

S.I.E.S College of Arts, Science and Commerce (Autonomous) Sion(W), Mumbai – 400 022.

CERTIFICATE

This is to certify that Mr. Kushal Ramesh Senghani Roll No. TCS2324069 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 **SIES College of Arts, Science & Commerce (Autonomous) Affiliated with Mumbai University** for the course of **T.Y. BSc. Computer Science [Semester-5]**

Subject. Name

**Game Programming**

Subject. In-Charge

**Ms. Soni Yadav**

Examination Date:

Examiner’s Signature & Date:

Head of the Department

**Prof. Manoj Singh**

College Seal

And

 Date

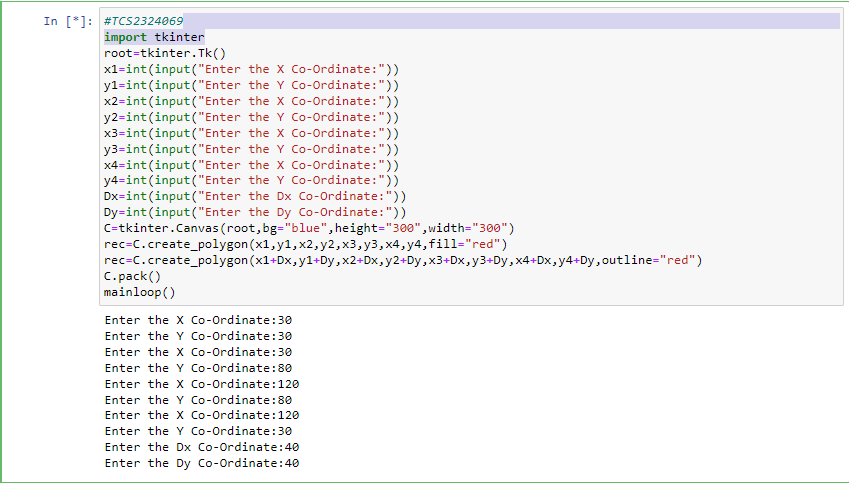
**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Practical No.** | **Topics** | **Page No.** | **Date** | **Signature** |
| 1 | Write a python program to perform translation operation on rectangle by taking initial coordinates from user | 4 | 24/07/23 |  |
| 2 | Write a python program to perform scaling operation on triangle by taking initial coordinates from user. | 5 | 24/07/23 |  |
| 3 | Write a python program to perform reflection operation on polygon by taking initial coordinates from user. | 6 | 30/07/23 |  |
| 4 | Write a python program to rotate right angle triangle by 45 degrees by taking initial coordinates from user. | 8 | 30/08/23 |  |
| 5 | Write a python program to perform shearing on rectangle in positive direction of x-axis by taking initial coordinates from user. | 10 | 14/08/23 |  |
| 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. | 12 | 21/08/23 |  |
| 7 | Implement space invader game in python using pygame module. | 13 | 21/08/23 |  |
| 8 | Implement space invader game in python using pygame module. | 14 | 24/08/23 |  |
| 9 | Implement 2D UFO game using unity | 16 | 28/08/23 |  |
| 10 | Implement 3D roll ball game using unity | 18 | 30/08/23 |  |

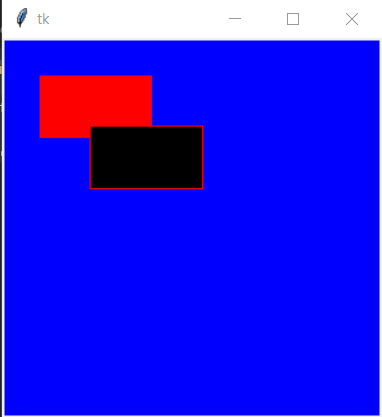
Practical No1:

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

Code:



Output:



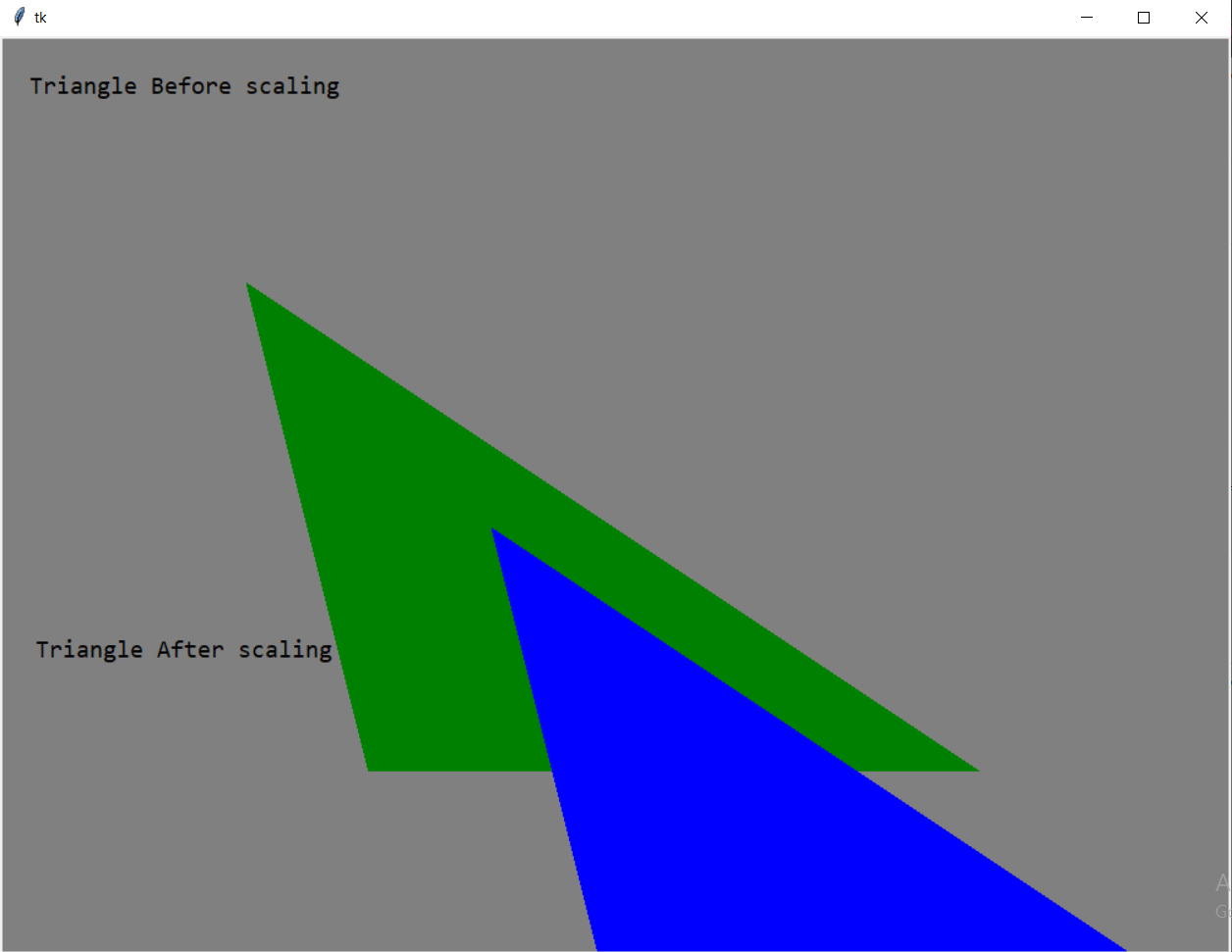
Practical No2:

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

Code:



Output:



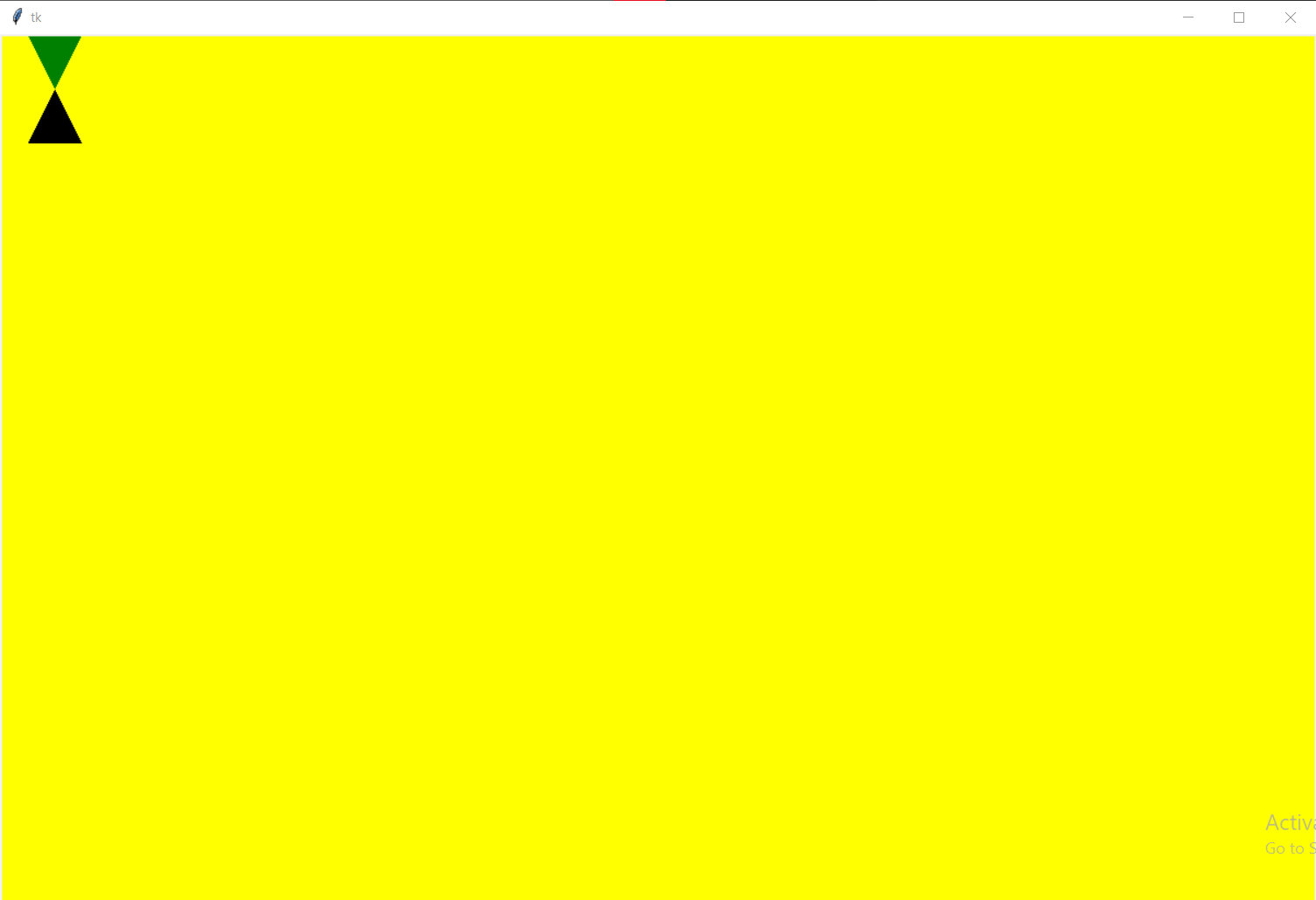
Practical No3:

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

Code:



Output:



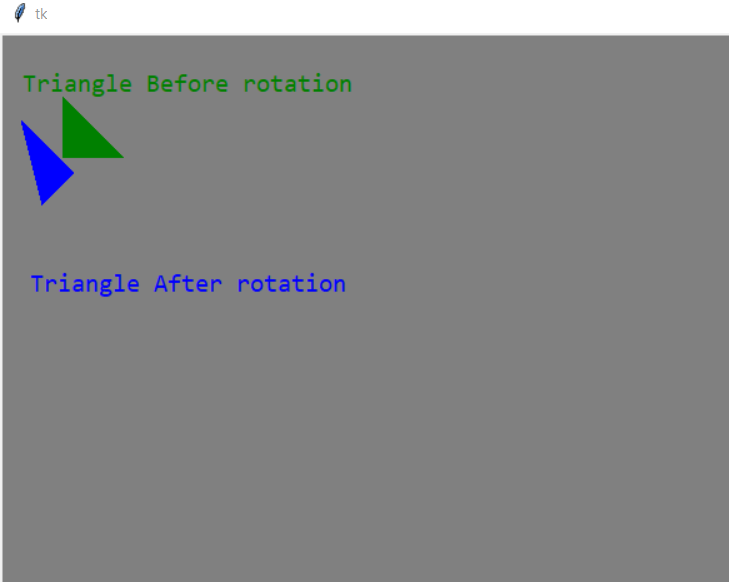
Practical No4:

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

Code:



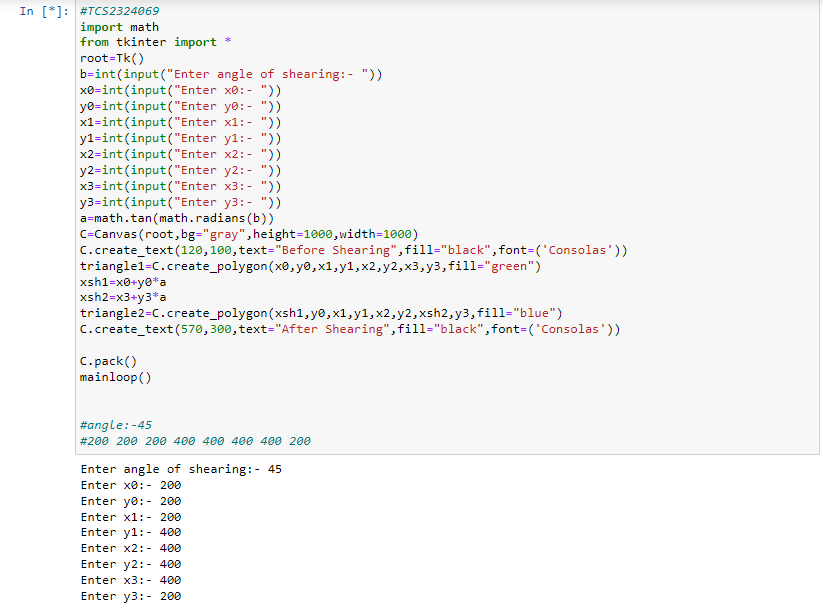
Output:



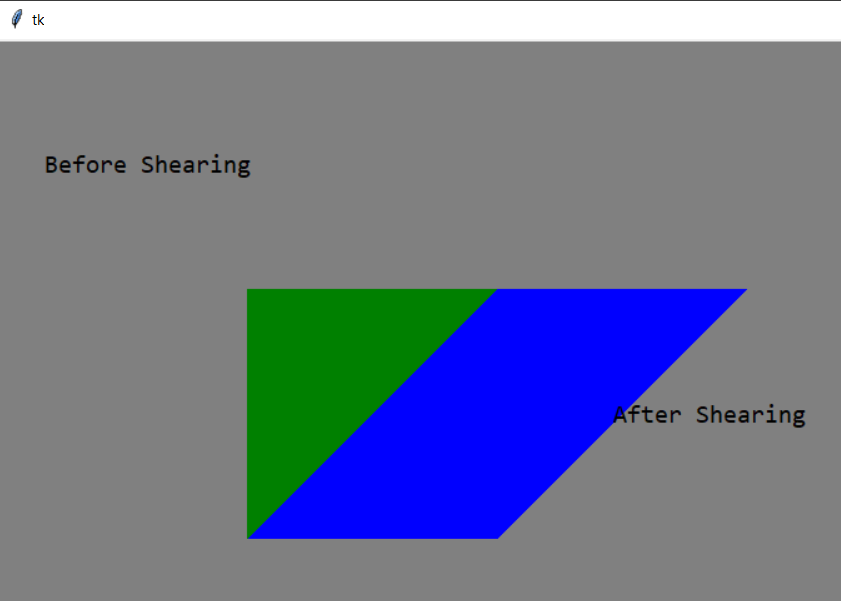
Practical No5:

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

Code:



Output:

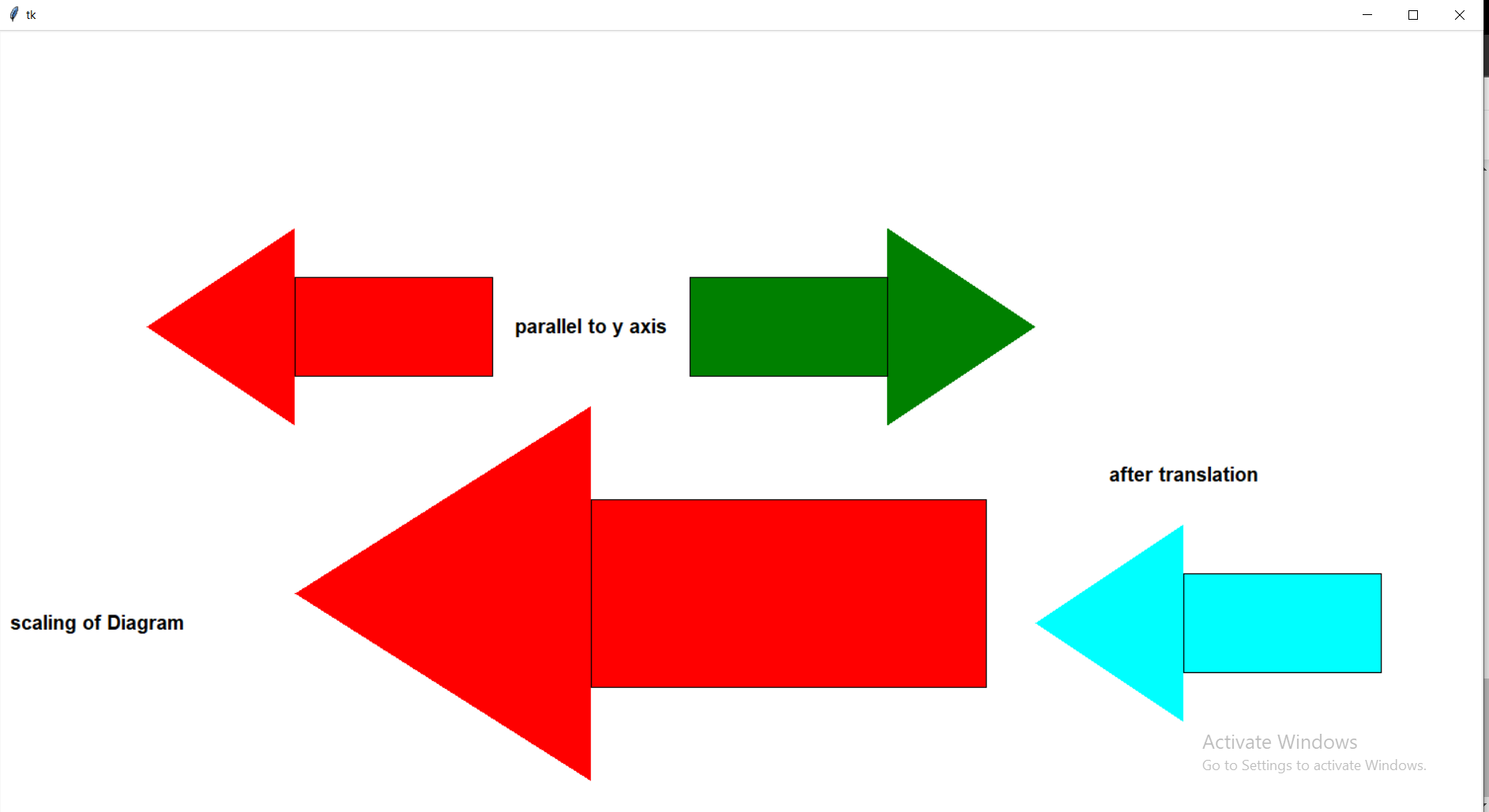


Practical No 6: Write a python program to create below shape and perform reflection about parallel to y-axis, followed by translation and scaling operation on it.

Code:

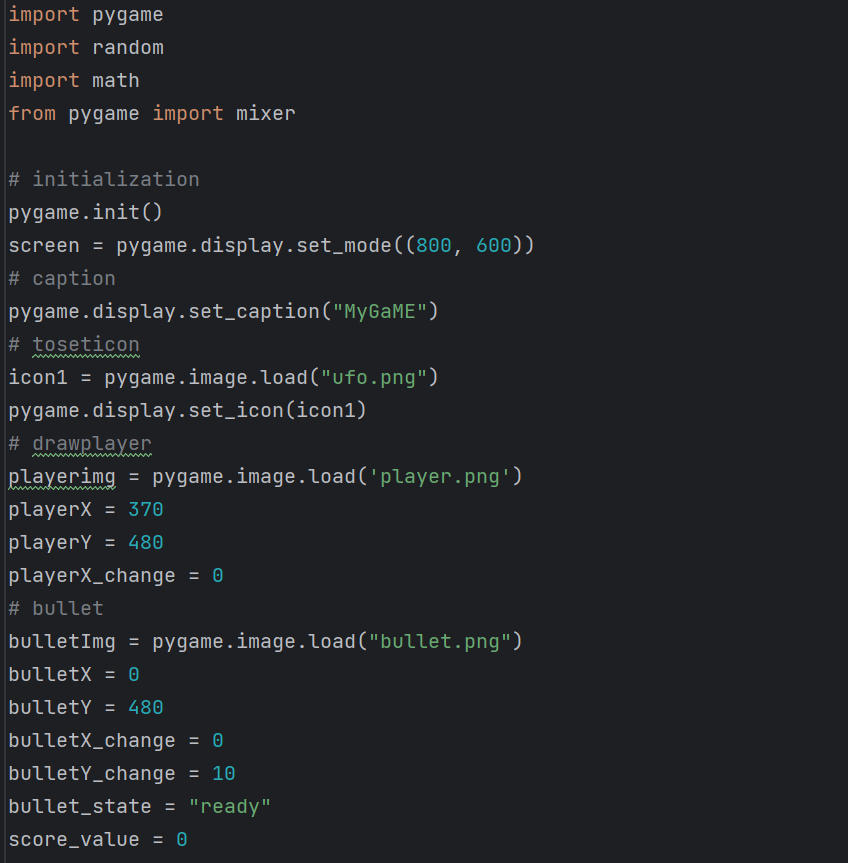


Output:

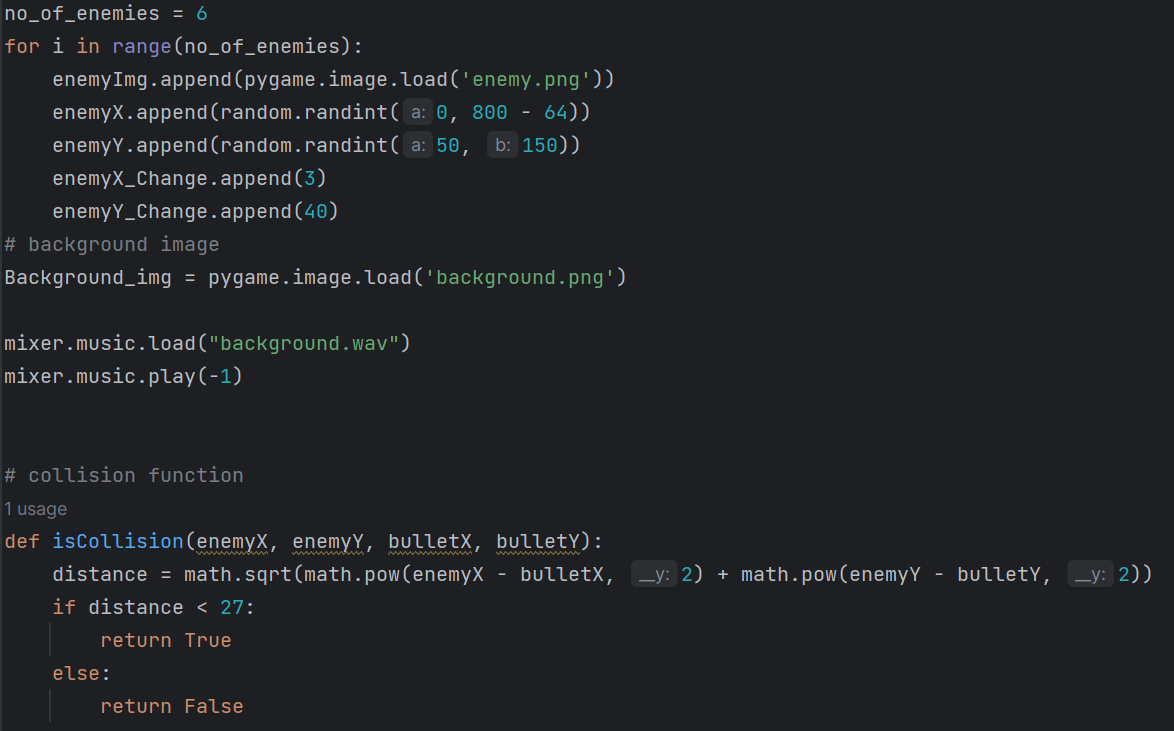


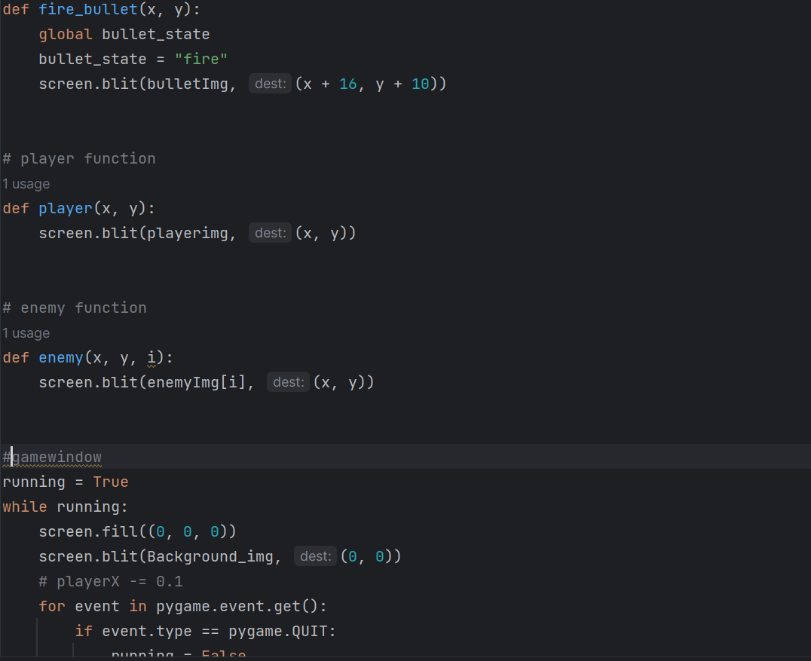
Practical No7: Implement space invader game in python using Py Game module.

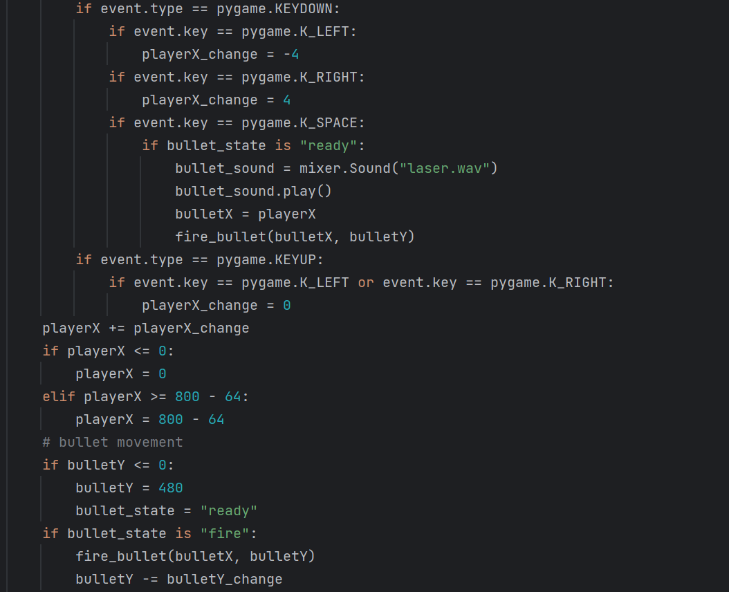
Code:

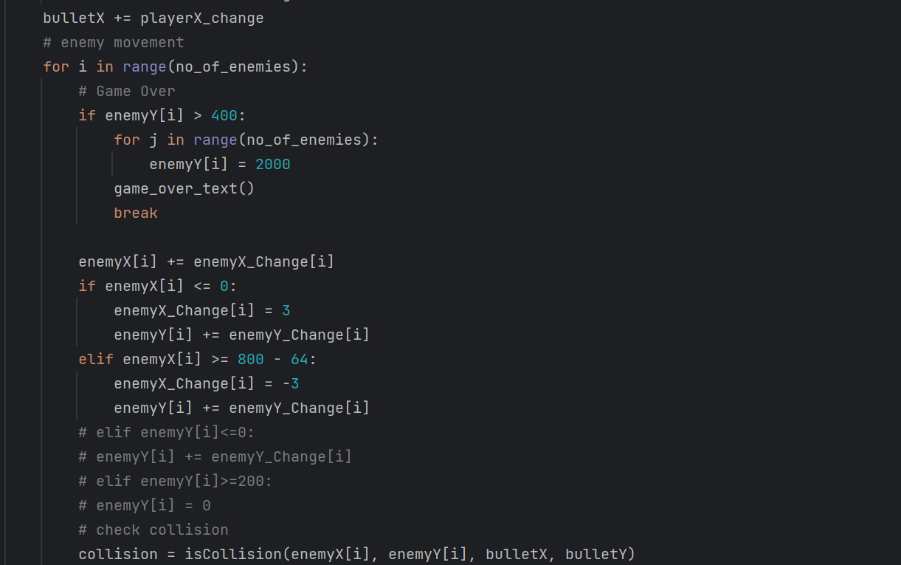


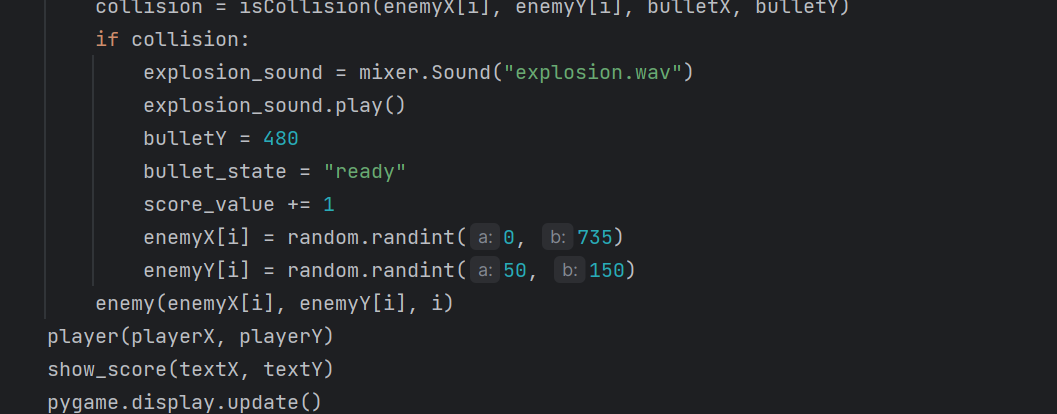




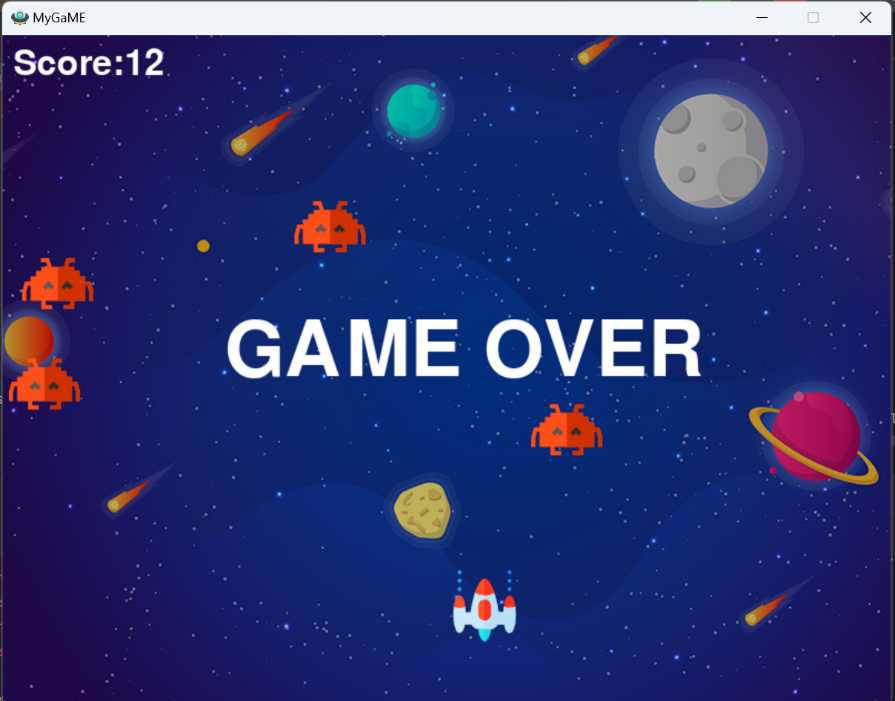








Output:



Practical No8: Implement Snake game in python using Py-game module.

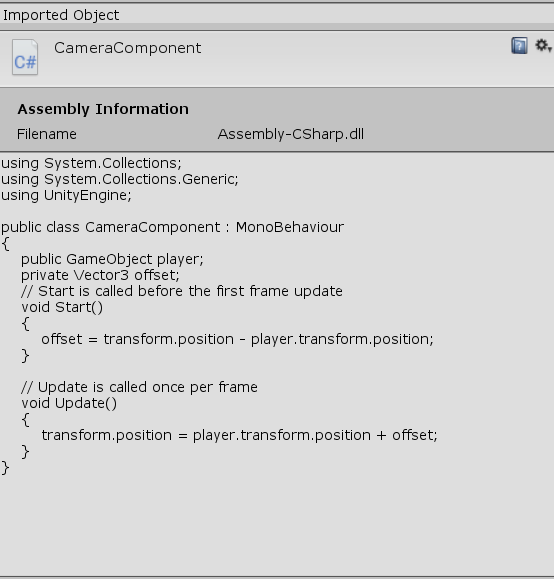
Code:   
screen\_height = 600  
  
screen = pygame.display.set\_mode((screen\_width, screen\_height))  
pygame.display.set\_caption('Snake')  
  
# define font  
font = pygame.font.SysFont(None, 40)  
  
# setup a rectangle for "Play Again" Option  
again\_rect = Rect(screen\_width // 2 - 80, screen\_height // 2, 160, 50)  
  
# define snake variables  
snake\_pos = [[int(screen\_width / 2), int(screen\_height / 2)]]  
snake\_pos.append([300, 310])  
snake\_pos.append([300, 320])  
snake\_pos.append([300, 330])  
direction = 1 # 1 is up, 2 is right, 3 is down, 4 is left  
  
# define game variables  
cell\_size = 10  
update\_snake = 0  
food = [0, 0]  
new\_food = True  
new\_piece = [0, 0]  
game\_over = False  
clicked = False  
score = 0  
  
# define colors  
bg = (255, 200, 150)  
body\_inner = (50, 175, 25)  
body\_outer = (100, 100, 200)  
food\_col = (200, 50, 50)  
blue = (0, 0, 255)  
red = (255, 0, 0)  
  
  
def draw\_screen():  
 screen.fill(bg)  
  
  
def draw\_score():  
 score\_txt = 'Score: ' + str(score)  
 score\_img = font.render(score\_txt, True, blue)  
 screen.blit(score\_img, (0, 0))  
  
  
def check\_game\_over(game\_over):  
 # first check is to see if the snake has eaten itself by checking if the head has clashed with the rest of the body  
 head\_count = 0  
 for x in snake\_pos:  
 if snake\_pos[0] == x and head\_count > 0:  
 game\_over = True  
 head\_count += 1  
  
 # second check is to see if the snake has gone out of bounds  
 if snake\_pos[0][0] < 0 or snake\_pos[0][0] > screen\_width or snake\_pos[0][1] < 0 or snake\_pos[0][1] > screen\_height:  
 game\_over = True  
  
 return game\_over  
  
  
def draw\_game\_over():  
 over\_text = "Game Over TCS2324069!"  
 over\_img = font.render(over\_text, True, blue)  
 pygame.draw.rect(screen, red, (screen\_width // 2 - 80, screen\_height // 2 - 60, 160, 50))  
 screen.blit(over\_img, (screen\_width // 2 - 80, screen\_height // 2 - 50))  
  
 again\_text = 'Play Again?'  
 again\_img = font.render(again\_text, True, blue)  
 pygame.draw.rect(screen, red, again\_rect)  
 screen.blit(again\_img, (screen\_width // 2 - 80, screen\_height // 2 + 10))  
  
  
run = True  
while run:  
  
 draw\_screen()  
 draw\_score()  
  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 run = False  
 if event.type == pygame.KEYDOWN:  
 if event.key == pygame.K\_UP and direction != 3:  
 direction = 1  
 if event.key == pygame.K\_RIGHT and direction != 4:  
 direction = 2  
 if event.key == pygame.K\_DOWN and direction != 1:  
 direction = 3  
 if event.key == pygame.K\_LEFT and direction != 2:  
 direction = 4  
  
 # create food  
 if new\_food == True:  
 new\_food = False  
 food[0] = cell\_size \* random.randint(0, (screen\_width / cell\_size) - 1)  
 food[1] = cell\_size \* random.randint(0, (screen\_height / cell\_size) - 1)  
  
 # draw food  
 pygame.draw.rect(screen, food\_col, (food[0], food[1], cell\_size, cell\_size))  
  
 # check if food has been eaten  
 if snake\_pos[0] == food:  
 new\_food = True  
 # create a new piece at the last point of the snake's tail  
 new\_piece = list(snake\_pos[-1])  
 # add an extra piece to the snake  
 if direction == 1:  
 new\_piece[1] += cell\_size  
 # heading down  
 if direction == 3:  
 new\_piece[1] -= cell\_size  
 # heading right  
 if direction == 2:  
 new\_piece[0] -= cell\_size  
 # heading left  
 if direction == 4:  
 new\_piece[0] += cell\_size  
  
 # attach new piece to the end of the snake  
 snake\_pos.append(new\_piece)  
  
 # increase score  
 score += 1  
  
 if game\_over == False:  
 # update snake  
 if update\_snake > 99:  
 update\_snake = 0  
 # first shift the positions of each snake piece back.  
 snake\_pos = snake\_pos[-1:] + snake\_pos[:-1]  
 # now update the position of the head based on direction  
 # heading up  
 if direction == 1:  
 snake\_pos[0][0] = snake\_pos[1][0]  
 snake\_pos[0][1] = snake\_pos[1][1] - cell\_size  
 # heading down  
 if direction == 3:  
 snake\_pos[0][0] = snake\_pos[1][0]  
 snake\_pos[0][1] = snake\_pos[1][1] + cell\_size  
 # heading right  
 if direction == 2:  
 snake\_pos[0][1] = snake\_pos[1][1]  
 snake\_pos[0][0] = snake\_pos[1][0] + cell\_size  
 # heading left  
 if direction == 4:  
 snake\_pos[0][1] = snake\_pos[1][1]  
 snake\_pos[0][0] = snake\_pos[1][0] - cell\_size  
 game\_over = check\_game\_over(game\_over)  
  
 if game\_over == True:  
 draw\_game\_over()  
 if event.type == pygame.MOUSEBUTTONDOWN and clicked == False:  
 clicked = True  
 if event.type == pygame.MOUSEBUTTONUP and clicked == True:  
 clicked = False  
 # reset variables  
 game\_over = False  
 update\_snake = 0  
 food = [0, 0]  
 new\_food = True  
 new\_piece = [0, 0]  
 # define snake variables  
 snake\_pos = [[int(screen\_width / 2), int(screen\_height / 2)]]  
 snake\_pos.append([300, 310])  
 snake\_pos.append([300, 320])  
 snake\_pos.append([300, 330])  
 direction = 1 # 1 is up, 2 is right, 3 is down, 4 is left  
 score = 0  
  
 head = 1  
 for x in snake\_pos:  
  
 if head == 0:  
 pygame.draw.rect(screen, body\_outer, (x[0], x[1], cell\_size, cell\_size))  
 pygame.draw.rect(screen, body\_inner, (x[0] + 1, x[1] + 1, cell\_size - 2, cell\_size - 2))  
 if head == 1:  
 pygame.draw.rect(screen, body\_outer, (x[0], x[1], cell\_size, cell\_size))  
 pygame.draw.rect(screen, (255, 0, 0), (x[0] + 1, x[1] + 1, cell\_size - 2, cell\_size - 2))  
 head = 0  
  
 pygame.display.update()  
  
 update\_snake += 1  
  
pygame.quit()

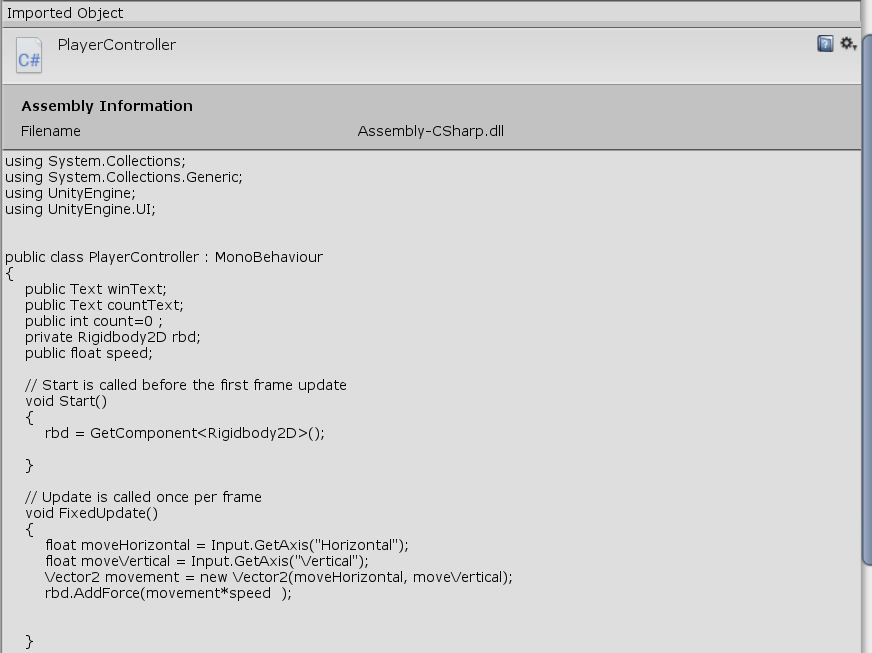
Output:



Practical No9: Implement 2D UFO game using unity.

Code

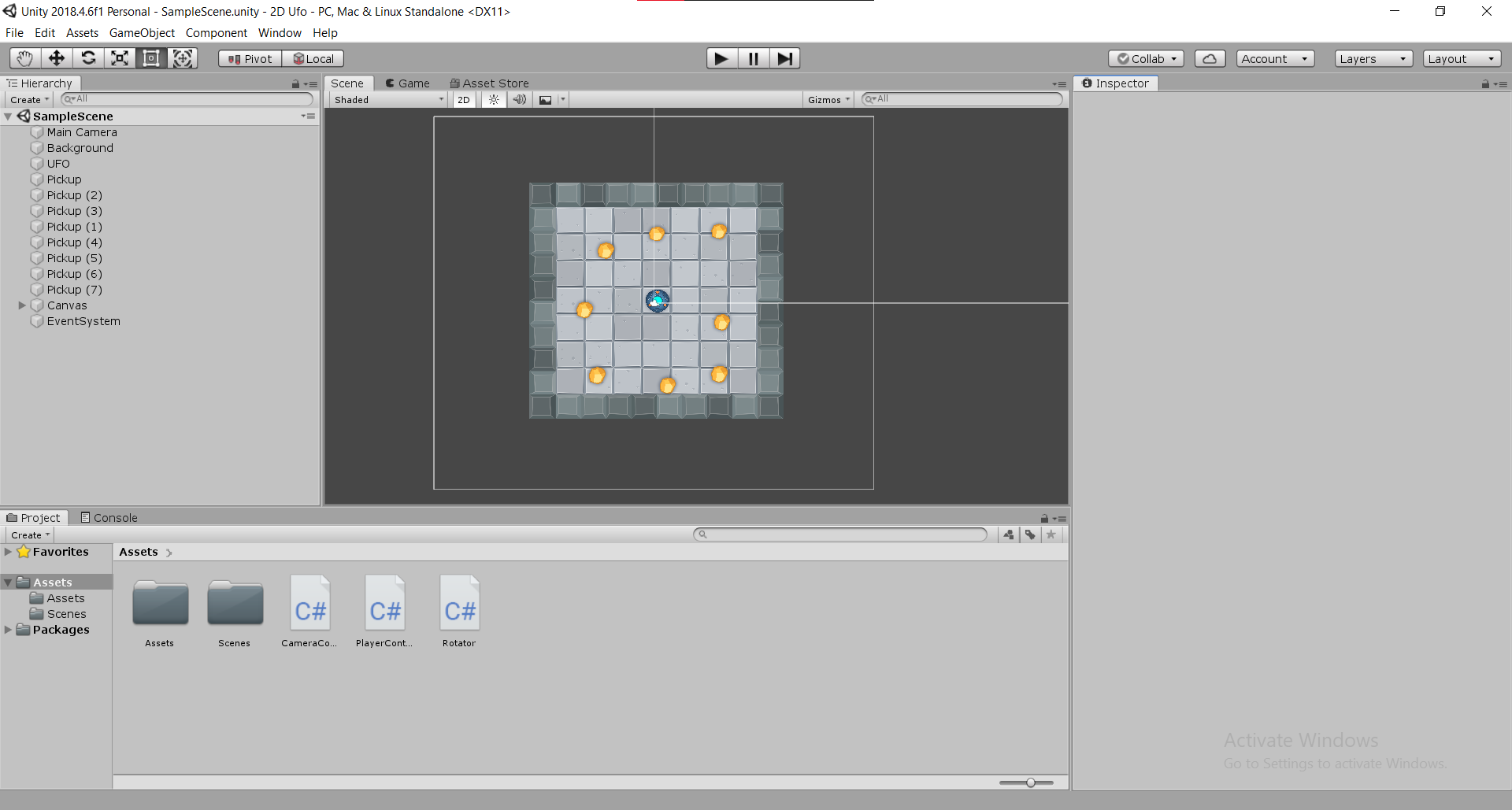


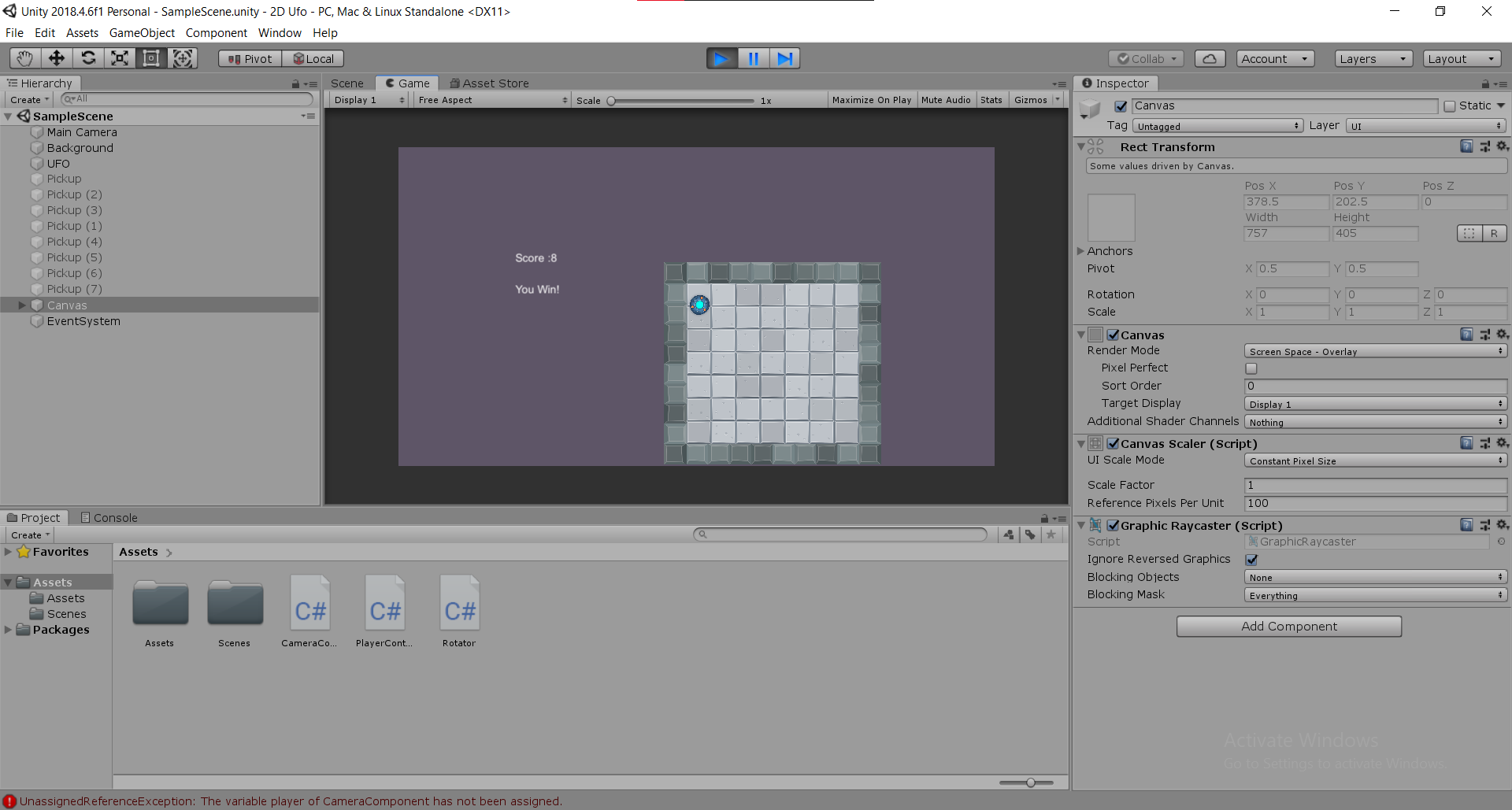






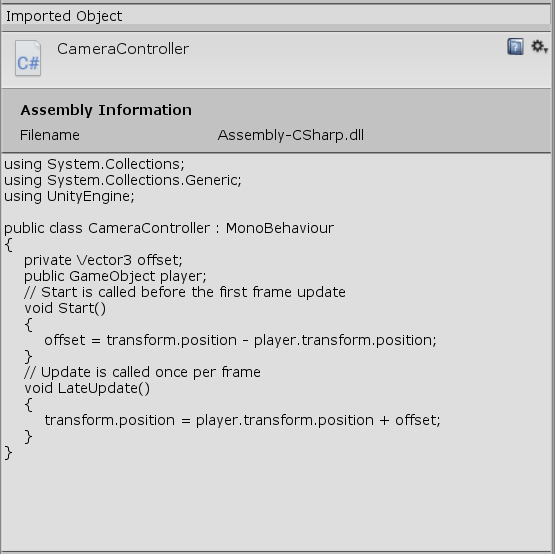
Output:

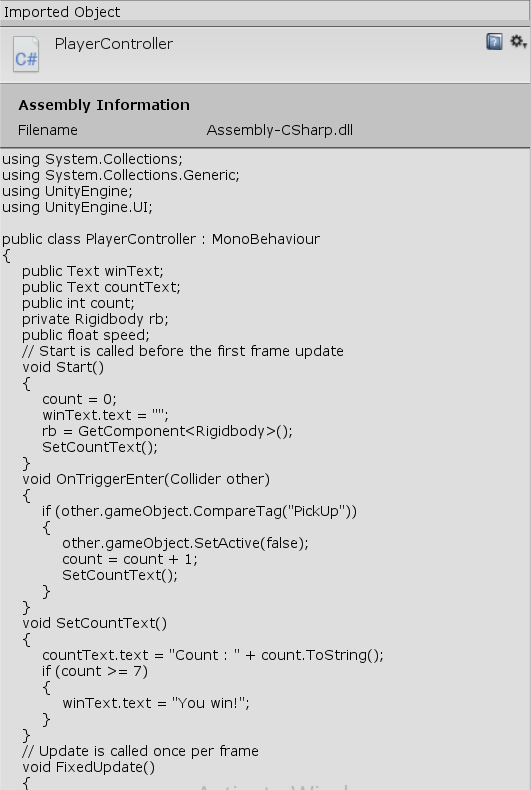


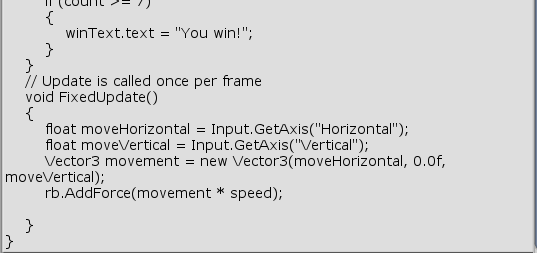


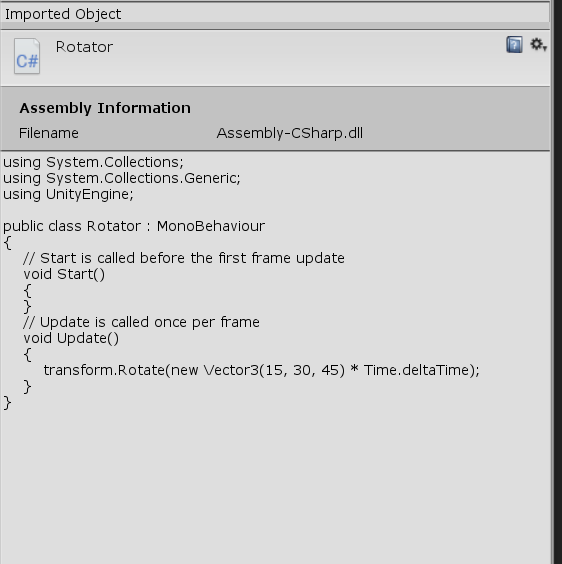
Practical No10: Implement 3D roll ball game using unity.

Code:









Output:

