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

**Code:**

# 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()

**Aim:-pygame for adding image.**

import pygame

import sys

# Initialize Pygame

pygame.init()

# Set up the display

width, height = 450,500

screen = pygame.display.set\_mode((width, height))

pygame.display.set\_caption('Display Image Example')

# Load the image

image = pygame.image.load(r'C:/Users/hp/Downloads/sun.jpeg')

# Main loop

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

# Fill the screen with a color (optional)

screen.fill((255, 255, 255)) # White background

# Blit (draw) the image onto the screen

screen.blit(image, (0, 0)) # Draw image at (0, 0)

# Update the display

pygame.display.flip()

**Aim: Pygame for key pressing and key release.**

**Code:**

import pygame

import sys

# Initialize Pygame

pygame.init()

# Set up the display

width, height = 800, 600

screen = pygame.display.set\_mode((width, height))

pygame.display.set\_caption('Key Press and Release Example')

# Define a rectangle object

rect\_x, rect\_y = width // 2, height // 2

rect\_width, rect\_height = 50, 50

rect\_color = (0, 128, 255) # Blue color

speed = 5 # Speed of the rectangle movement

# Main loop

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

# Key down event

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_UP:

print("Up key pressed")

elif event.key == pygame.K\_DOWN:

print("Down key pressed")

elif event.key == pygame.K\_LEFT:

print("Left key pressed")

elif event.key == pygame.K\_RIGHT:

print("Right key pressed")

# Key up event

if event.type == pygame.KEYUP:

if event.key == pygame.K\_UP:

print("Up key released")

elif event.key == pygame.K\_DOWN:

print("Down key released")

elif event.key == pygame.K\_LEFT:

print("Left key released")

elif event.key == pygame.K\_RIGHT:

print("Right key released")

# Handle key states

keys = pygame.key.get\_pressed()

if keys[pygame.K\_UP]:

rect\_y -= speed

if keys[pygame.K\_DOWN]:

rect\_y += speed

if keys[pygame.K\_LEFT]:

rect\_x -= speed

if keys[pygame.K\_RIGHT]:

rect\_x += speed

# Fill the screen with a color

screen.fill((255, 255, 255)) # White background

# Draw the rectangle

pygame.draw.rect(screen, rect\_color, (rect\_x, rect\_y, rect\_width, rect\_height))

# Update the display

pygame.display.flip()

**Aim: Pygame to display different shape.**

**Code:**

import pygame

import sys

# Initialize Pygame

pygame.init()

# Set up the display

width, height = 800, 600

screen = pygame.display.set\_mode((width, height))

pygame.display.set\_caption('Drawing Different Shapes')

# Define colors

white = (255, 255, 255)

black = (0, 0, 0)

blue = (0, 0, 255)

red = (255, 0, 0)

green = (0, 255, 0)

yellow = (255, 255, 0)

# Main loop

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

# Fill the screen with white background

screen.fill(white)

# Draw a rectangle (position: x=100, y=100, width=200, height=100)

pygame.draw.rect(screen, blue, (100, 100, 200, 100))

# Draw a circle (center: x=400, y=300, radius=50)

pygame.draw.circle(screen, red, (400, 300), 50)

# Draw an ellipse (bounding rect: x=500, y=200, width=150, height=80)

pygame.draw.ellipse(screen, green, (500, 200, 150, 80))

# Draw a line (start: x=50, y=500, end: x=750, y=500, thickness=5)

pygame.draw.line(screen, black, (50, 500), (750, 500), 5)

# Draw a polygon (a triangle in this case)

pygame.draw.polygon(screen, yellow, [(600, 150), (700, 100), (750, 200)])

# Update the display

pygame.display.flip()

**Aim:- Learn Basic Game Designing Techniques with pygame.**

import pygame

from math import pi

# Initialize Pygame

pygame.init()

# Set up display

width, height = 800, 800

screen = pygame.display.set\_mode((width, height))

pygame.display.set\_caption("Pygame Example")

# Load image

image = pygame.image.load(r"C:\Users\hp\Desktop\chess.jpeg")

# Main loop

running = True

clock = pygame.time.Clock()

while running:

# Event handling

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

if event.type == pygame.KEYDOWN:

key\_name = pygame.key.name(event.key).upper()

print(f'"{key\_name}" key pressed')

elif event.type == pygame.KEYUP:

key\_name = pygame.key.name(event.key).upper()

print(f'"{key\_name}" key released')

# Clear the screen

screen.fill((255, 255, 255))

# Draw shapes

pygame.draw.line(screen, (0, 255, 0), [0, 0], [50, 30], 5)

pygame.draw.lines(screen, (0, 0, 0), False, [[0, 80], [50, 90], [200, 80], [220, 30]], 5)

pygame.draw.rect(screen, (0, 0, 0), [75, 10, 50, 20], 2)

pygame.draw.rect(screen, (0, 0, 0), [150, 10, 50, 20])

pygame.draw.ellipse(screen, (255, 0, 0), [225, 10, 50, 20], 2)

pygame.draw.ellipse(screen, (255, 0, 0), [300, 10, 50, 20])

pygame.draw.polygon(screen, (0, 0, 0), [[100, 100], [0, 200], [200, 200]], 5)

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

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

# Display the image

screen.blit(image, (0, 0))

# Update the display

pygame.display.flip()

# Limit frames per second

clock.tick(60)

# Quit Pygame

pygame.quit()

**Aim:- Develop Snake Game using pygame**

**Code:**

import pygame

import random

# Initialize Pygame

pygame.init()

# Colors

white = (255, 255, 255)

yellow = (255, 255, 102)

black = (0, 0, 0)

red = (213, 50, 80)

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]) # Food color

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()

**Aim:- Create 2D Target Shooting Game.**

**Code:**

import pygame

import random

# Initialize Pygame

pygame.init()

# Screen dimensions

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 properties

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] and character.left > 0:

character.x -= character\_speed

if keys[pygame.K\_RIGHT] and character.right < width:

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

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()

**Aim:- Creating 2D Infinite Scrolling Background**

**Code:**

import pygame as py

import math

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))

bg = py.image.load("earth.jpg").convert()

# DEFINING MAIN VARIABLES IN SCROLLING

scroll = 0

# HERE 1 IS THE CONSTATNT FOR REMOVING BUFFERING

tiles = math.ceil(FrameWidth /bg.get\_width()) + 1

while 1:

clock.tick(50)

# 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 -= 6

# RESET THE SCROLL FRAME

if abs(scroll) > bg.get\_width():

scroll = 0

# CLOSING THE FRAME OF SCROLLING

for event in py.event.get():

if event.type == py.QUIT:

quit()

py.display.update()

py.quit()