GAME PROGRAMMING JOURNAL

Kamal.J.Vasa

ROLL NO TCS2324087

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 Commerce
Sion(W), Mumbai – 400 022.

CERTIFICATE

This is to certify that Mr. **Kamal Vasa**
Roll No. **TCS2324087** Has successfully completed the necessary course of experiments in the subject of **GAME PROGRAMMING** during the academic year **2023 – 2024** complying with the requirements of **University of Mumbai**, for the course of **TYCS. Computer Science [Semester-5]**

Prof. In-Charge
SONI YADAV

Examination Date:
Examiner's Signature & Date:

Head of the Department
Prof. Manoj Singh

College Seal
And
Date

INDEX

Practical no	Aim	Date
1	Write a python program to perform translation operation on rectangle by taking initial coordinates from user.	
2	Write a python program to perform scaling operation on triangle by taking initial coordinates from user.	
3	Write a python program to perform reflection operation on polygon by taking initial coordinates from user.	
4	Write a python program to rotate right angle triangle by 45 degree by taking initial coordinates from user.	
5	Write a python program to perform shearing on rectangle in positive direction of x-axis by taking initial coordinates from user.	
6	Write a python program to create below shape and perform reflection about parallel to y-axis, followed by translation and scaling operation on it.	
7	Implement space invader game in python using pygame module.	
8	Implement Snake game in python using pygame module.	
9	Implement 2D UFO game using unity.	
10	Implement 3D roll ball game using unity.	

Practical No1:

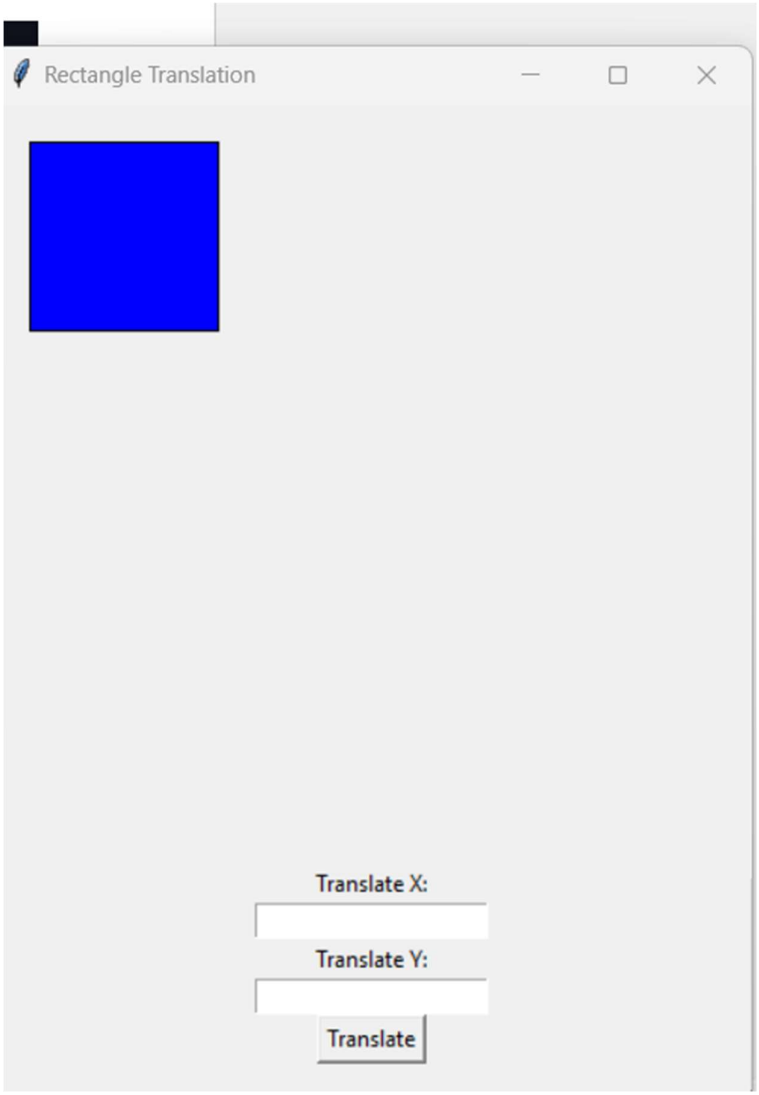
Aim: Write a python program to perform translation operation on rectangle by taking initial coordinates from user.

CODE:

```
import tkinter as tk

# Function to translate the rectangle
def translate_rectangle():
    # Get the translation values from the user
    try:
        dx = float(dx_entry.get())
        dy = float(dy_entry.get())
    except ValueError:
        result_label.config(text="Invalid input")
        return

    # Update the rectangle's coordinates
    canvas.move(rectangle, dx, dy)

    # Clear the result label
    result_label.config(text="")

# Create the main window
root = tk.Tk()
root.title("Rectangle Translation")

# Create a canvas to draw the rectangle
canvas = tk.Canvas(root, width=400, height=400)
canvas.pack()

# Get initial coordinates from the user
x1 = float(input("Enter initial x-coordinate of the top-left corner: "))
y1 = float(input("Enter initial y-coordinate of the top-left corner: "))
x2 = float(input("Enter initial x-coordinate of the bottom-right corner: "))
y2 = float(input("Enter initial y-coordinate of the bottom-right corner: "))

# Create a rectangle on the canvas
rectangle = canvas.create_rectangle(x1, y1, x2, y2, fill="blue")

# Create input fields for translation values
dx_label = tk.Label(root, text="Translate X:")
```

```
dx_label.pack()
dx_entry = tk.Entry(root)
dx_entry.pack()

dy_label = tk.Label(root, text="Translate Y:")
dy_label.pack()
dy_entry = tk.Entry(root)
dy_entry.pack()

# Create a button to perform translation
translate_button = tk.Button(root, text="Translate",
command=translate_rectangle)
translate_button.pack()

# Label to display the result or error message
result_label = tk.Label(root, text="")
result_label.pack()

# Start the Tkinter main loop
root.mainloop()
```

OUTPUT:



Practical No2:

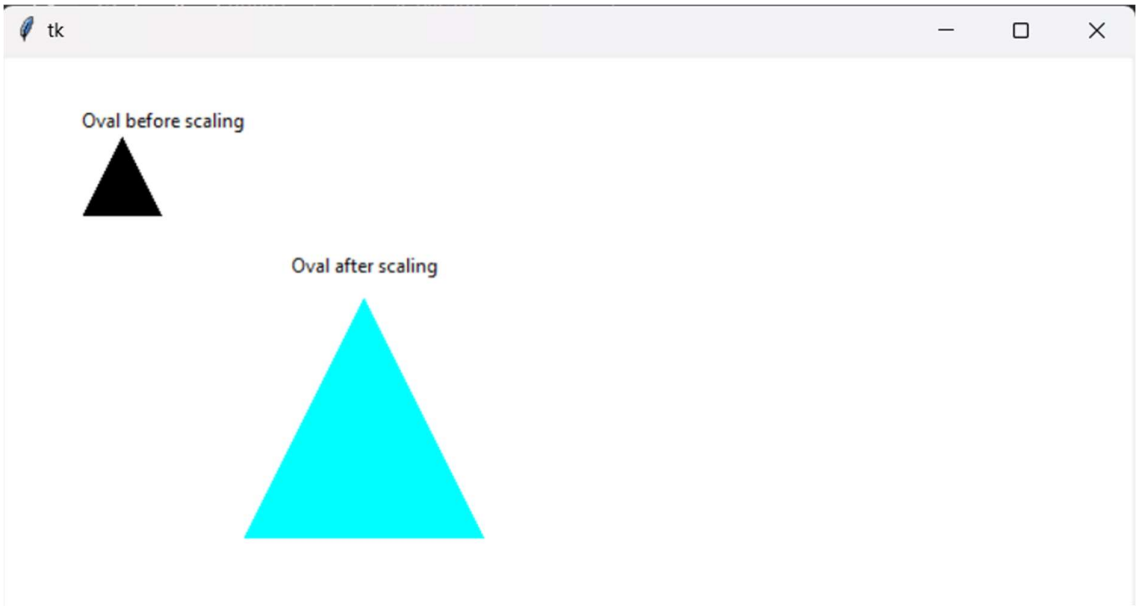
Aim: Write a python program to perform scaling operation on triangle by taking initial coordinates from user.

CODE:

```
from tkinter import *

root = Tk()
C = Canvas(root, bg="gray", height=700, width=700)
C.create_text(100, 40, text="Triangle Before scaling", fill="black",
font=('Consolas'))
x0 = int(input("Enter x0:- "))
y0 = int(input("Enter y0:- "))
x1 = int(input("Enter x1:- "))
y1 = int(input("Enter y1:- "))
x2 = int(input("Enter x2:- "))
y2 = int(input("Enter y2:- "))
C.create_polygon(x0, y0, x1, y1, x2, y2, fill="green")
sx = 2
sy = 2

# Calculate the scaled coordinates
x0_scaled = x0 * sx
y0_scaled = y0 * sy
x1_scaled = x1 * sx
y1_scaled = y1 * sy
x2_scaled = x2 * sx
y2_scaled = y2 * sy




# Create the text elements after creating the triangles
C.create_text(225, 130, text="Triangle After scaling", fill="black",
font=('Consolas'))
# Create the scaled triangle
C.create_polygon(x0_scaled, y0_scaled, x1_scaled, y1_scaled, x2_scaled,
y2_scaled, fill="blue")




C.pack()
```

```
mainloop()
```

OUTPUT:



Practical No3:

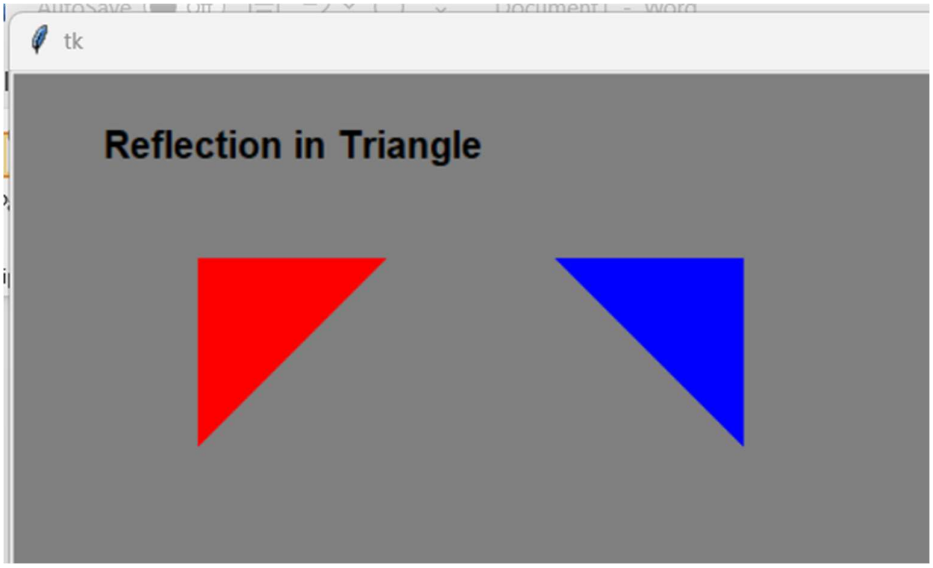
Aim: Write a python program to perform reflection operation on polygon by taking initial coordinates from user.

CODE:

```
#Write a python program to reflect polygon about an arbitary  point

import math
from tkinter import *

root = Tk()

C = Canvas(root, bg="gray", height=1000, width=1000)

x0 = int(input("Enter the x0 coordinate: "))
y0 = int(input("Enter the y0 coordinate: "))
x1 = int(input("Enter the x1 coordinate: "))
y1 = int(input("Enter the y1 coordinate: "))
x2 = int(input("Enter the x2 coordinate: "))
y2 = int(input("Enter the y2 coordinate: "))

C.create_text(150, 40, text="Reflection in Triangle", fill="black",
font=('Helvetica 15 bold'))

# Original triangle
triangle_1 = C.create_polygon(x0, y0, x1, y1, x2, y2, fill="red")
width = max(x0, x1, x2) - min(x0, x1, x2)

# Imaginary axis
a = x1 - 10  # Adjust the imaginary axis as needed
```

```
# Calculate the reflected coordinates of the vertices
x0_reflected = (2 * a - x0)+width+10
x1_reflected = (2 * a - x1)+width+10
x2_reflected = (2 * a - x2)+width+10

# Reflected triangle
triangle_2 = C.create_polygon(x0_reflected, y0, x1_reflected, y1,
x2_reflected, y2, fill="blue")

C.pack()
mainloop()
```

OUTPUT:

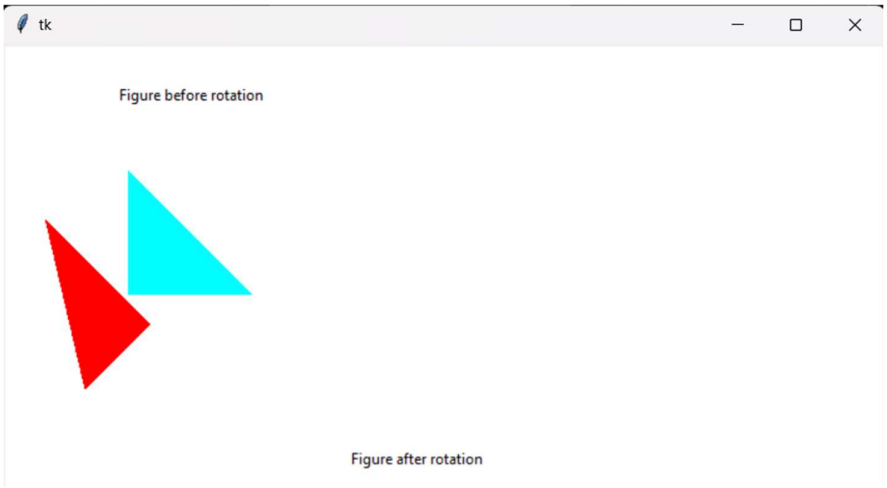


Practical No4:

Aim: Write a python program to rotate right angle triangle by 45 degree by taking initial coordinates from user.

```
import math
from tkinter import *
root=Tk()
n=int(input("Enter angle"))
a=math.cos(math.radians(n))
b=math.sin(math.radians(n))
x0=int(input("Enter x0:- "))
y0=int(input("Enter y0:- "))
x1=int(input("Enter x1:- "))
y1=int(input("Enter y1:- "))
x2=int(input("Enter x2:- "))
y2=int(input("Enter y2:- "))

C=Canvas(root,bg="gray",height=1000,width=1000)
C.create_text(150,40,text="Triangle Before
rotation",fill="green",font=('Consolas'))
triangle1=C.create_polygon(x0,y0,x1,y1,x2,y2,fill="green")
x11=abs(x0*a-y0*b)
y11=abs(x0*b+y0*a)
x12=abs(x1*a-y1*b)
y12=abs(x1*b+y1*a)
x13=abs(x2*a-y2*b)
y13=abs(x2*b+y2*a)
C.create_text(150,200,text="Triangle After
rotation",fill="blue",font=('Consolas'))
triangle2=C.create_polygon(x11,y11,x12,y12,x13,y13,fill="blue")

C.pack()
mainloop()
```

OUTPUT:



Practical No5:

Aim: Write a python program to perform shearing on rectangle in positive direction of x-axis by taking initial coordinates from user.

```
#Write a program in python to implement shearing of a triangle

from tkinter import*
import math
root=Tk();
C=Canvas(root,bg="gray",height=1000,width=1000)

#Taking  input of coordinates from the user

x0=int(input("Enter x0"))

y0=int(input("Enter y0"))
x1=int(input("Enter x1"))
y1=int(input("Enter y1"))
x2=int(input("Enter x2"))
y2=int(input("Enter y2"))
x3=int(input("Enter x3"))
y3=int(input("Enter y3"))


b=int(input("Enter angle of Shearing "))

shape1=C.create_polygon(x0,y0,x1,y1,x2,y2,x3,y3,fill="green")

xsh1=(x0+y0*math.tan(math.radians(b)))
xsh2=(x3+y3*math.tan(math.radians(b)))

shape1=C.create_polygon(xsh1,y0,x1,y1,x2,y2,xsh2,y3,fill="blue")




C.pack();
mainloop()
```

Practical No 6: Write a python program to create below shape and perform reflection about parallel to y-axis, followed by translation and scaling operation on it.

```
#Prac 6
import math
from tkinter import*
root = Tk()
C = Canvas(root,bg="yellow",height=1000,width=1200)
C.create_text(100,40,text="Reflection",fill="black")
x1=50
y1=10
x2=25
y2=30
x3=50
y3=50
x4=50
y4=40
x5=150
y5=40
x6=150
y6=20
x7=50
y7=20
C.create_polygon(x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6,x7,y7,fill="black")
a=int(input("Enter arbituary axis: "))
y11=(-y1+(2*a))
y21=(-y2+(2*a))
y31=(-y3+(2*a))
y41=(-y4+(2*a))
y51=(-y5+(2*a))
y61=(-y6+(2*a))
y71=(-y7+(2*a))
C.create_polygon(x1,y11,x2,y21,x3,y31,x4,y41,x5,y51,x6,y61,x7,y71,fill="black"
)
tx=500
```

```
ty=500
C.create_polygon(x1+tx,y11+ty,x2+tx,y21+ty,x3+tx,y31+ty,x4+tx,y41+ty,x5+tx,y51
+ty,x6+tx,y61+ty,x7+tx,y71+ty,fill="red")
sx=2
sy=2
C.create_polygon(x1*sx,y11*sy,x2*sx,y21*sy,x3*sx,y31*sy,x4*sx,y41*sy,x5*sx,y51
*sy,x6*sx,y61*sy,x7*sx,y71*sy,fill="red")
C.pack()
root.mainloop()
```

OUTPUT:

Practical No7: Implement space invader game in python using pygame module.

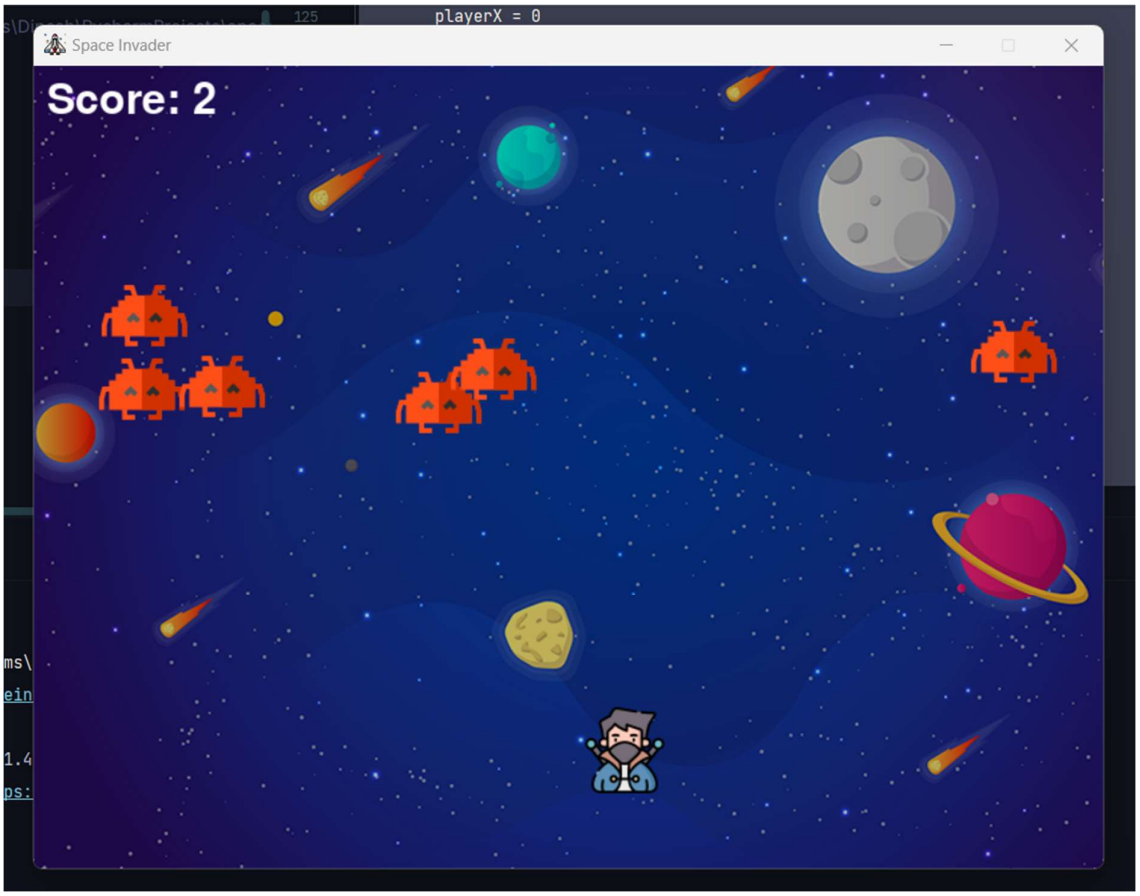
CODE:

```
import math
import pygame
import random
from pygame import mixer

# initialization
pygame.init()
# display size
screen = pygame.display.set_mode((800, 600))
# Title
pygame.display.set_caption("Space Invader")
# Icon
icon = pygame.image.load("spaceship.png")
# Background sound
mixer.music.load("background.wav")
mixer.music.play(-1)
pygame.display.set_icon(icon)
# player
playerImg = pygame.image.load("assasin.png")
playerX = 370
playerY = 480
playerX_change = 0
# Enemy
enemyImg = []
enemyX = []
enemyY = []
enemyX_change = []
enemyY_change = []
no_of_enemies = 6
for i in range(no_of_enemies):
    enemyImg.append(pygame.image.load("enemy (1).png"))
    enemyX.append(random.randint(0, 735))
    enemyY.append(random.randint(60, 150))
    enemyX_change.append(3)
    enemyY_change.append(40)

# Background
bgImg = pygame.image.load("background (1).png")
# Bullet
bulletImg = pygame.image.load("bullet.png")
bulletX = 0
bulletY = 480
bulletX_change = 0
bulletY_change = 10
bullet_state = "ready"
score_value = 0
# Text for score display
font = pygame.font.Font("freesansbold.ttf", 32)
textX = 10
textY = 10
# Game over text
game_over_font = pygame.font.Font("freesansbold.ttf", 64)


# Gameover Function
def game_over_text():
    gameover = game_over_font.render("Game Over", True, (255, 255, 255))
    screen.blit(gameover, (200, 250))


# Score Function
def show_score(x, y):
    score = font.render("Score: " + str(score_value), True, (255, 255, 255))
```

```
    screen.blit(score, (x, y))


# Collision function
def isCollision(enemyX, enemyY, bulletX, bulletY):
    distance = math.sqrt((math.pow(enemyX - bulletX, 2)) + (math.pow(enemyY -
bulletY, 2)))
    if distance < 27:
        return True
    else:
        return False


def player(x, y):
    screen.blit(playerImg, (x, y))


def enemy(x, y, i):
    screen.blit(enemyImg[i], (x, y))


def fire_bullet(x, y):
    global bullet_state
    bullet_state = "fire"
    screen.blit(bulletImg, (x + 16, y + 10))


running = True
# game window stays on until close
while running:
    screen.fill((0, 0, 0))  # Background Color
    screen.blit(bgImg, (0, 0))  # background Img
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                playerX_change = -4
            if event.key == pygame.K_RIGHT:
                playerX_change = 4
            if event.key == pygame.K_SPACE:
                if bullet_state == "ready":
                    bullet_sound = mixer.Sound("laser.wav")
                    bullet_sound.play()

                    bulletX = playerX
                    fire_bullet(bulletX, bulletY)

        if event.type == pygame.KEYUP:
            if event.key == pygame.K_LEFT or event.key == pygame.K_RIGHT:
                playerX_change = 0
    # Bullet Movement
    if bulletY <= 0:
        bulletY = 480
        bullet_state = "ready"
    if bullet_state is "fire":
        fire_bullet(bulletX, bulletY)
        bulletY -= bulletY_change

    playerX += playerX_change
    # player boundary
    if playerX <= 0:
        playerX = 0
    elif playerX >= 736:
        playerX = 736  # size of player is 64, so we subtract 64 from 800

    # player boundary
    for i in range(no_of_enemies):
```

```
        # gameover code
        if enemyY[i] > 400:
            for j in range(no_of_enemies):
                enemyY[i] = 2000
            game_over_text()
            break
        enemyX[i] += enemyX_change[i]
        if enemyX[i] <= 0:
            enemyX_change[i] = 3
            enemyY[i] += enemyY_change[i]
        elif enemyX[i] >= 735:
            enemyX_change[i] = -3
            enemyY[i] += enemyY_change[i]
        # Checking collision
        collision = isCollision(enemyX[i], enemyY[i], bulletX, bulletY)
        if collision:
            explosion_sound = mixer.Sound("explosion.wav")
            explosion_sound.play()
            bulletY = 480
            bullet_state = "ready"
            score_value += 1
            enemyX[i] = random.randint(0, 735)
            enemyY[i] = random.randint(50, 150)
        enemy(enemyX[i], enemyY[i], i)

    show_score(textX, textY)
    player(playerX, playerY)
    pygame.display.update()
```

OUTPUT:



Practical No8: Implement Snake game in python using pygame module.

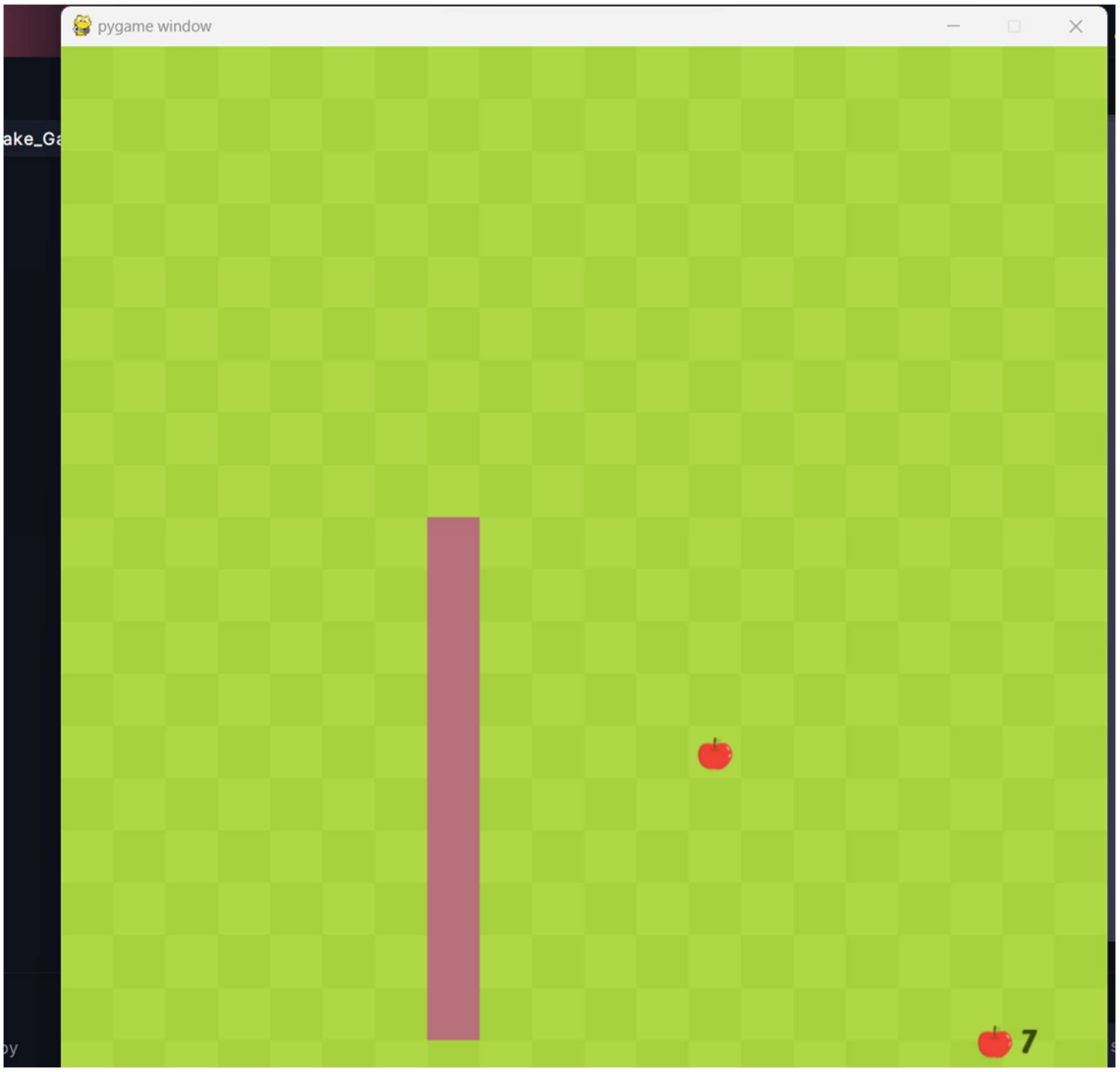
CODE:

```
import pygame, sys, random
from pygame.math import Vector2


class SNAKE:
    def __init__(self):
        self.body = [Vector2(5, 10), Vector2(4, 10), Vector2(3, 10)]
        self.direction = Vector2(1, 0)
        self.new_block = False

    def draw_snake(self):
        for block in self.body:
            x_pos = int(block.x * cell_size)
            y_pos = int(block.y * cell_size)
            block_rect = pygame.Rect(x_pos, y_pos, cell_size, cell_size)
            pygame.draw.rect(screen, (183, 111, 122), block_rect)

    def move_snake(self):
        if self.new_block == True:
            body_copy = self.body[:]
            body_copy.insert(0, body_copy[0] + self.direction)
            self.body = body_copy[:]
            self.new_block = False
        else:
            body_copy = self.body[:-1]
            body_copy.insert(0, body_copy[0] + self.direction)
            self.body = body_copy[:]

    def add_block(self):
        self.new_block = True

    def reset(self):
        self.body = [Vector2(5, 10), Vector2(4, 10), Vector2(3, 10)]



class FRUIT:
    def __init__(self):
        self.randomize()

    def draw_fruit(self):
        fruit_rect = pygame.Rect(int(self.pos.x * cell_size), int(self.pos.y *
cell_size), cell_size, cell_size)
        screen.blit(apple, fruit_rect)
        # pygame.draw.rect(screen, (126, 166, 114), fruit_rect)

    def randomize(self):
        self.x = random.randint(0, cell_number - 1)
        self.y = random.randint(0, cell_number - 1)
        self.pos = Vector2(self.x, self.y)


class MAIN:
    def __init__(self):
        self.snake = SNAKE()
```

```
        self.fruit = FRUIT()

    def update(self):
        self.snake.move_snake()
        self.check_collision()
        self.check_fail()

    def draw_elements(self):
        self.draw_grass()
        self.fruit.draw_fruit()
        self.snake.draw_snake()
        self.draw_score()

    def check_collision(self):
        if self.fruit.pos == self.snake.body[0]:
            self.fruit.randomize()
            self.snake.add_block()

        for block in self.snake.body[1:]:
            if block == self.fruit.pos:
                self.fruit.randomize()

    def check_fail(self):
        if not 0 <= self.snake.body[0].x < cell_number or not 0 <=
self.snake.body[0].y < cell_number:
            self.game_over()

        for block in self.snake.body[1:]:
            if block == self.snake.body[0]:
                self.game_over()

    def game_over(self):
        self.snake.reset()


    def draw_grass(self):
        grass_color = (167, 209, 61)
        for row in range(cell_number):
            if row % 2 == 0:
                for col in range(cell_number):
                    if col % 2 == 0:
                        grass_rect = pygame.Rect(col * cell_size, row * cell_size,
cell_size, cell_size)
                        pygame.draw.rect(screen, grass_color, grass_rect)
            else:
                for col in range(cell_number):
                    if col % 2 != 0:
                        grass_rect = pygame.Rect(col * cell_size, row * cell_size,
cell_size, cell_size)
                        pygame.draw.rect(screen, grass_color, grass_rect)

    def draw_score(self):
        score_text = str(len(self.snake.body) - 3)
        score_surface = game_font.render(score_text, True, (56, 74, 12))
        score_x = int(cell_size * cell_number - 60)
        score_y = int(cell_size * cell_number - 40)
        score_rect = score_surface.get_rect(center=(score_x, score_y))
        apple_rect = apple.get_rect(midright=(score_rect.left, score_rect.centery))

        screen.blit(score_surface, score_rect)
        screen.blit(apple, apple_rect)


pygame.init()
cell_size = 40
cell_number = 20
screen = pygame.display.set_mode((cell_number * cell_size, cell_number *
cell_size))
```

```
clock = pygame.time.Clock()
apple = pygame.image.load('apple_2.png').convert_alpha()
game_font = pygame.font.Font('Fonts/PoetsenOne-Regular.ttf', 24)

SCREEN_UPDATE = pygame.USEREVENT
pygame.time.set_timer(SCREEN_UPDATE, 150)

main_game = MAIN()

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
        if event.type == SCREEN_UPDATE:
            main_game.update()
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP:
                if main_game.snake.direction.y != 1:
                    main_game.snake.direction = Vector2(0, -1)
            if event.key == pygame.K_DOWN:
                if main_game.snake.direction.y != -1:
                    main_game.snake.direction = Vector2(0, 1)
            if event.key == pygame.K_RIGHT:
                if main_game.snake.direction.x != -1:
                    main_game.snake.direction = Vector2(1, 0)
            if event.key == pygame.K_LEFT:
                if main_game.snake.direction.x != 1:
                    main_game.snake.direction = Vector2(-1, 0)

    screen.fill(((175, 215, 70)))
    main_game.draw_elements()

    pygame.display.update()
    clock.tick(60)  # frame rate
```

OUTPUT:

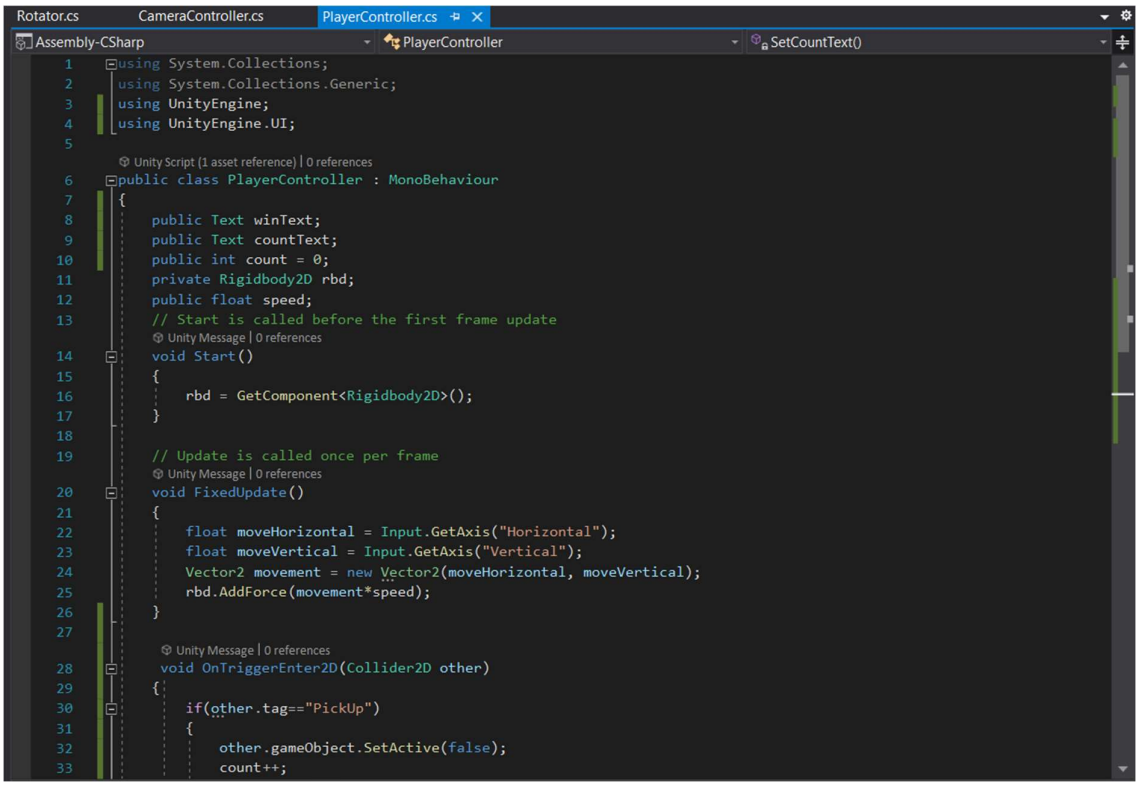


Practical No9:

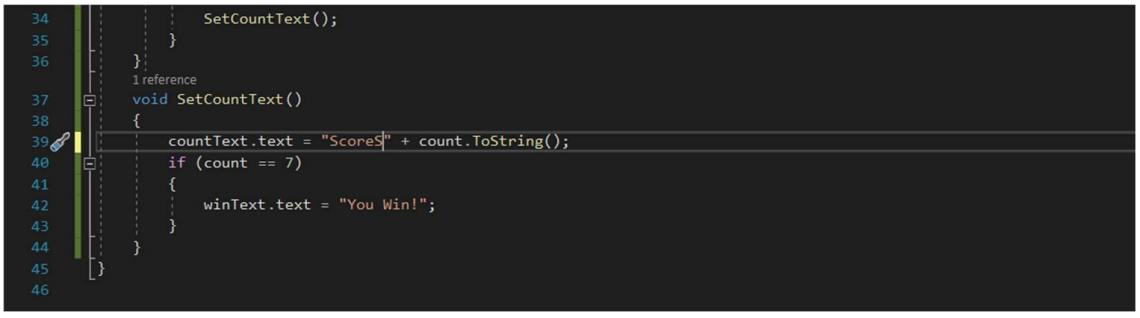
Aim: Implement 2D UFO game using unity

Code:

PlayerController.cs

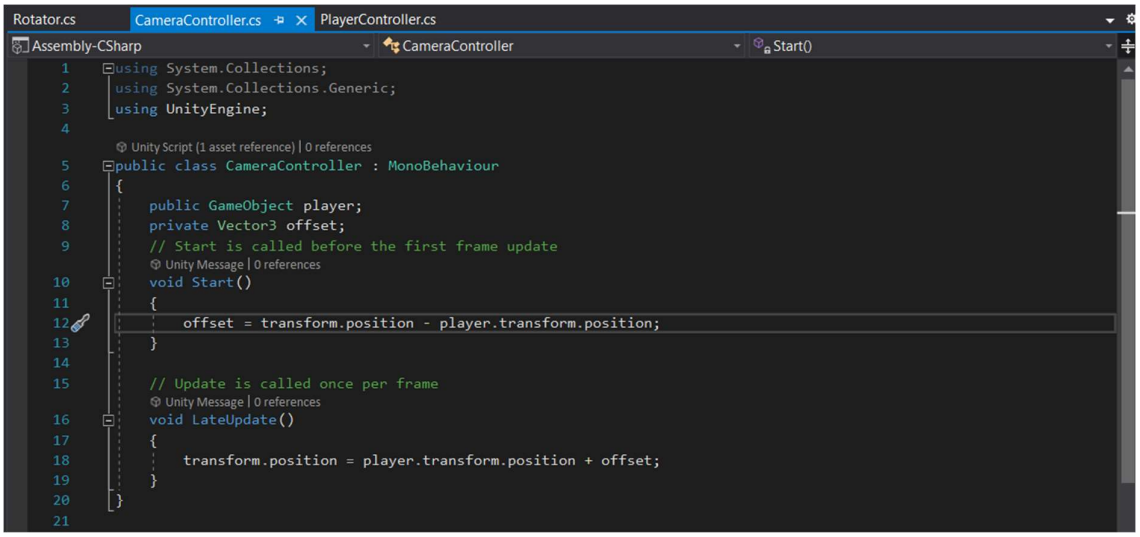


```
Rotator.cs   CameraController.cs   PlayerController.cs
Assembly-CSharp   PlayerController   SetCountText()
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
   Unity Script (1 asset reference) | 0 references
6  public class PlayerController : MonoBehaviour
7  {
8      public Text winText;
9      public Text countText;
10     public int count = 0;
11     private Rigidbody2D rbd;
12     public float speed;
13     // Start is called before the first frame update
       Unity Message | 0 references
14     void Start()
15     {
16         rbd = GetComponent<Rigidbody2D>();
17     }
18
19     // Update is called once per frame
       Unity Message | 0 references
20     void FixedUpdate()
21     {
22         float moveHorizontal = Input.GetAxis("Horizontal");
23         float moveVertical = Input.GetAxis("Vertical");
24         Vector2 movement = new Vector2(moveHorizontal, moveVertical);
25         rbd.AddForce(movement*speed);
26     }
27
       Unity Message | 0 references
28     void OnTriggerEnter2D(Collider2D other)
29     {
30         if(other.tag=="PickUp")
31         {
32             other.gameObject.SetActive(false);
33             count++;
```

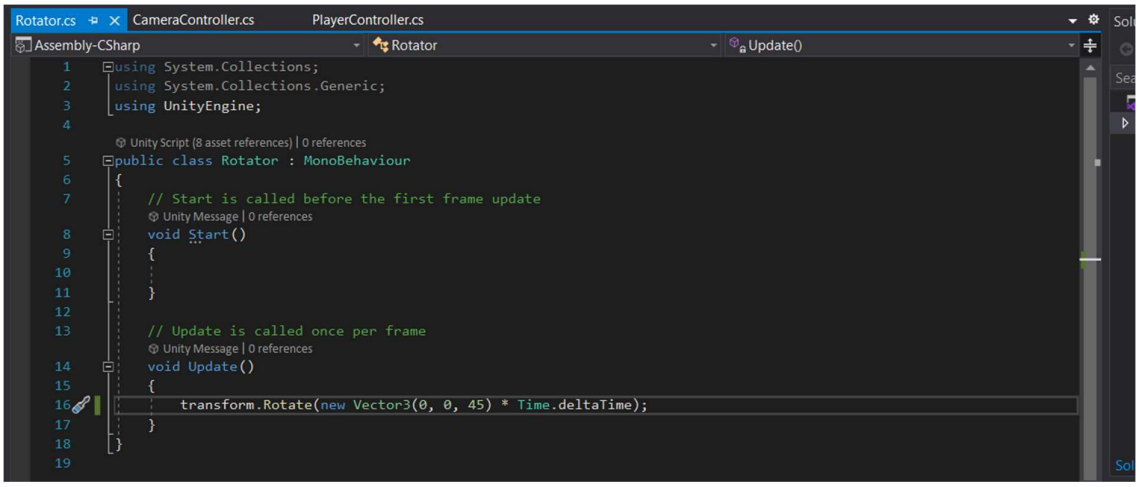


```
34             SetCountText();
35         }
36     }
       1 reference
37     void SetCountText()
38     {
39         countText.text = "ScoreS" + count.ToString();
40         if (count == 7)
41         {
42             winText.text = "You Win!";
43         }
44     }
45 }
46
```

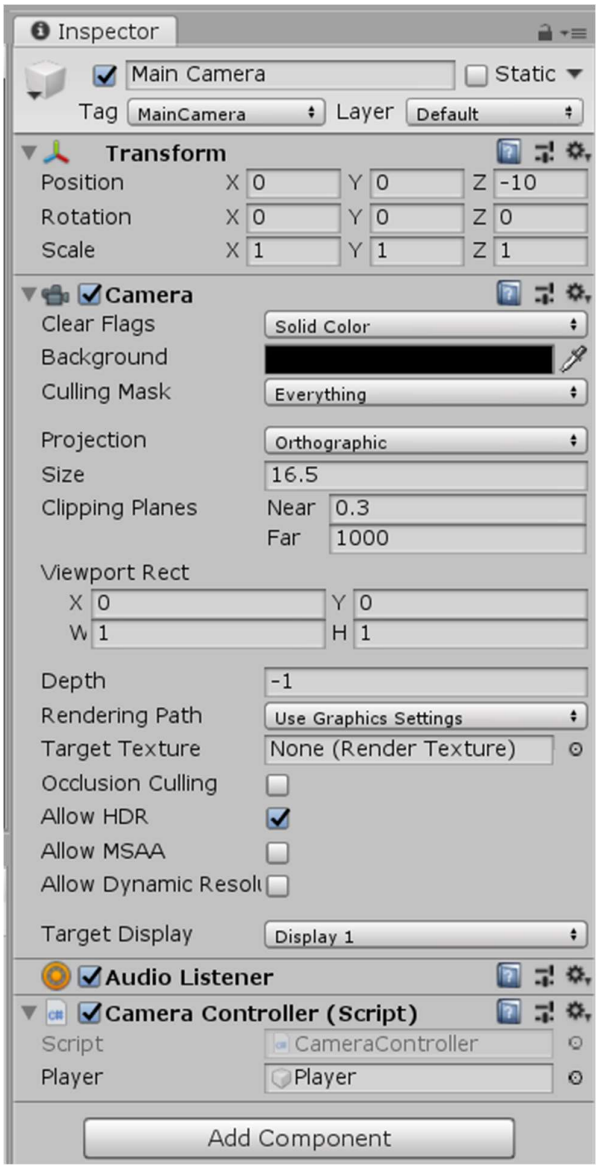
CameraController.cs



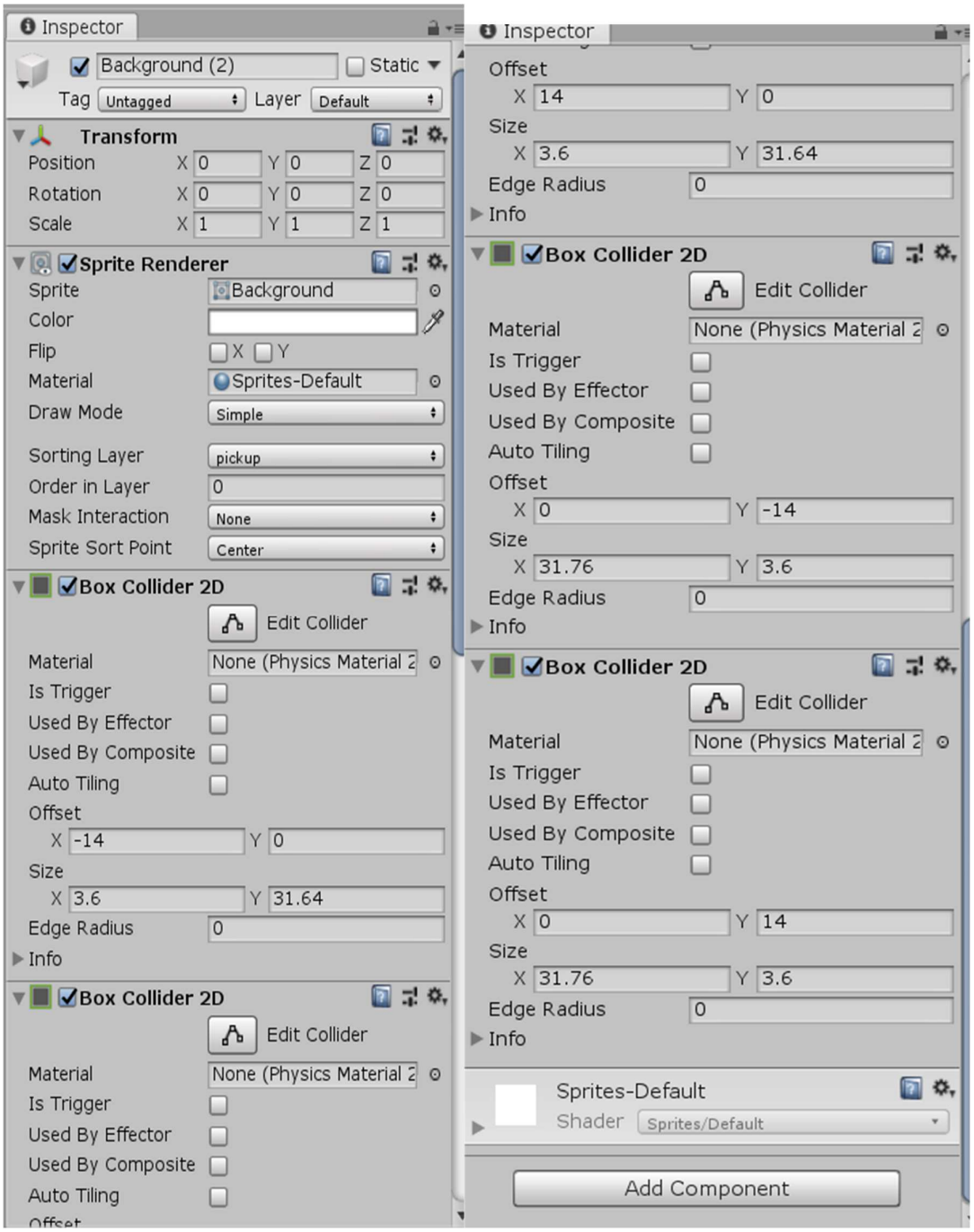
Rotator.cs



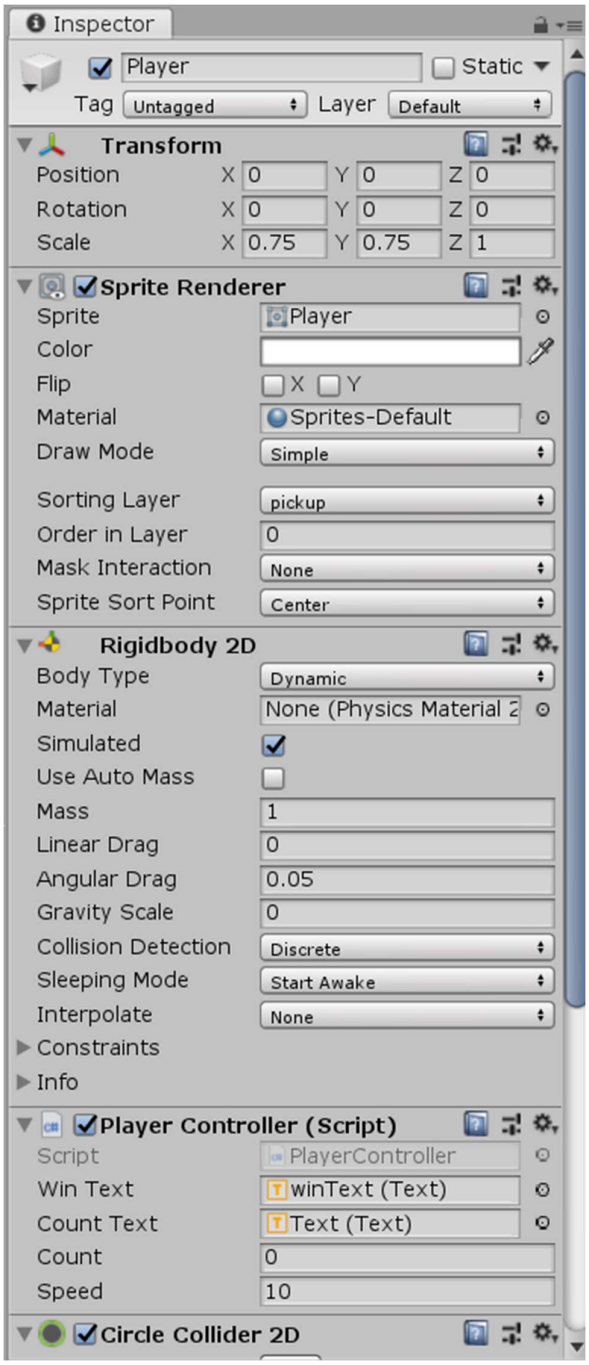
Main Camera Inspector



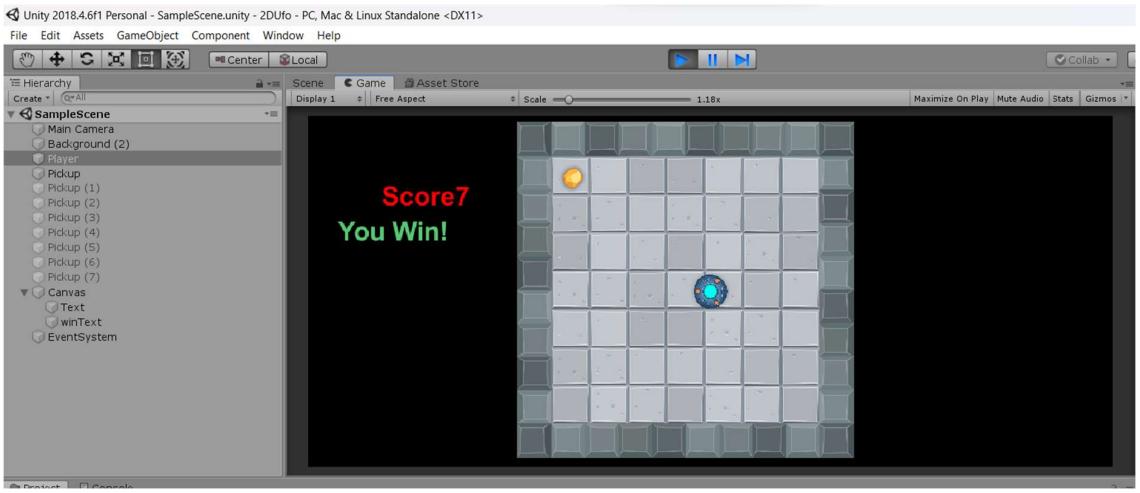
Background Inspector



Player Inspector



Output:

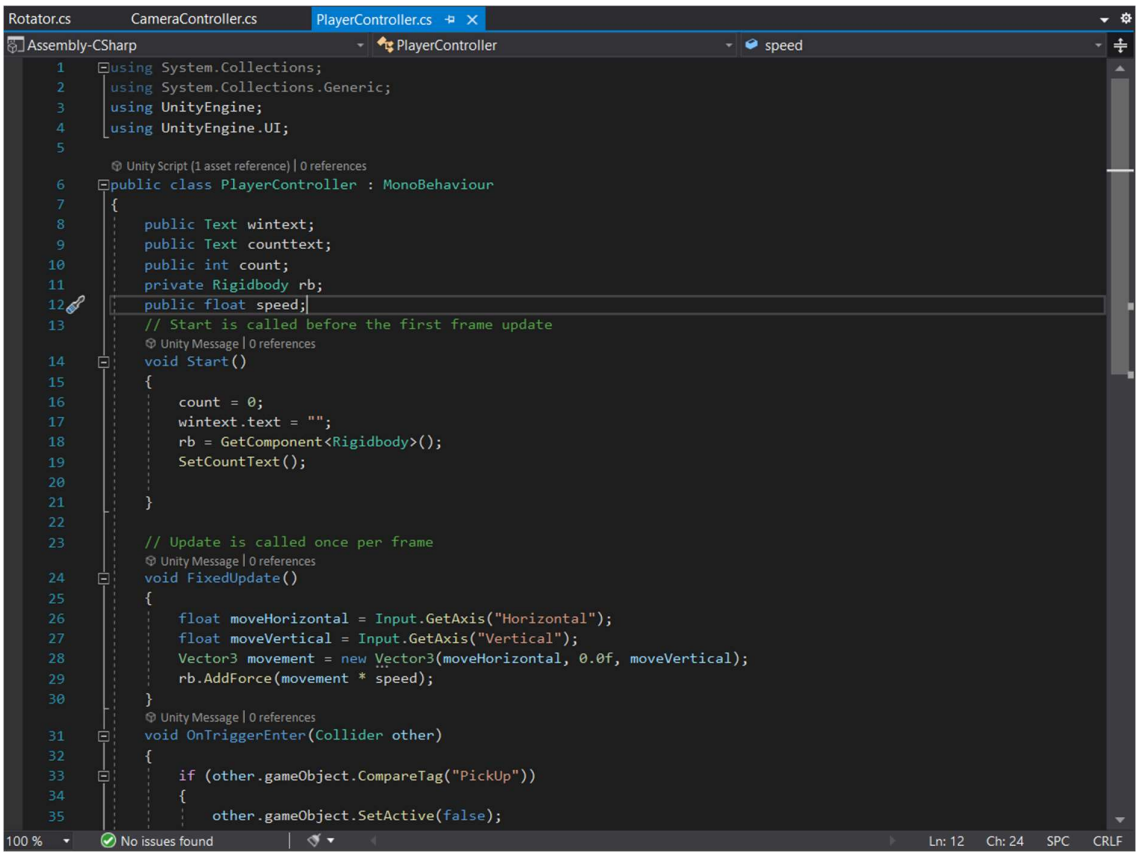

Unity 2018.4.6f1 Personal - SampleScene.unity - 2DUfo - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Collab
Hierarchy
Create
SampleScene
Main Camera
Background (2)
Player
Pickup
Pickup (1)
Pickup (2)
Pickup (3)
Pickup (4)
Pickup (5)
Pickup (6)
Pickup (7)
Canvas
Text
winText
EventSystem
Scene
Game
Asset Store
Display 1
Free Aspect
Scale
1.18x
Maximize On Play
Mute Audio
Stats
Gizmos
Score7
You Win!

Practical No10:

Aim: Implement 3D roll ball game using unity.

Code:

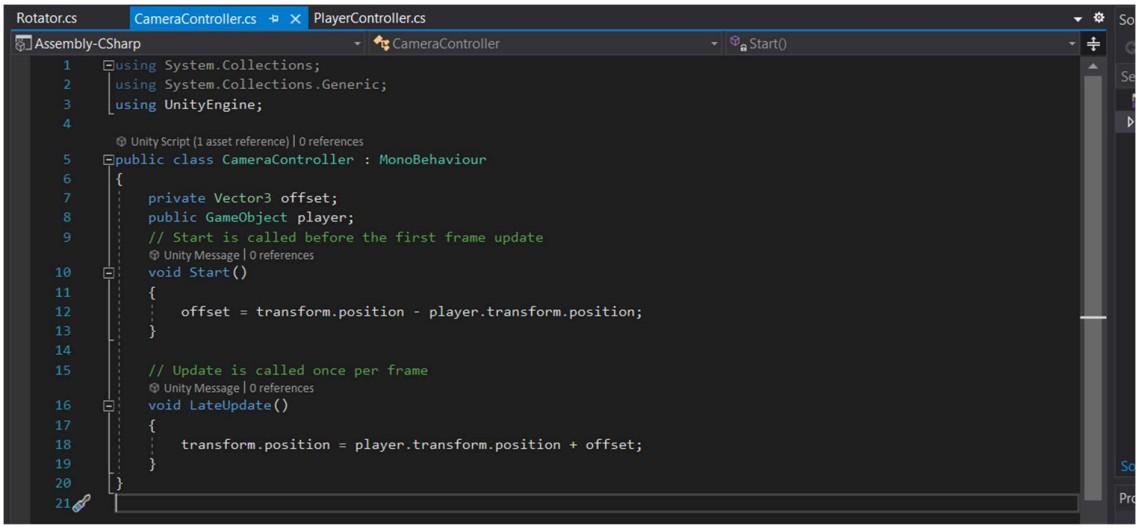
PlayerController.cs



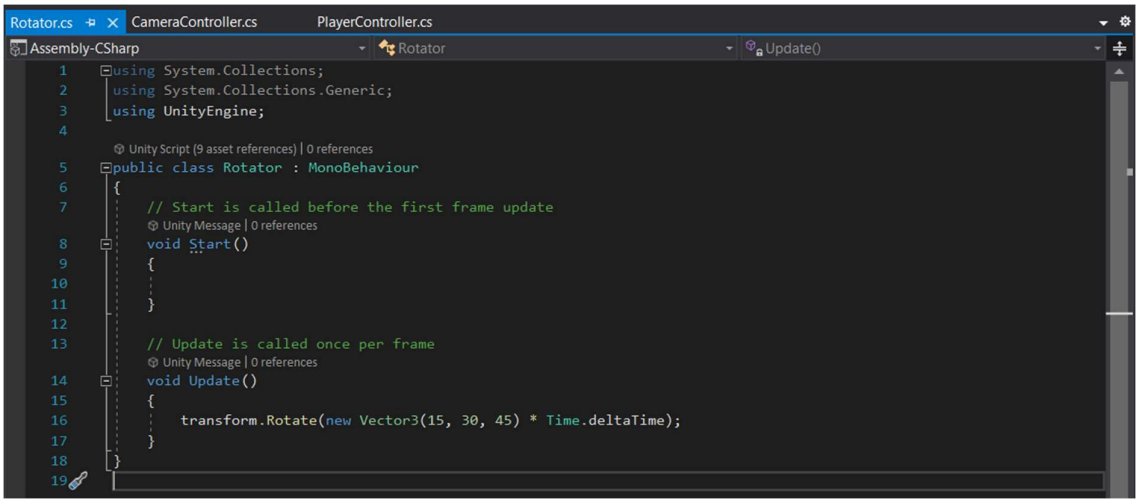
```
Rotator.cs | CameraController.cs | PlayerController.cs
Assembly-CSharp | PlayerController | speed

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

Unity Script (1 asset reference) | 0 references
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
    Unity Message | 0 references
    void Start()
    {
        count = 0;
        wintext.text = "";
        rb = GetComponent<Rigidbody>();
        SetCountText();

    }

    // Update is called once per frame
    Unity Message | 0 references
    void FixedUpdate()
    {
        float moveHorizontal = Input.GetAxis("Horizontal");
        float moveVertical = Input.GetAxis("Vertical");
        Vector3 movement = new Vector3(moveHorizontal, 0.0f, moveVertical);
        rb.AddForce(movement * speed);
    }
    Unity Message | 0 references
    void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.CompareTag("PickUp"))
        {
            other.gameObject.SetActive(false);

100 %   No issues found                                   Ln: 12   Ch: 24   SPC   CRLF
```

```
            count = count + 1;
            SetCountText();
        }
    }
    2 references
    void SetCountText()
    {
        counttext.text = "Score:" + count.ToString();
        if (count >= 8)
        {
            wintext.text = "You Win!";
        }
    }
}
```

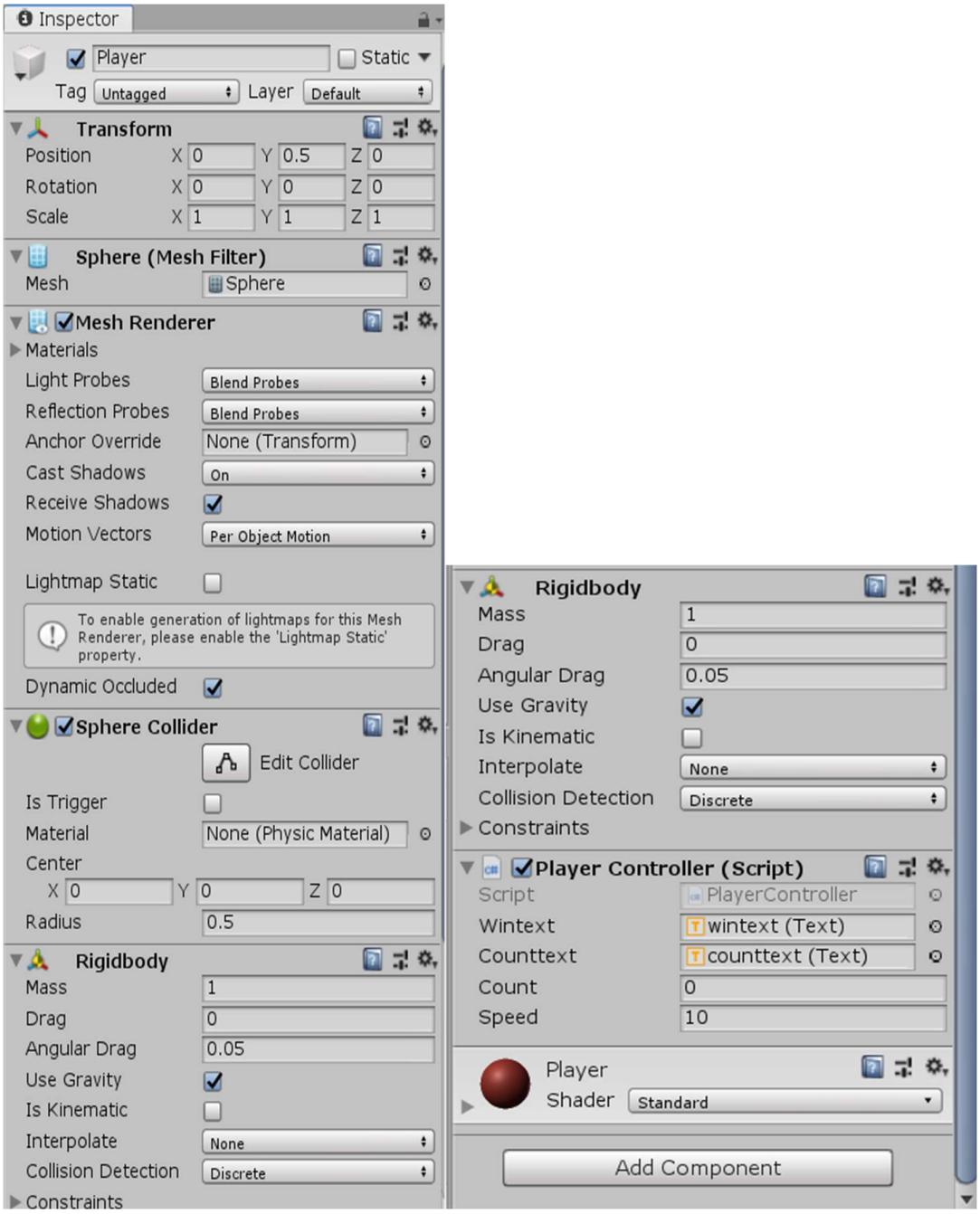
CameraController.cs



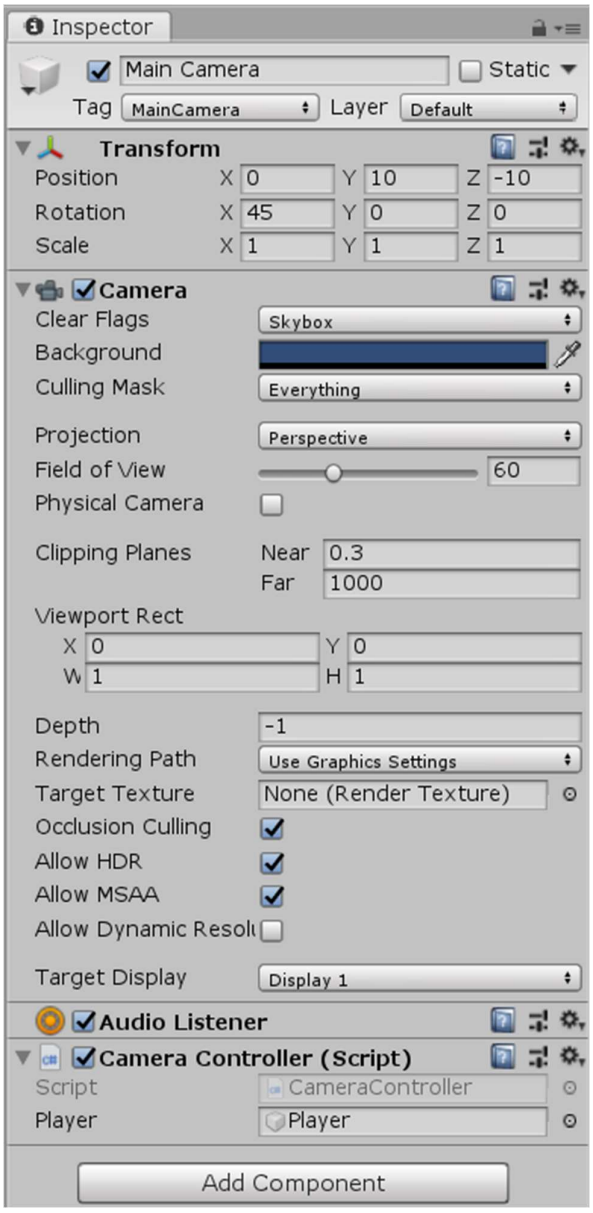
Rotator.cs



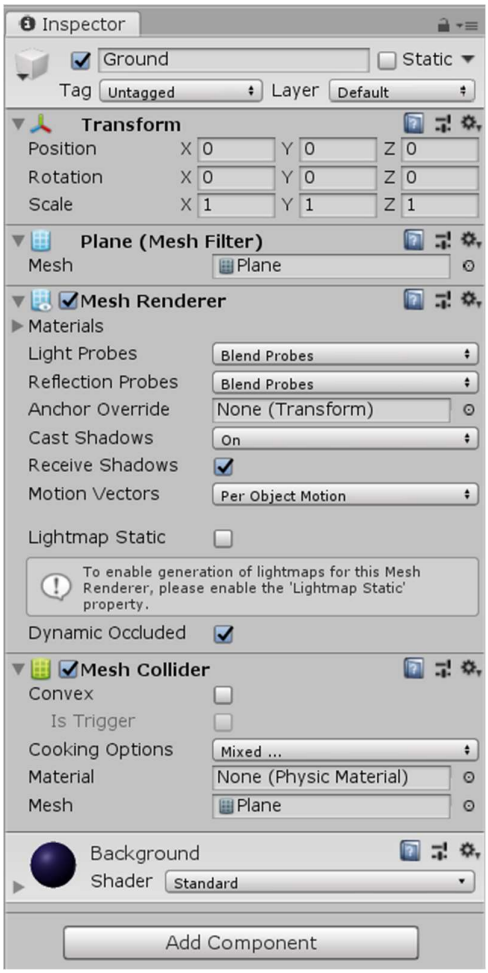
Player Inspector



Main Camera Inspector



Ground Inspector



Output: