2) Basic game design technique

```
import pygame
pygame.init()
window = pygame.display.set_mode((600, 600))
window.fill((255, 255, 255))
pygame.draw.rect(window, (0, 0, 255), [100, 100, 400, 100], 0)
pygame.draw.circle(window, (0, 255, 0), [300, 300], 170, 0)
pygame.draw.polygon(window, (255, 0, 0), [[300, 300], [100, 400], [100, 300]])
pygame.display.update()
```

3) Snake game

```
import pygame
import time
import random

# Game settings
snake_speed = 20
window_x = 720
window_y = 480

# Colors
black = pygame.Color(0, 0, 0)
white = pygame.Color(255, 255, 255)
red = pygame.Color(255, 0, 0)
green = pygame.Color(0, 255, 0)
blue = pygame.Color(0, 0, 255)

# Initialize game
pygame.init()
pygame.display.set_caption("Snake Game")
```

```
game_window = pygame.display.set_mode((window_x, window_y))
fps = pygame.time.Clock()

# Snake settings
snake_position = [100, 50]
snake_body = [[100, 50], [90, 50], [80, 50], [70, 50]]

# Fruit
fruit_position = [random.randrange(1, (window_x // 10)) * 10,
                  random.randrange(1, (window_y // 10)) * 10]
fruit_spawn = True

# Direction control
direction = 'RIGHT'
change_to = direction

# Score
score = 0

# Display score
def show_score(choice, color, font, size):
    score_font = pygame.font.SysFont(font, size)
    score_surface = score_font.render('Score: ' + str(score), True, color)
    score_rect = score_surface.get_rect()
    if choice == 1:
        score_rect.topleft = (10, 10)
    else:
        score_rect.midtop = (window_x / 2, window_y / 4)
    game_window.blit(score_surface, score_rect)
```

```
# Game over function
def game_over():
    my_font = pygame.font.SysFont('times new roman', 50)
    game_over_surface = my_font.render('Your Score is: ' + str(score), True, red)
    game_over_rect = game_over_surface.get_rect()
    game_over_rect.midtop = (window_x / 2, window_y / 4)
    game_window.blit(game_over_surface, game_over_rect)
    pygame.display.flip()
    time.sleep(2)
    pygame.quit()
    quit()

# Main game loop
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            quit()
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP and direction != 'DOWN':
                change_to = 'UP'
            elif event.key == pygame.K_DOWN and direction != 'UP':
                change_to = 'DOWN'
            elif event.key == pygame.K_LEFT and direction != 'RIGHT':
                change_to = 'LEFT'
            elif event.key == pygame.K_RIGHT and direction != 'LEFT':
                change_to = 'RIGHT'

    # Apply direction change
    direction = change_to
```

```
# Move snake
if direction == 'UP':
    snake_position[1] -= 10
if direction == 'DOWN':
    snake_position[1] += 10
if direction == 'LEFT':
    snake_position[0] -= 10
if direction == 'RIGHT':
    snake_position[0] += 10

# Snake body growing mechanism
snake_body.insert(0, list(snake_position))
if snake_position == fruit_position:
    score += 10
    fruit_spawn = False
else:
    snake_body.pop()

# Spawn fruit
if not fruit_spawn:
    fruit_position = [random.randrange(1, (window_x // 10)) * 10,
                      random.randrange(1, (window_y // 10)) * 10]
    fruit_spawn = True

# Background
game_window.fill(black)

# Draw snake
for pos in snake_body:
```

```
        pygame.draw.rect(game_window, green, pygame.Rect(pos[0], pos[1], 10, 10))

    # Draw fruit
    pygame.draw.rect(game_window, white, pygame.Rect(fruit_position[0], fruit_position[1],
10, 10))

    # Game Over conditions
    if snake_position[0] < 0 or snake_position[0] > window_x - 10:
        game_over()
    if snake_position[1] < 0 or snake_position[1] > window_y - 10:
        game_over()

    # Self collision
    for block in snake_body[1:]:
        if snake_position[0] == block[0] and snake_position[1] == block[1]:
            game_over()

    # Display score and update screen
    show_score(1, white, 'times new roman', 20)
    pygame.display.update()
    fps.tick(snake_speed)
```

4) 2D game design

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

```
# Initialize pygame
pygame.init()

# Screen dimensions
screen_width = 800
screen_height = 600
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("2D Target Shooting Game")

# Background
background = pygame.image.load('Space.jpg')

# Background music
mixer.music.load("music.mp3")
mixer.music.play(-1)

# Score
score_val = 0
font = pygame.font.Font('freesansbold.ttf', 32)
scoreX = 10
scoreY = 10

# Game Over font
game_over_font = pygame.font.Font('freesansbold.ttf', 64)

# Clock
clock = pygame.time.Clock()

# Player
```

```
playerImage = pygame.image.load("NRG.png")
playerImage = pygame.transform.scale(playerImage, (75, 75))
player_X = 370
player_Y = 500
player_Xchange = 0
player_Ychange = 0
player_speed = 3

# Invaders
invaderImage = []
invader_X = []
invader_Y = []
invader_Xchange = []
invader_Ychange = []
no_of_invaders = 6

for i in range(no_of_invaders):
    img = pygame.image.load("enemy.png")
    img = pygame.transform.scale(img, (65, 65))
    invaderImage.append(img)
    invader_X.append(random.randint(64, 736))
    invader_Y.append(random.randint(30, 200))
    invader_Xchange.append(1.5)
    invader_Ychange.append(40)

# Bullets (multiple)
bulletImage = pygame.image.load("lazer.png")
bulletImage = pygame.transform.scale(bulletImage, (30, 45))
bullets = []  # List of bullets; each bullet is a dict with x and y
```

```
# Functions
def show_score(x, y):
    score = font.render("Score: " + str(score_val), True, (255, 255, 255))
    screen.blit(score, (x, y))

def game_over():
    over_text = game_over_font.render("GAME OVER", True, (255, 255, 255))
    screen.blit(over_text, (200, 250))

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

def invader(x, y, i):
    screen.blit(invaderImage[i], (x, y))

def fire_bullet(x, y):
    bullets.append({'x': x + 22, 'y': y})  # Center the bullet
    bullet_sound = mixer.Sound("lazermp3.mp3")
    bullet_sound.play()

def isCollision(inv_x, inv_y, bullet_x, bullet_y):
    distance = math.sqrt((math.pow(inv_x - bullet_x, 2)) + (math.pow(inv_y - bullet_y, 2)))
    return distance < 40

# Game Loop
running = True
while running:
    screen.fill((0, 0, 0))
    screen.blit(background, (0, 0))
```

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        running = False

    # Movement
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_LEFT:
            player_Xchange = -player_speed
        if event.key == pygame.K_RIGHT:
            player_Xchange = player_speed
        if event.key == pygame.K_UP:
            player_Ychange = -player_speed
        if event.key == pygame.K_DOWN:
            player_Ychange = player_speed
        if event.key == pygame.K_SPACE:
            fire_bullet(player_X, player_Y)

    if event.type == pygame.KEYUP:
        if event.key in (pygame.K_LEFT, pygame.K_RIGHT):
            player_Xchange = 0
        if event.key in (pygame.K_UP, pygame.K_DOWN):
            player_Ychange = 0

    # Mouse click on NRG
    if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
        mouse_x, mouse_y = event.pos
        player_rect = pygame.Rect(player_X, player_Y, 75, 75)
        if player_rect.collidepoint(mouse_x, mouse_y):
            fire_bullet(player_X, player_Y)
```

```
# Update player position
player_X += player_Xchange
player_Y += player_Ychange

# Keep player inside screen bounds
player_X = max(0, min(player_X, screen_width - 75))
player_Y = max(0, min(player_Y, screen_height - 75))

# Invader Movement
for i in range(no_of_invaders):
    if invader_Y[i] > 440:
        for j in range(no_of_invaders):
            invader_Y[j] = 2000
        explosion = mixer.Sound("explosion.mp3")
        explosion.play()
        game_over()
        break

    invader_X[i] += invader_Xchange[i]
    if invader_X[i] <= 0 or invader_X[i] >= 736:
        invader_Xchange[i] *= -1
        invader_Y[i] += invader_Ychange[i]

    # Check collision with all bullets
    for bullet in bullets:
        if isCollision(invader_X[i], invader_Y[i], bullet['x'], bullet['y']):
            explosion = mixer.Sound("explosion.mp3")
            explosion.play()
            bullets.remove(bullet)
            score_val += 1
```

```
            invader_X[i] = random.randint(64, 736)
            invader_Y[i] = random.randint(30, 200)
            break

        invader(invader_X[i], invader_Y[i], i)

    # Move and draw bullets
    for bullet in bullets[:]:
        bullet['y'] -= 5
        screen.blit(bulletImage, (bullet['x'], bullet['y']))
        if bullet['y'] < 0:
            bullets.remove(bullet)

    player(player_X, player_Y)
    show_score(scoreX, scoreY)
    pygame.display.update()
    clock.tick(60)
```

5) 2D Infinite Scrolling Background

```
import pygame as py
import math

# Initialize pygame
py.init()
clock = py.time.Clock()

# Screen dimensions
```

```
FrameHeight = 184
FrameWidth = 600
py.display.set_caption("Infinite Scrolling in Pygame")
screen = py.display.set_mode((FrameWidth, FrameHeight))

# Load background image
bg = py.image.load("img.jpg").convert()
bg_width = bg.get_width()

# Scroll variables
scroll = 0
tiles = math.ceil(FrameWidth / bg_width) + 1

# Main game loop
running = True
while running:
    clock.tick(33)

    # Draw background images side by side
    for i in range(tiles):
        screen.blit(bg, (i * bg_width + scroll, 0))

    # Scroll background
    scroll -= 6

    # Reset scroll position
    if abs(scroll) > bg_width:
        scroll = 0

    # Handle events
```

```
    for event in py.event.get():
        if event.type == py.QUIT:
            running = False

    py.display.update()

py.quit()
```