEYDOWN:
          if event.key == pygame.K_q:
            game_over = True
            game_close = False
          if event.key == pygame.K_c:
            gameLoop()

    for event in pygame.event.get():
      if event.type == pygame.QUIT:
        game_over = True
   if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_LEFT and x1_change == 0:
          x1_change =-snake_block
        y1_change = 0
        elif event.key == pygame.K_RIGHT and x1_change == 0:
          x1_change = snake_block
          y1_change = 0
        elif event.key == pygame.K_UP and y1_change == 0:
          y1_change =-snake_block
          x1_change = 0
        elif event.key == pygame.K_DOWN and y1_change == 0:
          y1_change = snake_block
          x1_change = 0

    if x1 >= dis_width or x1 < 0 or y1 >= dis_height or y1 < 0:
      game_close = True

    x1 += x1_change
    y1 += y1_change
    dis.fill(blue)
    pygame.draw.rect(dis, (139, 69, 19), [foodx, foody, snake_block, snake_block])  # Changed color to
brown
    snake_Head = []
    snake_Head.append(x1)
    snake_Head.append(y1)
    snake_List.append(snake_Head)
```

```
        if len(snake_List) > Length_of_snake:
            del snake_List[0]

        for segment in snake_List[:-1]:
            if segment == snake_Head:
                game_close = True
              our_snake(snake_block, snake_List)
        # Display Score
        score_text = score_font.render("Score: " + str(score), True, yellow)
        dis.blit(score_text, [10, 10])

        pygame.display.update()

        if x1 == foodx and y1 == foody:

              foodx = round(random.randrange(0, dis_width - snake_block) / 10.0) * 10.0
               foody = round(random.randrange(0, dis_height - snake_block) / 10.0) * 10.0
            Length_of_snake += 1
            score += 1
        clock.tick(snake_speed)

    pygame.quit()
    quit()

gameLoop()
```

OUTPUT:-



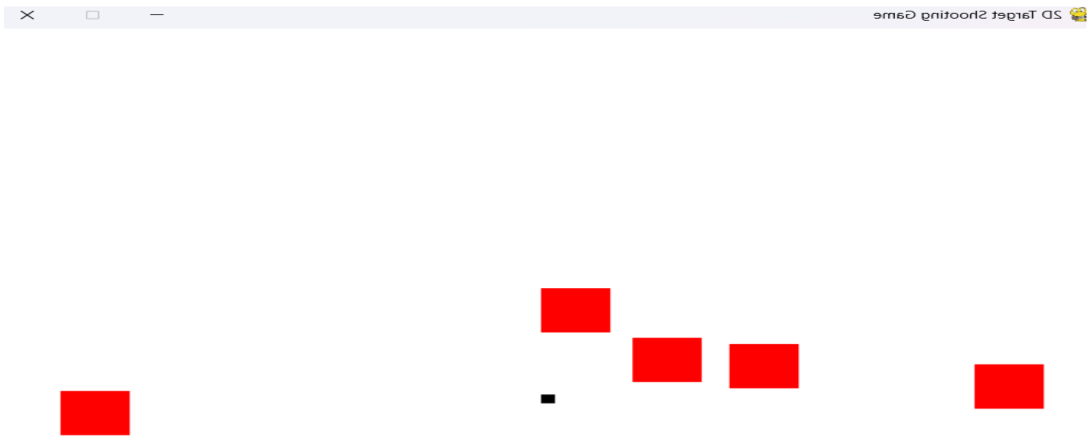
practical:4

Aim:- Create 2D Target Shooting Game

```
import pygame
import random

# Screen
pygame.init()
width, height = 800, 600
screen = pygame.display.set_mode((width, height))
pygame.display.set_caption("Shooting Game")

# Colors
white = (255, 255, 255)
red = (255, 0, 0)
blue = (0, 0, 255)

# Character
character_size = 50
character_speed = 5
character = pygame.Rect(width // 2 - character_size // 2, height - character_size, character_size,
character_size)

# Bullets (triangles) properties
bullet_size = 10
bullets = []

# Enemy (circle) properties
enemy_radius = 20
enemies = []

# Initialize score
score = 0

# Clock for controlling frame rate
```

```
clock = pygame.time.Clock()

# Main game loop
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_SPACE:
                bullet = pygame.Rect(character.centerx - bullet_size // 2, character.top, bullet_size,
bullet_size)
                bullets.append(bullet)

    # Move bullets
    for bullet in bullets[:]:
        bullet.y -= 10
        if bullet.top < 0:
            bullets.remove(bullet)

    # Spawn enemies
    if random.randint(1, 100) <= 2:
        enemy_x = random.randint(enemy_radius, width - enemy_radius)
        enemy = pygame.Rect(enemy_x - enemy_radius, 0, enemy_radius * 2, enemy_radius * 2)
        enemies.append(enemy)

    # Move enemies
    for enemy in enemies[:]:
        enemy.y += 5
        if enemy.top > height:
            enemies.remove(enemy)

    # Move character
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT]:
        character.x -= character_speed
    if keys[pygame.K_RIGHT]:
        character.x += character_speed

    # Check for collisions
    for bullet in bullets[:]:
        for enemy in enemies[:]

if bullet.colliderect(enemy):
                score += 1
                bullets.remove(bullet)
                enemies.remove(enemy)
                break  # Exit loop once collision is detected to avoid removing the same bullet multiple times

    for enemy in enemies:
        if character.colliderect(enemy):
            running = False

    # Clear the screen
    screen.fill(white)

    # Draw bullets
    for bullet in bullets:
        pygame.draw.polygon(screen, blue, [(bullet.left, bullet.bottom), (bullet.centerx, bullet.top),
(bullet.right, bullet.bottom)])

    # Draw character
    pygame.draw.rect(screen, red, character)

    # Draw enemies
    for enemy in enemies:
        pygame.draw.circle(screen, red, enemy.center, enemy_radius)

    # Display score
    font = pygame.font.Font(None, 36)
    score_text = font.render(f"Score: {score}", True, red)
    screen.blit(score_text, (10, 10))
    # Update display
    pygame.display.flip()

    # Limit frame rate to 60 FPS
    clock.tick(60)

# Game over display
screen.fill(white)
font = pygame.font.Font(None, 72)
game_over_text = font.render("Game Over", True, red)
screen.blit(game_over_text, (width // 2 - game_over_text.get_width() // 2, height // 2 -
game_over_text.get_height() // 2))
pygame.display.flip()

# Wait for a few seconds before closing the game
pygame.time.wait(3000)

# Clean up
pygame.quit()
```

OUTPUT:-



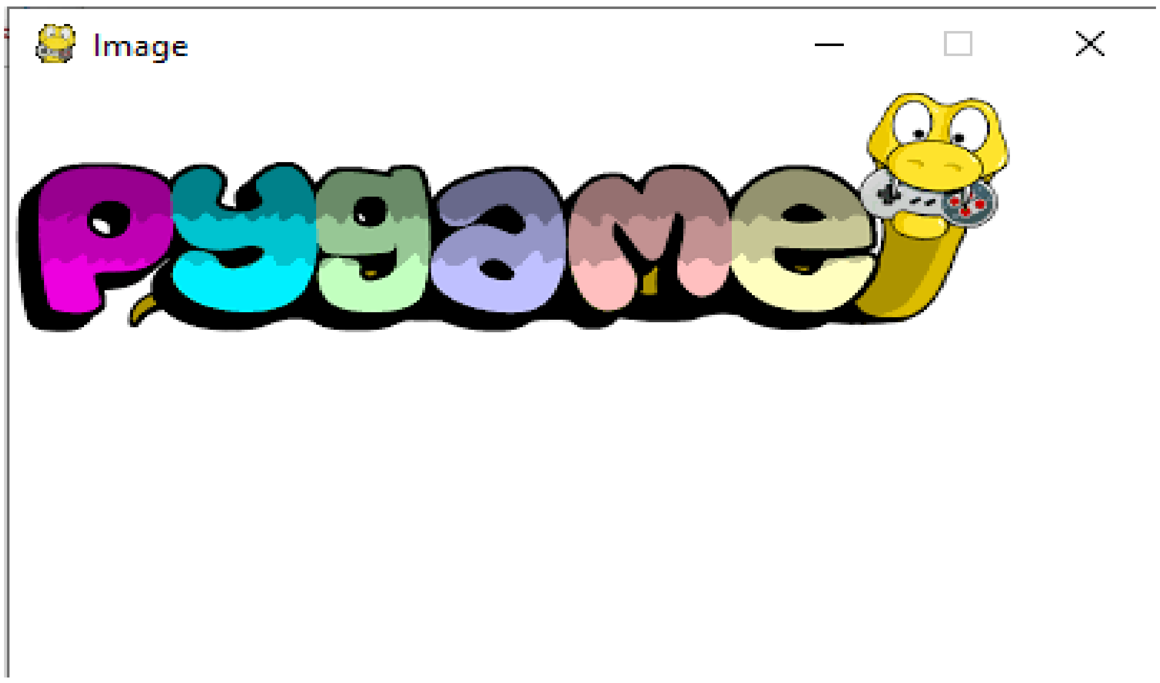
practical 5

Aim:- Creating 2D Infinite Scrolling Background

```
#scrolling background
import math
import pygame as py

py.init()

clock = py.time.Clock()

FrameHeight = 600
FrameWidth = 1200

# PYGAME FRAME WINDOW
py.display.set_caption("Endless Scrolling in pygame")
screen = py.display.set_mode((FrameWidth, FrameHeight))

# IMAGE
bg = py.image.load(r'C:\Users\latab\OneDrive\Desktop\background.png').convert()

# DEFINING MAIN VARIABLES IN SCROLLING
scroll = 0

# CHANGE THE BELOW 1 TO UPPER NUMBER IF
# YOU GET BUFFERING OF THE IMAGE
# HERE 1 IS THE CONSTATNT FOR REMOVING BUFFERING
tiles = math.ceil(FrameWidth / bg.get_width()) + 1

# MAIN LOOP
while 1:
	# THIS WILL MANAGE THE SPEED OF
	# THE SCROLLING IN PYGAME
	clock.tick(33)
	# APPENDING THE IMAGE TO THE BACK
```

```
        # OF THE SAME IMAGE
        i = 0
        while(i < tiles):
                screen.blit(bg, (bg.get_width()*i+ scroll, 0))

        i += 1
        # FRAME FOR SCROLLING
        scroll -= 10

        # RESET THE SCROLL FRAME
        if abs(scroll) > bg.get_width():
                scroll = 0
        # CLOSINF THE FRAME OF SCROLLING
        for event in py.event.get():
                if event.type == py.QUIT:
                        quit()

        py.display.update()

py.quit()
```

OUTPUT:-



<u>**Practical No.6**</u>

Step 1: Set Up Your Unity Project

1. **Open Unity: Start a new or existing Unity project.**
2. **Create a Scene: If you're in a new project, create a new scene.**

Step 2: Create the Shake Script

1. **Create the Shake Script:**
   - **In the Assets folder, right-click and select Create > C# Script.**
   - **Name the script Shake.**
2. **Open the Shake Script: Double-click the Shake script to open it in your code editor (e.g., Visual Studio).**
3. **Write the Shake Script: Replace the contents of the Shake script with the following code:**

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

public class Shake : MonoBehaviour
{
   public float shakeDuration = 0.2f;
```

```
    public float shakeIntensity = 0.1f;

    private Vector3 initialPosition;
    private float currentShakeDuration = 0f;

    void Start()
    {
        initialPosition = transform.localPosition; // Store initial
position
    }

    void Update()
    {
        if (currentShakeDuration > 0)
        {
            Vector3 randomOffset = Random.insideUnitSphere *
shakeIntensity; // Generate random offset
            transform.localPosition = initialPosition +
randomOffset; // Apply shake

            currentShakeDuration -= Time.deltaTime; // Reduce
shake duration
```

```
        }
        else
        {
            transform.localPosition = initialPosition; // Reset
position
        }
    }

    public void StartShake()
    {
        Debug.Log("Shake Started!"); // Log for debugging
        currentShakeDuration = shakeDuration; // Set shake
duration
    }
}
```

Step 3: Create the Shoot Script

1. **Create the Shoot Script:**
    - **In the Assets folder, right-click and select Create > C# Script.**
    - **Name the script Shoot.**
2. **Open the Shoot Script: Double-click the Shoot script to open it in your code editor.**

3. Write the Shoot Script: Replace the contents of the Shoot script with the following code:

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

public class Shoot : MonoBehaviour
{
   public Shake s;  // Reference to the Shake script

   void Update()
   {
      if (Input.GetKeyDown(KeyCode.Mouse0)) // Check for
left mouse click
      {
         Debug.Log("Mouse Click Detected!"); // Log for
debugging
         s.StartShake(); // Call shake method
      }
   }
}
```

Step 4: Set Up the Unity Scene

1. **Create an Empty GameObject:**
   - **In the Unity Editor, right-click in the Hierarchy panel and select Create Empty.**
   - **Name the GameObject ShakeController.**
2. **Attach the Shake Script:**
   - **Select the ShakeController GameObject in the Hierarchy.**
   - **In the Inspector panel, click Add Component.**
   - **Search for Shake and select it to attach the script.**
3. **Attach the Shoot Script:**
   - **With the ShakeController still selected, click Add Component again.**
   - **Search for Shoot and select it to attach the script.**
4. **Link the Shake Script:**
   - **In the Inspector, find the Shoot component.**
   - **Drag the Shake component from the ShakeController GameObject into the s field of the Shoot component.**

Step 5: Test the Shake Effect

1. **Play the Scene:**
   - **Click the Play button at the top of the Unity Editor.**
2. **Trigger the Shake:**

- **While the game is running, click the left mouse button (Mouse0). You should see the GameObject with the Shake script shaking.**

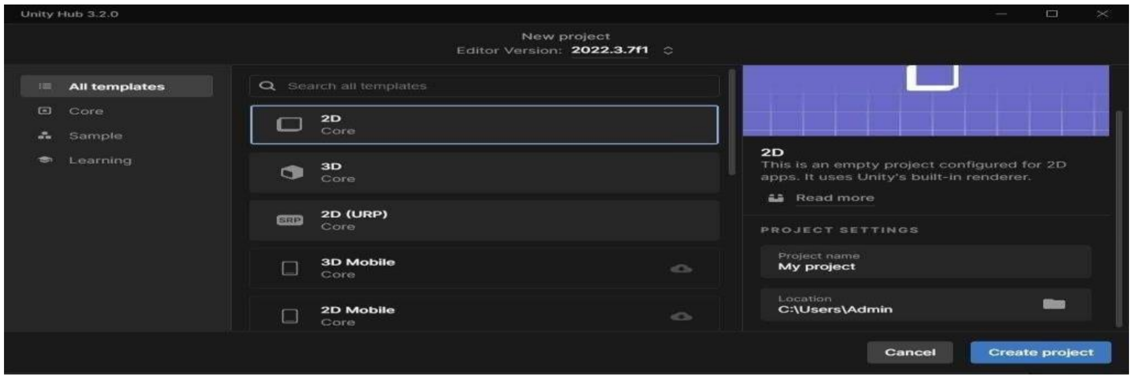
PRACTICAL 7

Aim : Design and Animate Game Character in Unity

Following are the steps to design and animate game character in Unity:

STEP 1 : Start Unity

STEP 2 : Create new project 2D. Add project name



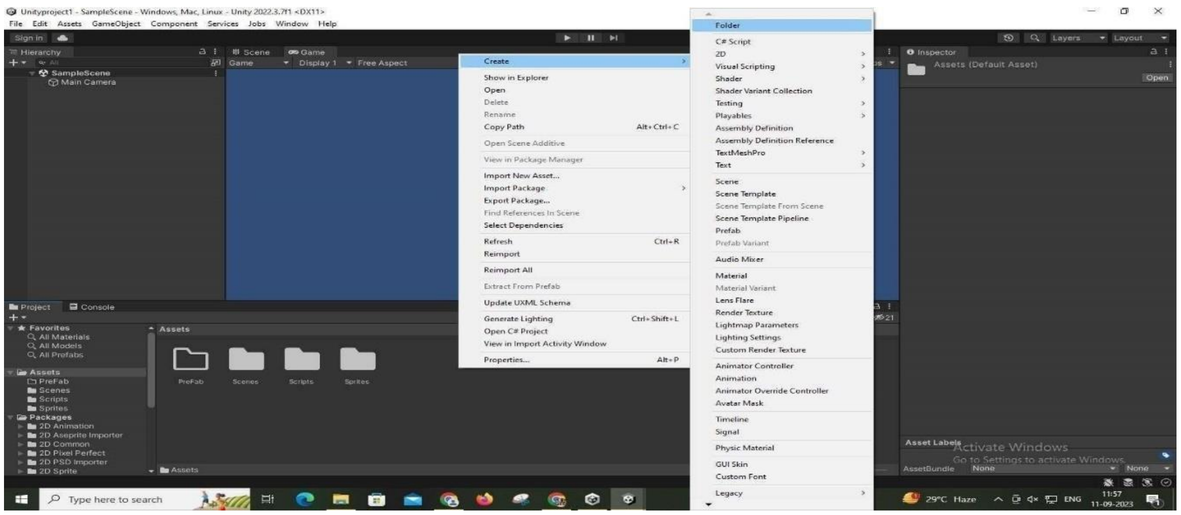
Editor Version: 2022.3.7f1

2D
This is an empty project configured for 2D apps. It uses Unity's built-in renderer.

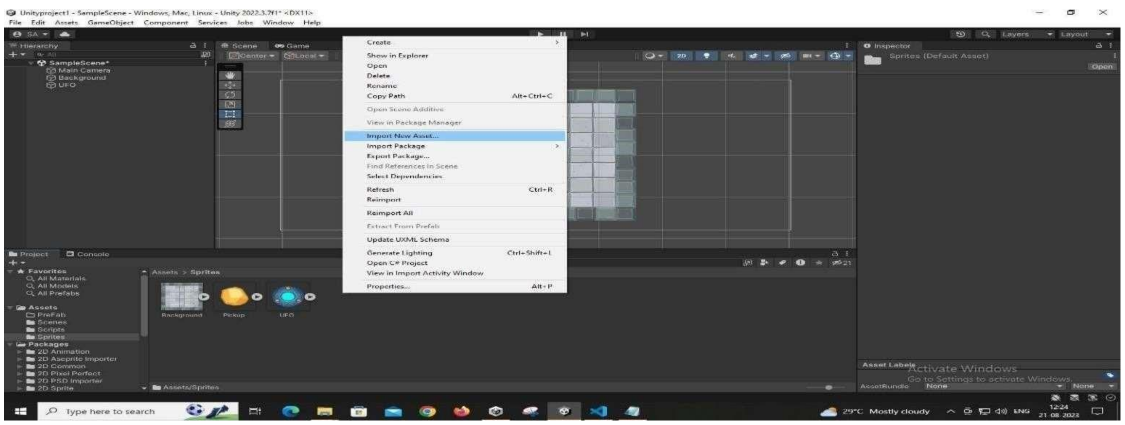
Project name
My project

Location
C:\Users\Admin

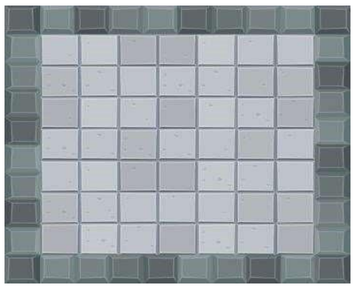
STEP 3 : Add folders such as Sprites, Scenes, Scripts, etc



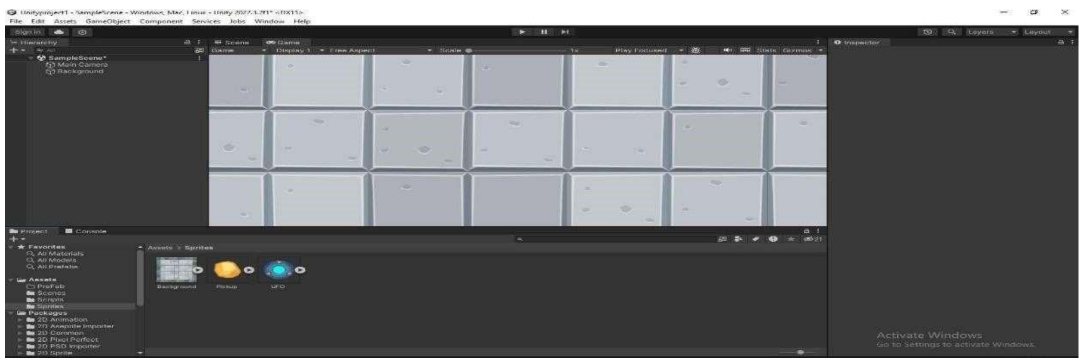
STEP 4 : Add Assets by right-click > Import Asset



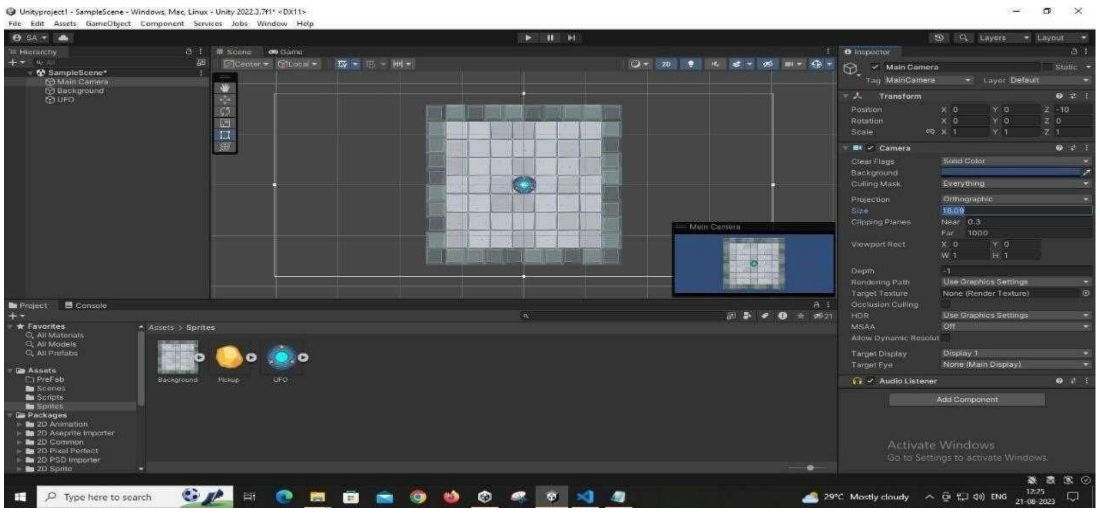
STEP 5 : Import the Following assets : Background and UFO



STEP 6 : Drag the assets to the Hierarchy > Click and hold asset and drag under MainCamera



Similarly, Drag and add the UFO under the background STEP 7 : Adjust the MainCamera by changing the size



Main Camera
Tag MainCamera Layer Default

Transform

Position	X 0	Y 0	Z -10
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

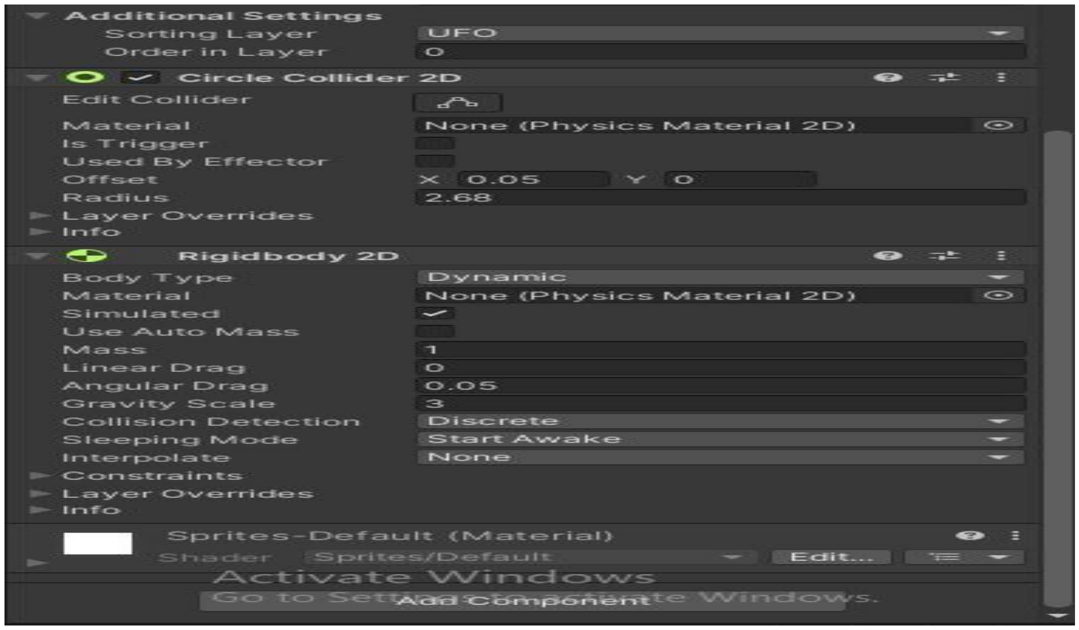
Camera

Clear Flags	Solid Color
Background	
Culling Mask	Everything
Projection	Orthographic
Size	16.09
Clipping Planes	Near 0.3
	Far 1000
Viewport Rect	X 0 Y 0
	W 1 H 1
Depth	-1
Rendering Path	Use Graphics Settings
Target Texture	None (Render Texture)
Occlusion Culling	
HDR	Use Graphics Settings
MSAA	Off
Allow Dynamic Resolut	
Target Display	Display 1
Target Eye	None (Main Display)

Audio Listener

STEP 8 : Select the UFO from Hierarchy. To make the UFO layer appear above the Background, Add a sorting layer, and select the layer. Updatethe following settings for

UFO. Add a circle collider and adjust the radius.Adjust gravity Scale



Additional Settings
Sorting Layer: UFO
Order in Layer: 0

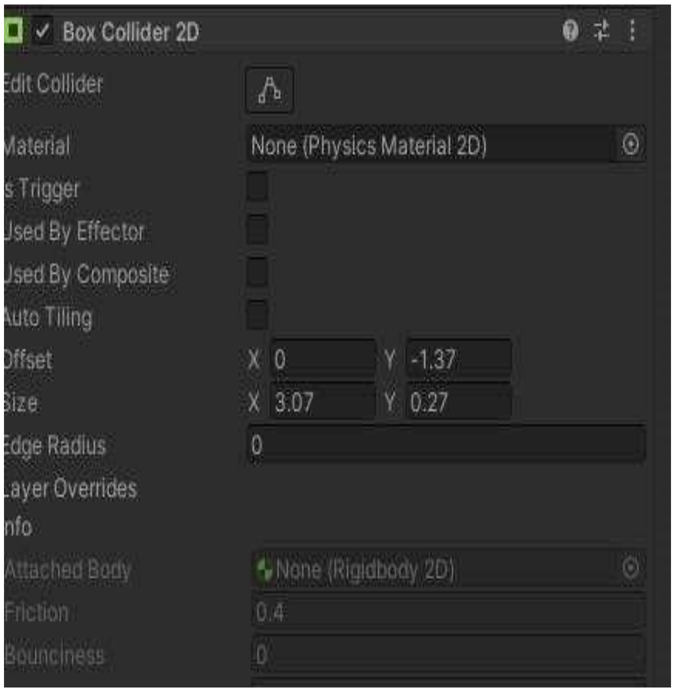
Circle Collider 2D
Edit Collider
Material: None (Physics Material 2D)
Is Trigger
Used By Effector
Offset: X 0.05 Y 0
Radius: 2.68
Layer Overrides
Info

Rigidbody 2D
Body Type: Dynamic
Material: None (Physics Material 2D)
Simulated: ✓
Use Auto Mass
Mass: 1
Linear Drag: 0
Angular Drag: 0.05
Gravity Scale: 3
Collision Detection: Discrete
Sleeping Mode: Start Awake
Interpolate: None
Constraints
Layer Overrides
Info

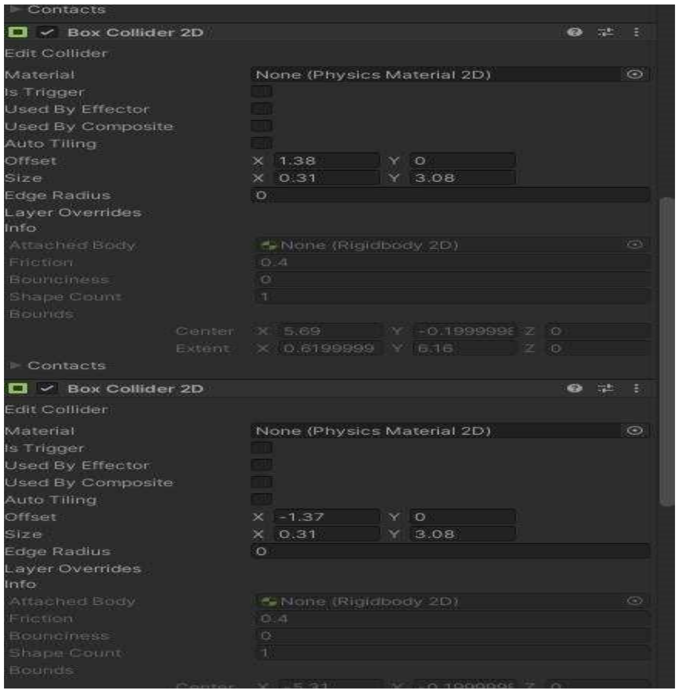
Sprites-Default (Material)
Shader: Sprites/Default

Rigidbodies are components that allow a GameObject to react to real-time physics. This includes reactions to forces and gravity, mass, drag and momentum. You can attach a
Rigidbody to your GameObject by simply clicking on Add Component and typing in Rigidbody2D in the search field

STEP 9 : Select the Background from Hierarchy. Add a box collider from Add component > Box Collider 2D. Do this step 4 times for each border of the box collider.
Also, set the Shake Frequency which set the intensity of the Shake Effect
For top box and down collider(Box collider no1 and 4) For Left and right box collider(2 and 3)

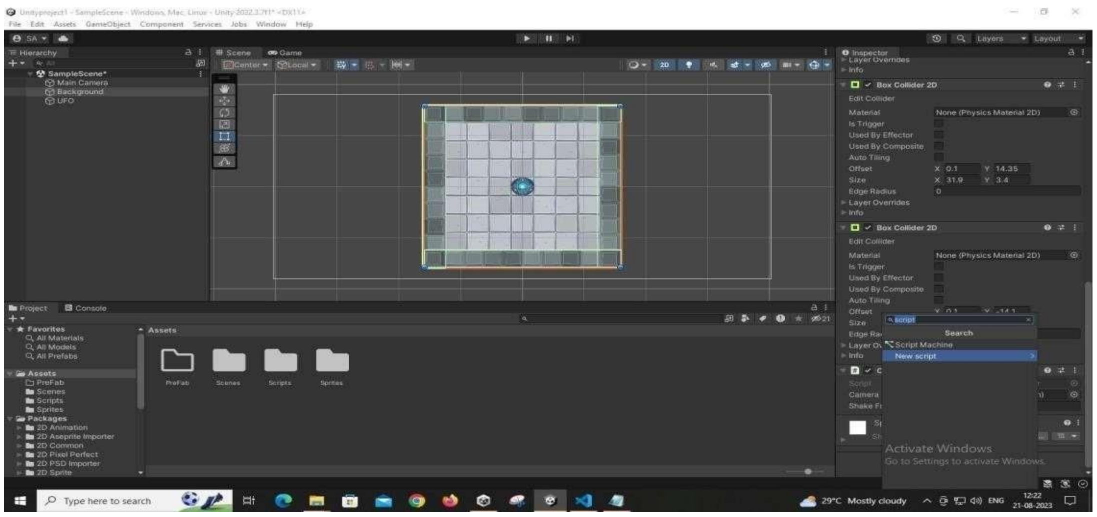


Box Collider 2D
Edit Collider
Material None (Physics Material 2D)
Is Trigger
Used By Effector
Used By Composite
Auto Tiling
Offset X 0 Y -1.37
Size X 3.07 Y 0.27
Edge Radius 0
Layer Overrides
Info
Attached Body None (Rigidbody 2D)
Friction 0.4
Bounciness 0



Contacts
Box Collider 2D
Edit Collider
Material None (Physics Material 2D)
Is Trigger
Used By Effector
Used By Composite
Auto Tiling
Offset X 1.38 Y 0
Size X 0.31 Y 3.08
Edge Radius 0
Layer Overrides
Info
Attached Body None (Rigidbody 2D)
Friction 0.4
Bounciness 0
Shape Count 1
Bounds
Center X 5.69 Y -0.1999998 Z 0
Extent X 0.6199999 Y 6.16 Z 0
Contacts
Box Collider 2D
Edit Collider
Material None (Physics Material 2D)
Is Trigger
Used By Effector
Used By Composite
Auto Tiling
Offset X -1.37 Y 0
Size X 0.31 Y 3.08
Edge Radius 0
Layer Overrides
Info
Attached Body None (Rigidbody 2D)
Friction 0.4
Bounciness 0
Shape Count 1
Bounds

STEP 10: To implement the Animate method, add a new script 'Animate' from UFO > Add Component > New Script



Material None (Physics Material 2D)
Offset X 0.1 Y 14.35
Size X 31.9 Y 3.4
Edge Radius 0

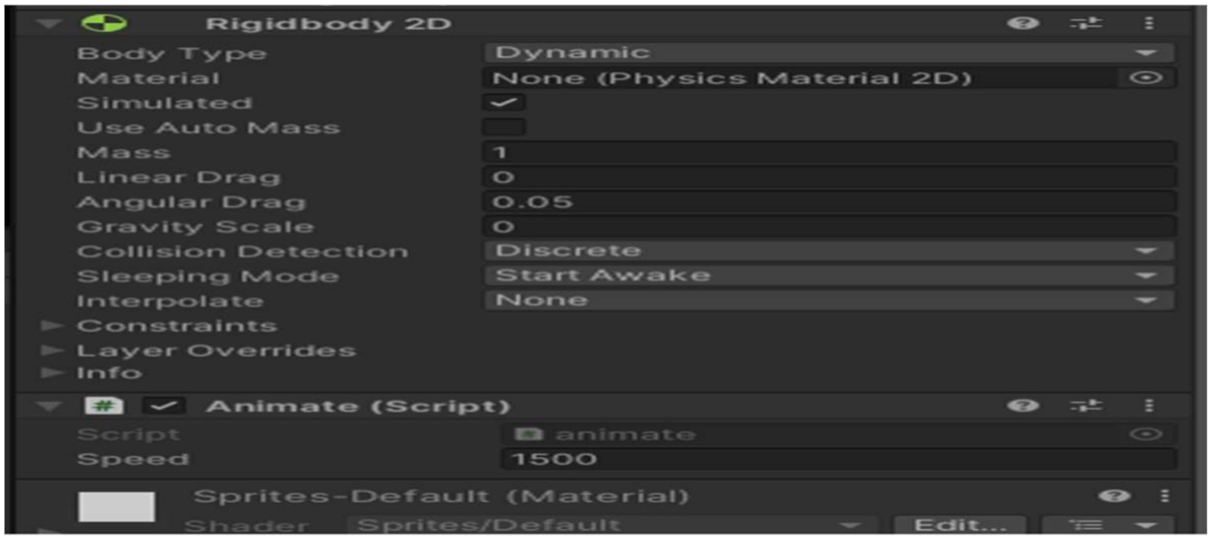
On clicking on the new script, VS code Editor will open. Add the following code
:

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

public class Animate : MonoBehaviour
{     public float
speed;
    Rigidbody2D rb2d;
    // Start is called before the first frame update     void Start()
    {
        rb2d = GetComponent<Rigidbody2D>();
    }

    // Update is called once per frame     void
Update()
```

```
    {         float moveHorizontal = Input.GetAxis("Horizontal");
  float moveVertical = Input.GetAxis("Vertical");         var movement
  = new Vector2(moveHorizontal,         moveVertical).normalized *
  speed * Time.deltaTime;         rb2d.AddForce(movement);

    }
  }
```

After typing the code in Visual Studio go to file and click on save all Then go to Unity and Adjust the following :

STEP 11 : Select UFO from Hierarchy, Set animate Speed = 1500 and add component Rigidbody2D. Set Gravity Scale = 0.



Rigidbody 2D

Body Type	Dynamic
Material	None (Physics Material 2D)
Simulated	✓
Use Auto Mass	
Mass	1
Linear Drag	0
Angular Drag	0.05
Gravity Scale	0
Collision Detection	Discrete
Sleeping Mode	Start Awake
Interpolate	None

Constraints
Layer Overrides
Info

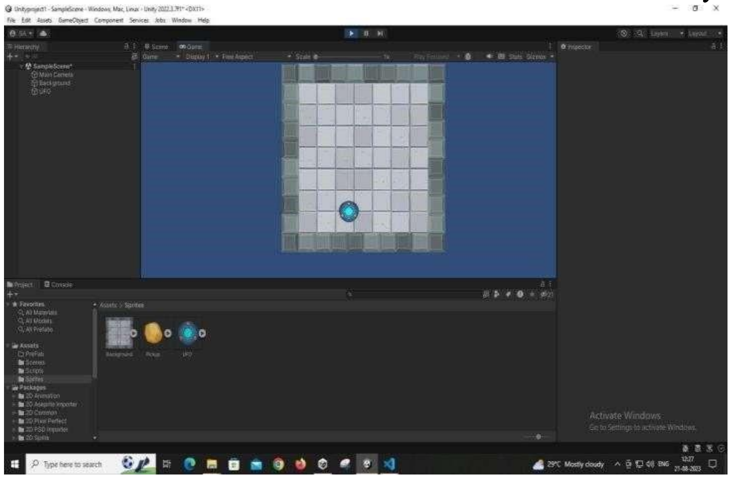
Animate (Script)

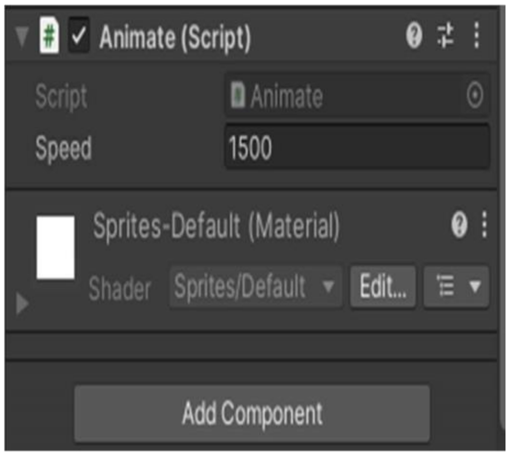
Script	animate
Speed	1500

Sprites-Default (Material)
Shader Sprites/Default Edit...

After this we will Click on File>Save and then click on Play button

Run the code and the UFO should move with the Arrow Keys





Animate (Script)

Script: Animate

Speed: 1500

Sprites-Default (Material)

Shader: Sprites/Default

Theory:

The Animate script controls the movement of a game character using Unity's Rigidbody2D component, which applies physics-based forces to move the character. This script reads input from the user (via the arrow keys or WASD keys) and moves the character accordingly.

Components of the Script:

1. Variables and Components:

   - public float speed: This public variable determines how fast the character moves. You can set this value in the Unity Inspector.
   - Rigidbody2D rb2d: This private variable holds a reference to the Rigidbody2D component attached to the game object. Rigidbody2D is used for applying physics-based movement.

2. Initialization (Start method):

    - The Start() method is called before the first frame update.
    - GetComponent<Rigidbody2D>() is used to get and store a reference to the Rigidbody2D component. This is necessary for applying forces to move the character.

3. Handling Input and Movement (Update method):

    - The Update() method is called once per frame.
    - Input.GetAxis("Horizontal") and Input.GetAxis("Vertical") get the horizontal and vertical input values, which range from -1 to 1. These are typically controlled by the arrow keys or WASD keys.
    - new Vector2(moveHorizontal, moveVertical) creates a movement vector based on user input.
    - .normalized ensures that the movement vector has a magnitude of 1 if there's any input, preventing faster diagonal movement.
    - speed * Time.deltaTime scales the movement based on the frame rate. Time.deltaTime ensures consistent movement speed regardless of the frame rate.
    - rb2d.AddForce(movement) applies the calculated force to the Rigidbody2D component, moving the character.

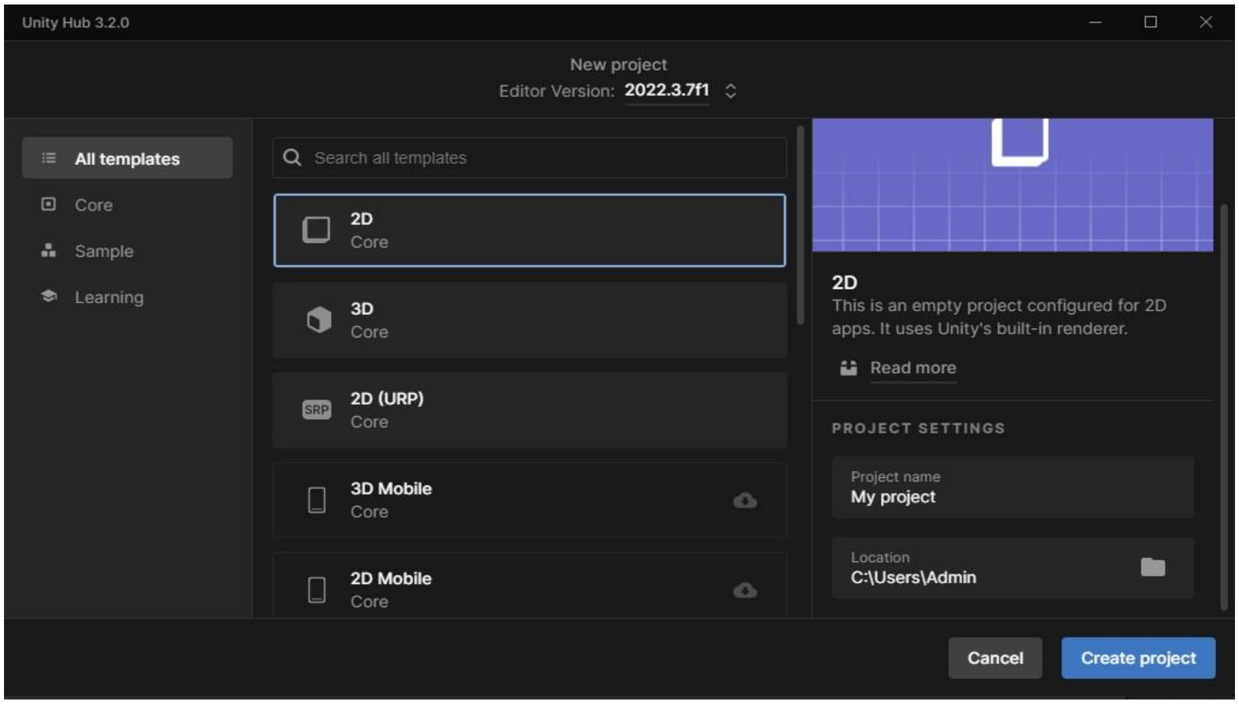
PRACTICAL NO. 08

AIM : Create Snowfall Particle Effect in Unity
DATE : 25-09-23

Following are the steps to Create Snowfall Particle Effect in Unity:

STEP 1 : Start Unity

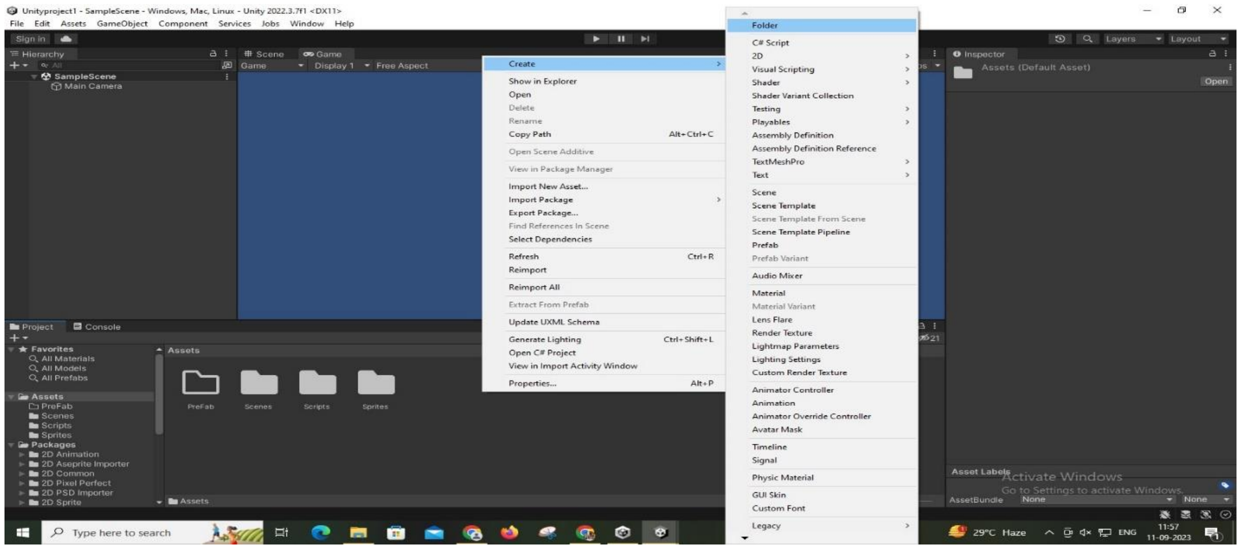
STEP 2 : Create new project 2D. Add project name



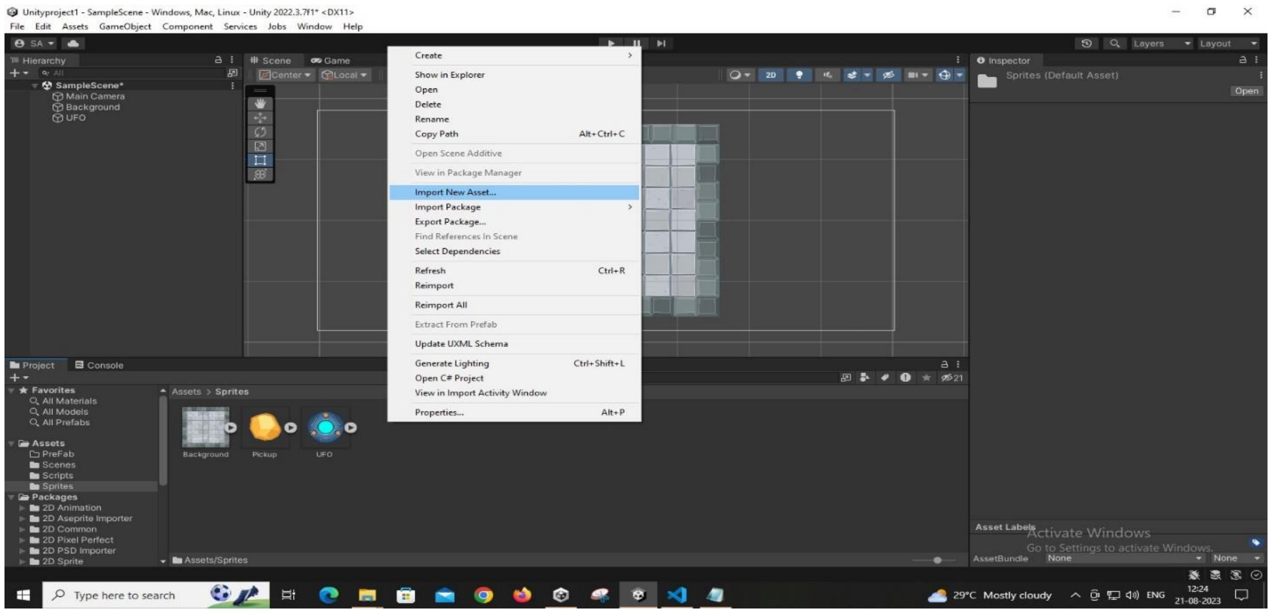
2D
This is an empty project configured for 2D apps. It uses Unity's built-in renderer.

Project name: My project
Location: C:\Users\Admin

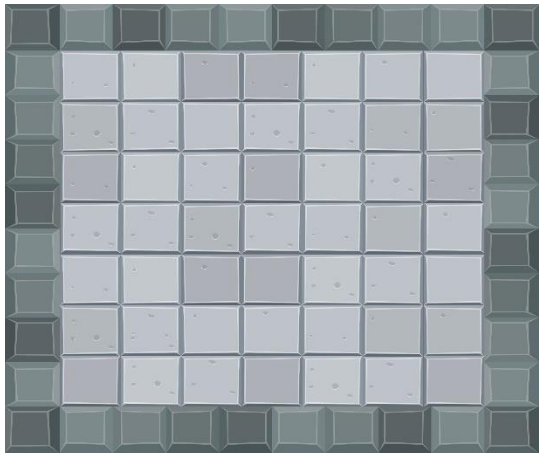
3 Add folders such as Sprites, Scenes, Scripts, etc


Unityproject1 - SampleScene - Windows, Mac, Linux - Unity 2022.3.7f1 <DX11>
File Edit Assets GameObject Component Services Jobs Window Help
Hierarchy
SampleScene
Main Camera
Scene Game
Display 1 Free Aspect
Create
Show in Explorer
Open
Delete
Rename
Copy Path Alt+Ctrl+C
Open Scene Additive
View in Package Manager
Import New Asset...
Import Package
Export Package...
Find References In Scene
Select Dependencies
Refresh Ctrl+R
Reimport
Reimport All
Extract From Prefab
Update UXML Schema
Generate Lighting Ctrl+Shift+L
Open C# Project
View in Import Activity Window
Properties... Alt+P
Folder
C# Script
2D
Visual Scripting
Shader
Shader Variant Collection
Testing
Playables
Assembly Definition
Assembly Definition Reference
TextMeshPro
Text
Scene
Scene Template
Scene Template From Scene
Scene Template Pipeline
Prefab
Prefab Variant
Audio Mixer
Material
Material Variant
Lens Flare
Render Texture
Lightmap Parameters
Lighting Settings
Custom Render Texture
Animator Controller
Animation
Animator Override Controller
Avatar Mask
Timeline
Signal
Physic Material
GUI Skin
Custom Font
Legacy
Inspector
Assets (Default Asset)
Open
Project Console
Favorites
All Materials
All Models
All Prefabs
Assets
PreFab
Scenes
Scripts
Sprites
Packages
2D Animation
2D Aseprite Importer
2D Common
2D Pixel Perfect
2D PSD Importer
2D Sprite
Asset Labels
AssetBundle None None
Activate Windows
Go to Settings to activate Windows.
Type here to search
29°C Haze
ENG
11:57
11-09-2023

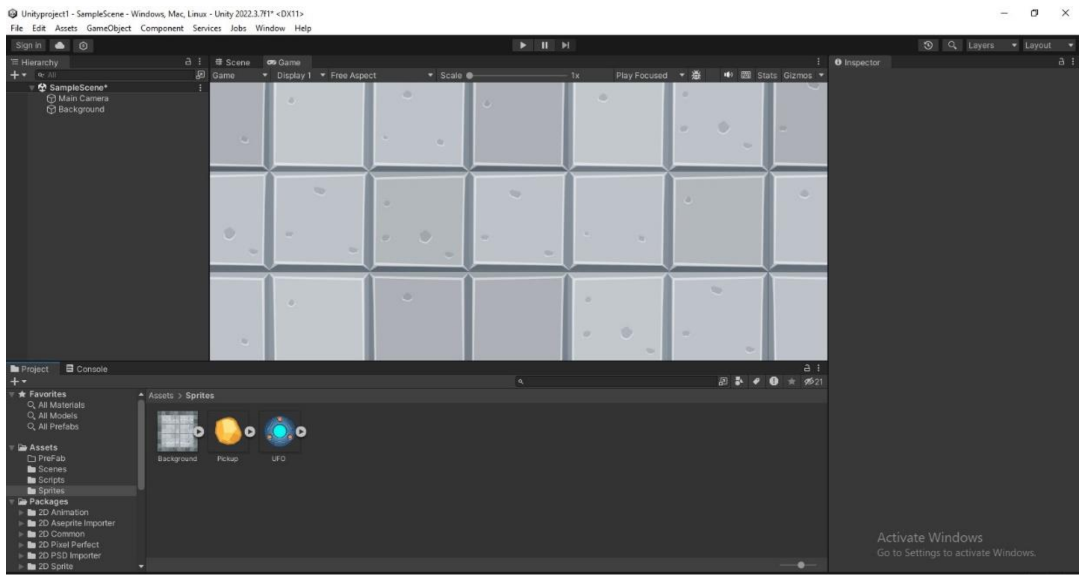
STEP 4 : Add Assets by right-click > Import Asset



5 Import the Following assets : Background and UFO

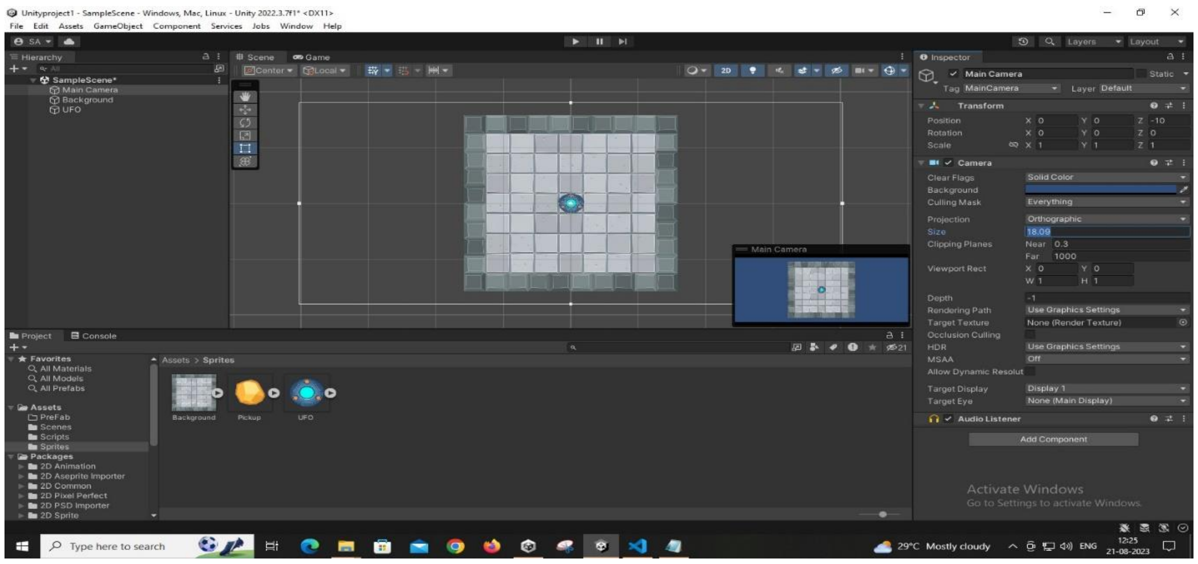


STEP 6 : Drag the assets to the Hierarchy > Click and hold asset and drag under MainCamera



SImilarly, Drag and add the UFO under the background

7 Adjust the MainCamera by changing the size



Unityproject1 - SampleScene - Windows, Mac, Linux - Unity 2022.3.7f1* <DX11>

Hierarchy: SampleScene* — Main Camera, Background, UFO

Inspector: Main Camera — Tag MainCamera, Layer Default

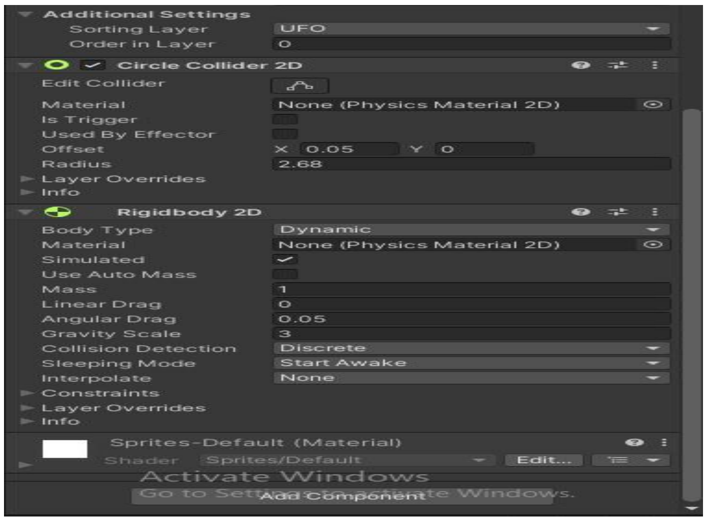
Transform
Position X 0 Y 0 Z -10
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Camera
Clear Flags: Solid Color
Culling Mask: Everything
Projection: Orthographic
Size: 18.09
Clipping Planes: Near 0.3, Far 1000
Viewport Rect: X 0 Y 0 W 1 H 1
Depth: -1
Rendering Path: Use Graphics Settings
Target Texture: None (Render Texture)
HDR: Use Graphics Settings
MSAA: Off
Target Display: Display 1
Target Eye: None (Main Display)

Audio Listener

Project: Assets > Sprites — Background, Pickup, UFO

STEP 8 : Select the UFO from Hierarchy. To make the UFO layer appear above the Background, Add a sorting layer, and select the layer. Update the following settings for UFO. Add a circle collider and adjust the radius.

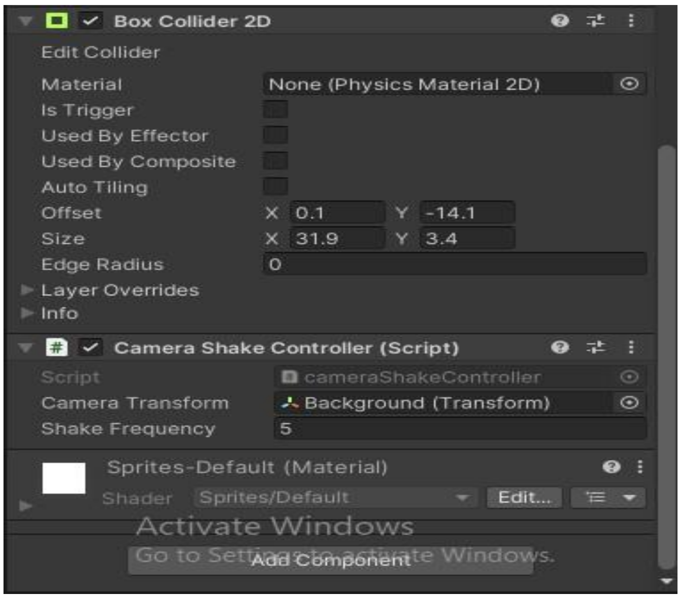


```
Additional Settings
  Sorting Layer           UFO
  Order in Layer          0
Circle Collider 2D
  Edit Collider
  Material                None (Physics Material 2D)
  Is Trigger
  Used By Effector
  Offset                  X 0.05   Y 0
  Radius                  2.68
  Layer Overrides
  Info
Rigidbody 2D
  Body Type               Dynamic
  Material                None (Physics Material 2D)
  Simulated               ✓
  Use Auto Mass
  Mass                    1
  Linear Drag             0
  Angular Drag            0.05
  Gravity Scale           3
  Collision Detection     Discrete
  Sleeping Mode           Start Awake
  Interpolate             None
  Constraints
  Layer Overrides
  Info
Sprites-Default (Material)
  Shader                  Sprites/Default     Edit...
Add Component
```

Add a Rigidbody 2D and Adjust gravity Scale

Rigidbodies are components that allow a GameObject to react to real-time physics. This includes reactions to forces and gravity, mass, drag and momentum. You can attach a Rigidbody to your GameObject by simply clicking on Add Component and typing in Rigidbody2D in the search field.

STEP 9 : Select the Background from Hierarchy. Add a box collider from Add component > Box Collider 2D. Do this step 4 times for each border of the box collider.



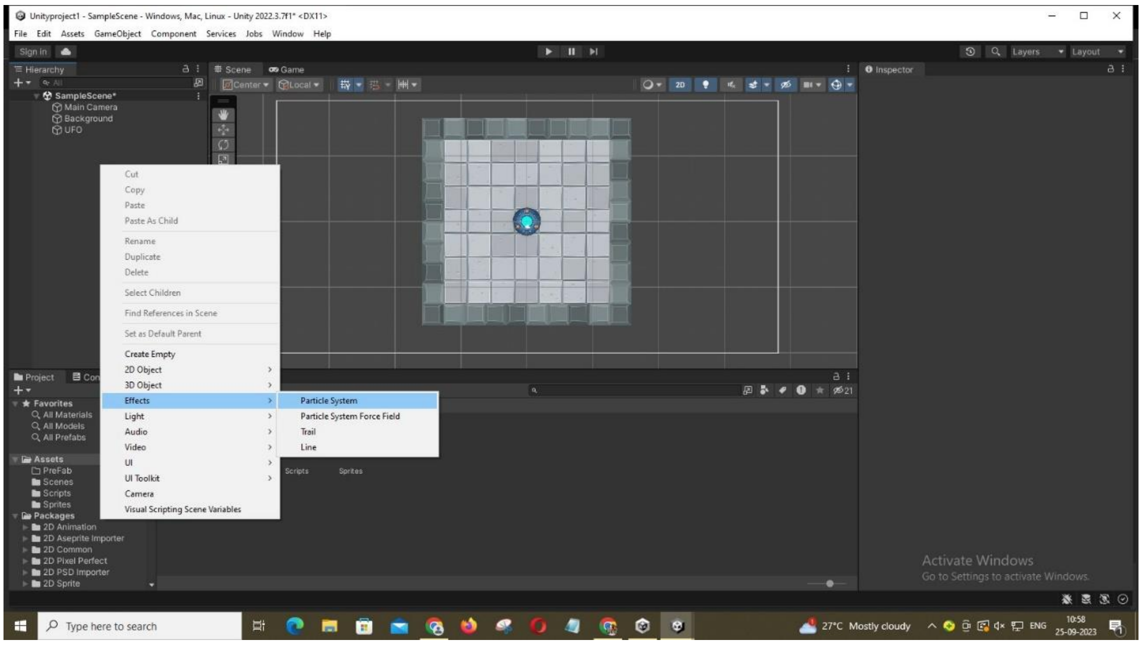
Box Collider 2D
Edit Collider
Material: None (Physics Material 2D)
Is Trigger
Used By Effector
Used By Composite
Auto Tiling
Offset: X 0.1 Y -14.1
Size: X 31.9 Y 3.4
Edge Radius: 0
Layer Overrides
Info

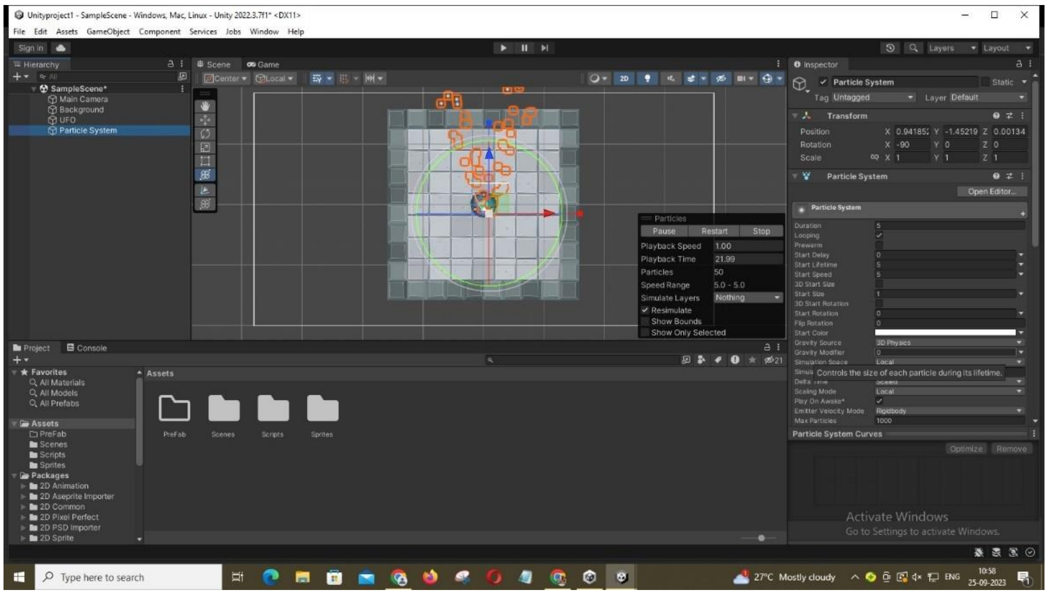
Camera Shake Controller (Script)
Script: cameraShakeController
Camera Transform: Background (Transform)
Shake Frequency: 5

Sprites-Default (Material)
Shader: Sprites/Default

STEP 10 : To add Particle effect, right-click on Effects under Hierarchy and

Select > Particle System





Playback Speed 1.00
Playback Time 21.99
Particles 50
Speed Range 5.0 - 5.0
Simulate Layers Nothing
✓ Resimulate
Show Bounds
Show Only Selected

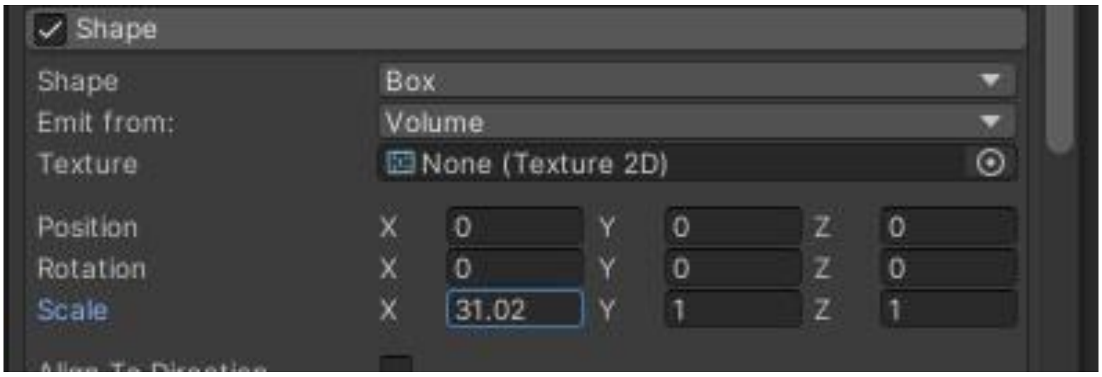
Particle System — Tag Untagged — Layer Default

	X	Y	Z
Position	0.941852	-1.45219	0.00134
Rotation	-90	0	0
Scale	1	1	1

Duration	5
Looping	✓
Prewarm	
Start Delay	0
Start Lifetime	5
Start Speed	5
3D Start Size	
Start Size	1
3D Start Rotation	
Start Rotation	0
Flip Rotation	0
Start Color	
Gravity Source	3D Physics
Gravity Modifier	0
Simulation Space	Local
Delta Time	Scaled
Scaling Mode	Local
Play On Awake*	✓
Emitter Velocity Mode	Rigidbody
Max Particles	1000

This is what the initial screen looks like.

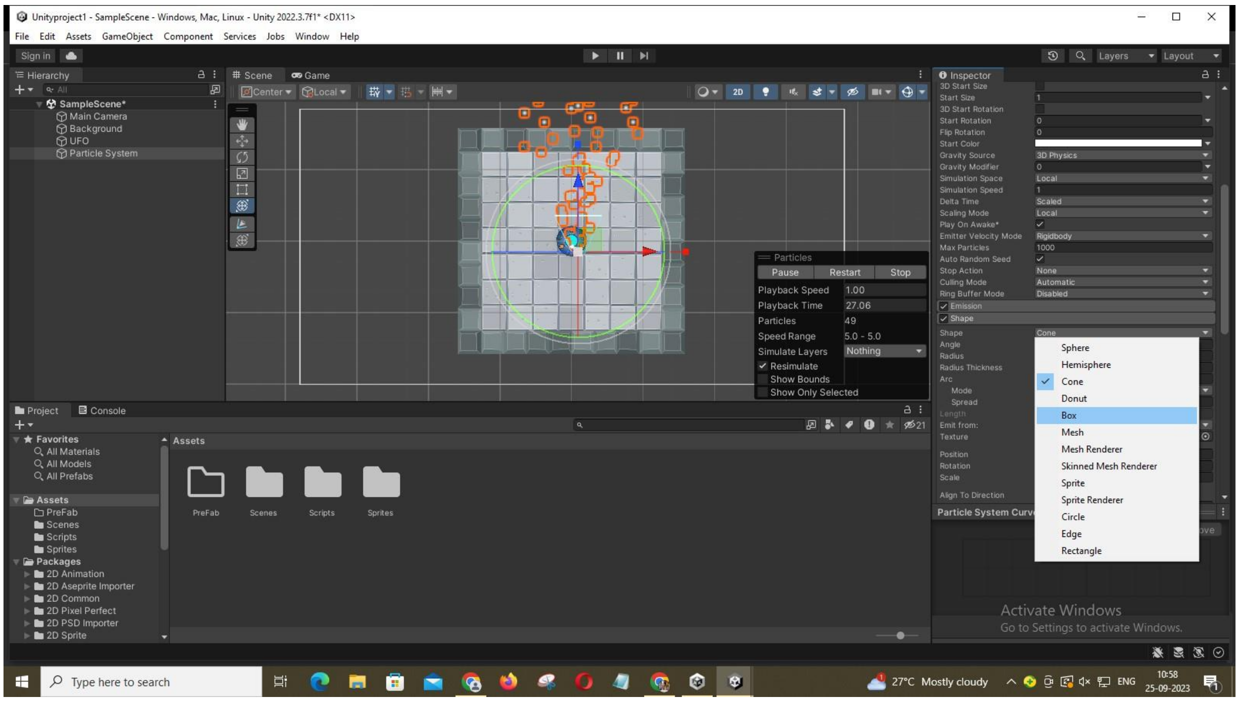
STEP 11 : Scroll down in Inspector to the Shape and Change it to **BOX** from Cone



```
✓ Shape
Shape          Box
Emit from:     Volume
Texture        None (Texture 2D)

Position   X 0      Y 0   Z 0
Rotation   X 0      Y 0   Z 0
Scale      X 31.02  Y 1   Z 1
```

STEP 12 : Adjust the 'X" Scale under Shape to adjust the width of the effect.

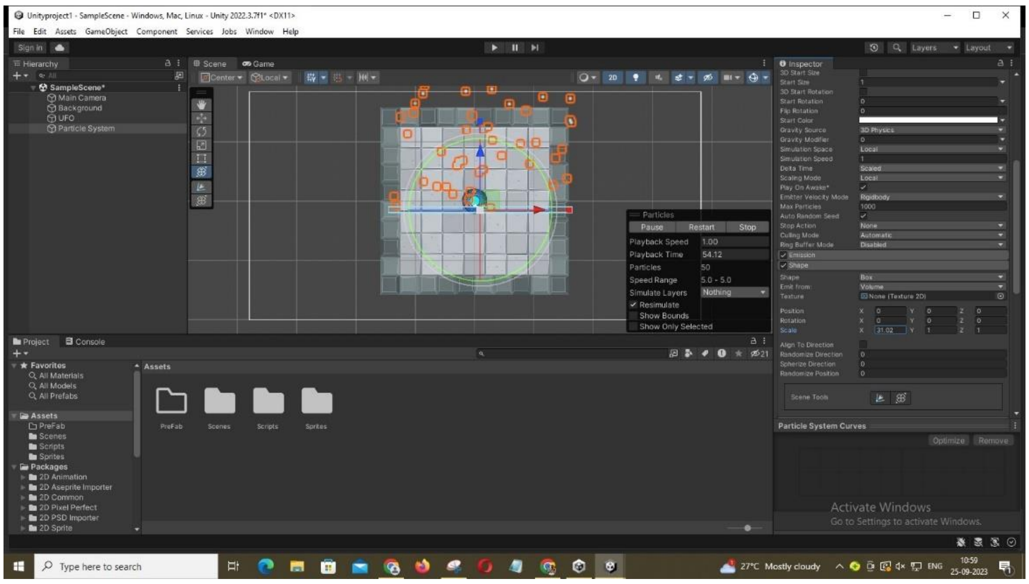
It will the look like this after Adjusting:



Unityproject1 - SampleScene - Windows, Mac, Linux - Unity 2022.3.7f1* <DX11>

```
Particles
Playback Speed    1.00
Playback Time     27.06
Particles         49
Speed Range       5.0 - 5.0
Simulate Layers   Nothing
✓ Resimulate
  Show Bounds
  Show Only Selected
```

```
Inspector
3D Start Size
Start Size              1
3D Start Rotation
Start Rotation          0
Flip Rotation           0
Start Color
Gravity Source          3D Physics
Gravity Modifier        0
Simulation Space        Local
Simulation Speed        1
Delta Time              Scaled
Scaling Mode            Local
Play On Awake*          ✓
Emitter Velocity Mode   Rigidbody
Max Particles           1000
Auto Random Seed        ✓
Stop Action             None
Culling Mode            Automatic
Ring Buffer Mode        Disabled
✓ Emission
✓ Shape
Shape                   Cone
```



Unityproject1 - SampleScene - Windows, Mac, Linux - Unity 2022.3.7f1* <DX11>

Hierarchy: SampleScene*, Main Camera, Background, UFO, Particle System

Particles

Pause | Restart | Stop

Playback Speed 1.00
Playback Time 54.12
Particles 50
Speed Range 5.0 - 5.0
Simulate Layers Nothing
✓ Resimulate
Show Bounds
Show Only Selected

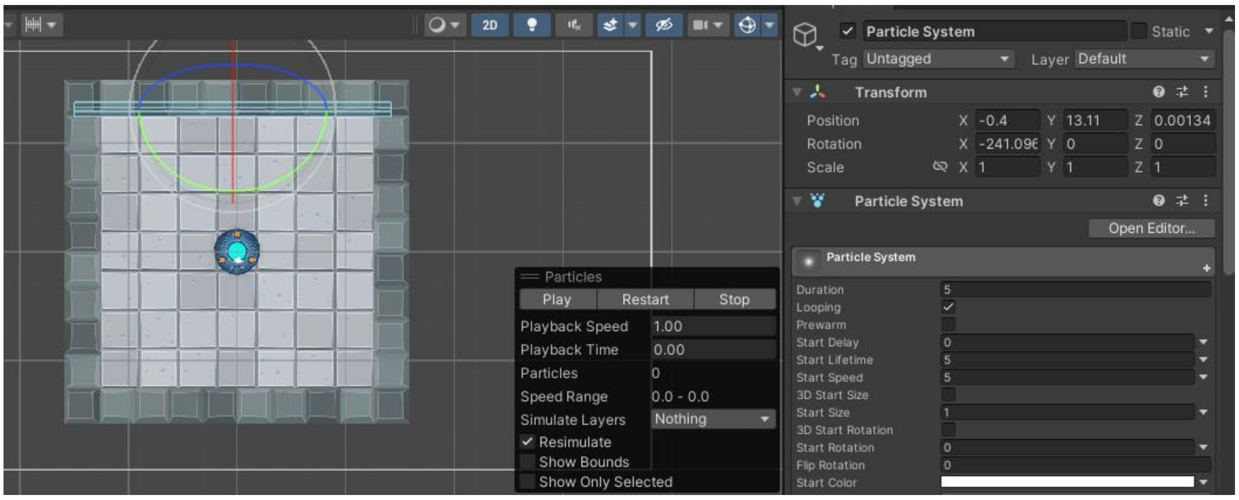
Inspector

3D Start Size	
Start Size	1
3D Start Rotation	
Start Rotation	0
Flip Rotation	0
Start Color	
Gravity Source	3D Physics
Gravity Modifier	0
Simulation Space	Local
Simulation Speed	1
Delta Time	Scaled
Scaling Mode	Local
Play On Awake*	✓
Emitter Velocity Mode	Rigidbody
Max Particles	1000
Auto Random Seed	✓
Stop Action	None
Culling Mode	Automatic
Ring Buffer Mode	Disabled
✓ Emission	
✓ Shape	
Shape	Box
Emit from	Volume
Texture	None (Texture 2D)
Position	X 0 Y 0 Z 0
Rotation	X 0 Y 0 Z 0
Scale	X 31.02 Y 1 Z 1
Align To Direction	
Randomize Direction	0
Spherize Direction	0
Randomize Position	0

Particle System Curves

Project: Assets – PreFab, Scenes, Scripts, Sprites

STEP 13 : Adjust ‘Y’ scale under Transform to move the particle upwards



Particles

Play | Restart | Stop

Playback Speed 1.00
Playback Time 0.00
Particles 0
Speed Range 0.0 - 0.0
Simulate Layers Nothing
✓ Resimulate
Show Bounds
Show Only Selected

✓ Particle System — Static
Tag Untagged — Layer Default

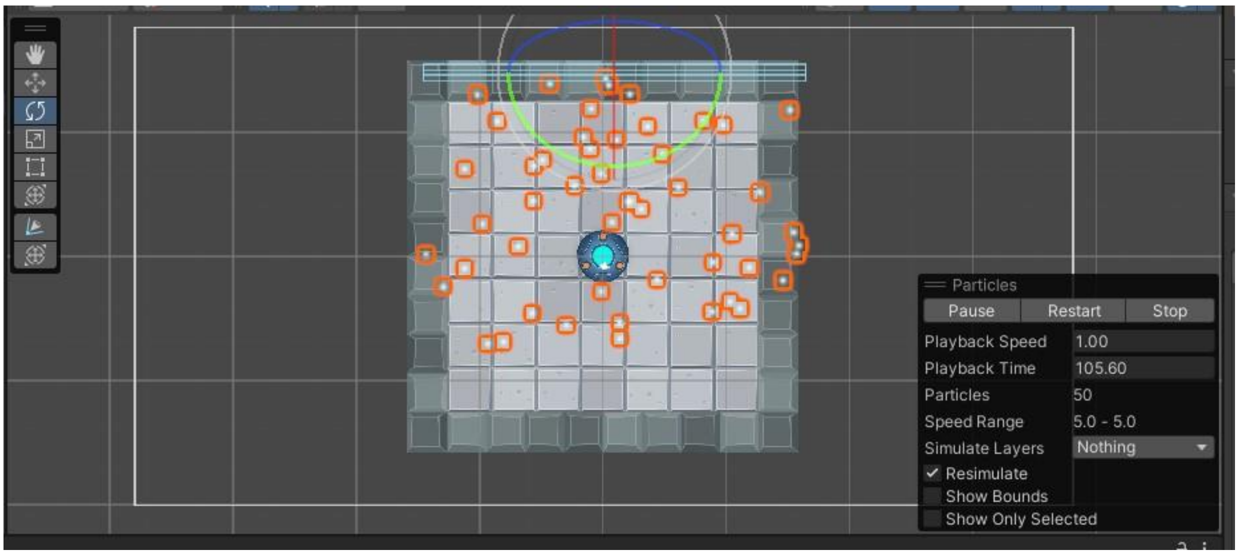
Transform

Position	X -0.4	Y 13.11	Z 0.00134
Rotation	X -241.096	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Particle System — Open Editor...

Duration	5
Looping	✓
Prewarm	
Start Delay	0
Start Lifetime	5
Start Speed	5
3D Start Size	
Start Size	1
3D Start Rotation	
Start Rotation	0
Flip Rotation	0
Start Color	

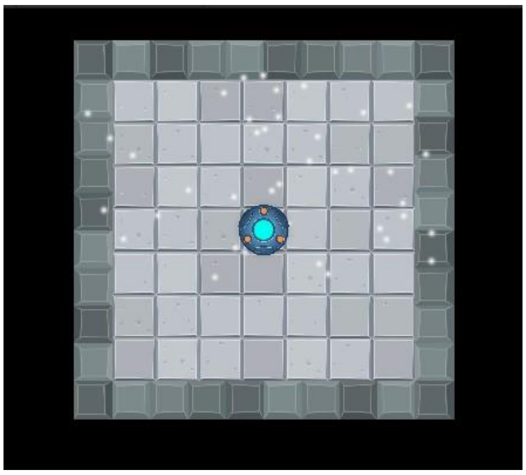
STEP 14 : Now, rotate the particle using the rotate tool present on the scene screen (third from top)



Particles

Playback Speed 1.00
Playback Time 105.60
Particles 50
Speed Range 5.0 - 5.0
Simulate Layers Nothing
✓ Resimulate
Show Bounds
Show Only Selected

The final outcome would be like this :



(Another example)