1. WAP to draw a white rectangle on a black background

python

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((0, 0, 0))
pygame.draw.rect(screen, (255, 255, 255), (100, 100, 200, 100))
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

2. WAP to draw a lineloop on a white background

python

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((255, 255, 255))
points = [(100, 100), (200, 200), (300, 100), (100, 100)]
pygame.draw.lines(screen, (0, 0, 0), True, points, 2)
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

3. WAP to draw a polygon on a white background

python

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((255, 255, 255))
points = [(100, 100), (200, 50), (300, 100), (250, 200), (150, 200)]
pygame.draw.polygon(screen, (0, 0, 0), points)
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
```

```
            pygame.quit()
            exit()
```

4. WAP to draw a triangleStrip on a blue background

python

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((0, 0, 255))
points = [(100, 100), (150, 200), (200, 100), (250, 200)]
pygame.draw.polygon(screen, (255, 255, 255), points)
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

5. WAP to draw a point located at [100, 100, -25]

python

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((0, 0, 0))
pygame.draw.circle(screen, (255, 255, 255), (100, 100), 2)
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

6. WAP to draw a triangle with specified points

python

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((255, 255, 255))
points = [(100, 100), (200, 100), (150, 200)]
```

```
pygame.draw.polygon(screen, (0, 0, 0), points)
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

7. WAP to draw a square with specified points

python

```
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((255, 255, 255))
pygame.draw.rect(screen, (0, 0, 0), (100, 100, 100, 100))
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

8. WAP to display hello on white background

python

```
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((255, 255, 255))
font = pygame.font.SysFont(None, 48)
text = font.render("Hello", True, (0, 0, 0))
screen.blit(text, (200, 200))
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

9. WAP to draw a moving square from left to right

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
x = 0
clock = pygame.time.Clock()
while True:
    screen.fill((255, 255, 255))
    pygame.draw.rect(screen, (0, 0, 0), (x, 200, 50, 50))
    x = (x + 2) % 500
    pygame.display.flip()
    clock.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

10. WAP to draw a moving circle from left to right

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
x = 0
clock = pygame.time.Clock()
while True:
    screen.fill((255, 255, 255))
    pygame.draw.circle(screen, (0, 0, 0), (x, 250), 25)
    x = (x + 2) % 500
    pygame.display.flip()
    clock.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

11. WAP to draw a moving polygon from left to right

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
x = 0
clock = pygame.time.Clock()
while True:
```

```
    screen.fill((255, 255, 255))
    points = [(x, 100), (x+50, 50), (x+100, 100), (x+50, 150)]
    pygame.draw.polygon(screen, (0, 0, 0), points)
    x = (x + 2) % 500
    pygame.display.flip()
    clock.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

12. WAP to draw a moving circle controlled by mouse

python

```
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
clock = pygame.time.Clock()
while True:
    screen.fill((255, 255, 255))
    pos = pygame.mouse.get_pos()
    pygame.draw.circle(screen, (0, 0, 0), pos, 25)
    pygame.display.flip()
    clock.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

13. WAP to draw a moving circle controlled by keyboard

python

```
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
x, y = 250, 250
clock = pygame.time.Clock()
while True:
    screen.fill((255, 255, 255))
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT]: x -= 2
    if keys[pygame.K_RIGHT]: x += 2
    if keys[pygame.K_UP]: y -= 2
    if keys[pygame.K_DOWN]: y += 2
    pygame.draw.circle(screen, (0, 0, 0), (x, y), 25)
```

```
    pygame.display.flip()
    clock.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

14. WAP to draw a moving polygon controlled by mouse

python

```
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
clock = pygame.time.Clock()
while True:
    screen.fill((255, 255, 255))
    pos = pygame.mouse.get_pos()
    points = [(pos[0], pos[1]-50), (pos[0]+50, pos[1]), (pos[0], pos[1]+50), (pos[0]-50
, pos[1])]
    pygame.draw.polygon(screen, (0, 0, 0), points)
    pygame.display.flip()
    clock.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

15. WAP to draw a moving polygon controlled by keyboard

python

```
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
x, y = 250, 250
clock = pygame.time.Clock()
while True:
    screen.fill((255, 255, 255))
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT]: x -= 2
    if keys[pygame.K_RIGHT]: x += 2
    if keys[pygame.K_UP]: y -= 2
    if keys[pygame.K_DOWN]: y += 2
    points = [(x, y-50), (x+50, y), (x, y+50), (x-50, y)]
    pygame.draw.polygon(screen, (0, 0, 0), points)
    pygame.display.flip()
```

```
    clock.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

16. WAP to draw a moving square controlled by mouse

python

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
clock = pygame.time.Clock()
while True:
    screen.fill((255, 255, 255))
    pos = pygame.mouse.get_pos()
    pygame.draw.rect(screen, (0, 0, 0), (pos[0]-25, pos[1]-25, 50, 50))
    pygame.display.flip()
    clock.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

17. WAP to draw a moving square controlled by keyboard

python

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
x, y = 250, 250
clock = pygame.time.Clock()
while True:
    screen.fill((255, 255, 255))
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT]: x -= 2
    if keys[pygame.K_RIGHT]: x += 2
    if keys[pygame.K_UP]: y -= 2
    if keys[pygame.K_DOWN]: y += 2
    pygame.draw.rect(screen, (0, 0, 0), (x-25, y-25, 50, 50))
    pygame.display.flip()
    clock.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
```

```
            exit()
```

18. WAP to draw an animated (multiple movement) square

python

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
x, y, dx, dy = 250, 250, 2, 2
clock = pygame.time.Clock()
while True:
    screen.fill((255, 255, 255))
    if x <= 25 or x >= 475: dx *= -1
    if y <= 25 or y >= 475: dy *= -1
    x += dx
    y += dy
    pygame.draw.rect(screen, (0, 0, 0), (x-25, y-25, 50, 50))
    pygame.display.flip()
    clock.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

19. WAP to draw an animated (multiple movement) circle

python

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
x, y, dx, dy = 250, 250, 2, 2
clock = pygame.time.Clock()
while True:
    screen.fill((255, 255, 255))
    if x <= 25 or x >= 475: dx *= -1
    if y <= 25 or y >= 475: dy *= -1
    x += dx
    y += dy
    pygame.draw.circle(screen, (0, 0, 0), (x, y), 25)
    pygame.display.flip()
    clock.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

20. WAP to draw an animated (multiple movement) polygon

python

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
x, y, dx, dy = 250, 250, 2, 2
clock = pygame.time.Clock()
while True:
    screen.fill((255, 255, 255))
    if x <= 50 or x >= 450: dx *= -1
    if y <= 50 or y >= 450: dy *= -1
    x += dx
    y += dy
    points = [(x, y-50), (x+50, y), (x, y+50), (x-50, y)]
    pygame.draw.polygon(screen, (0, 0, 0), points)
    pygame.display.flip()
    clock.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

21. WAP to draw a human-like structure using circles and lines

python

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((255, 255, 255))
# Head
pygame.draw.circle(screen, (0, 0, 0), (250, 100), 30)
# Body
pygame.draw.line(screen, (0, 0, 0), (250, 130), (250, 250), 2)
# Arms
pygame.draw.line(screen, (0, 0, 0), (250, 150), (200, 200), 2)
pygame.draw.line(screen, (0, 0, 0), (250, 150), (300, 200), 2)
# Legs
pygame.draw.line(screen, (0, 0, 0), (250, 250), (200, 300), 2)
pygame.draw.line(screen, (0, 0, 0), (250, 250), (300, 300), 2)
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

22. WAP to draw a house/hut-like structure using lines and rectangles

python

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((0, 128, 0))
# House base
pygame.draw.rect(screen, (139, 69, 19), (150, 200, 200, 150))
# Roof
pygame.draw.polygon(screen, (128, 0, 0), [(100, 200), (400, 200), (250, 100)])
# Door
pygame.draw.rect(screen, (0, 0, 0), (225, 250, 50, 100))
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

23. WAP to display "Hello" in yellow on blue background and draw a car

python

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((0, 0, 255))
font = pygame.font.SysFont(None, 48)
text = font.render("Hello", True, (255, 255, 0))
screen.blit(text, (200, 50))
# Car body
pygame.draw.rect(screen, (255, 255, 0), (100, 200, 200, 50))
# Wheels
pygame.draw.circle(screen, (0, 0, 0), (150, 275), 25)
pygame.draw.circle(screen, (0, 0, 0), (250, 275), 25)
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

24. WAP to draw a desktop computer (monitor, keyboard, mouse)

python

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((255, 255, 255))
# Monitor
pygame.draw.rect(screen, (0, 0, 0), (150, 100, 200, 150), 2)
pygame.draw.rect(screen, (200, 200, 200), (160, 110, 180, 130))
# Keyboard
pygame.draw.rect(screen, (0, 0, 0), (100, 300, 300, 50), 2)
# Mouse
pygame.draw.ellipse(screen, (0, 0, 0), (420, 320, 50, 30), 2)
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

25. WAP to draw mountain-like structures with hills, trees, and grass

python

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((135, 206, 235))  # Sky
# Mountains
pygame.draw.polygon(screen, (139, 137, 137), [(0, 300), (100, 100), (200, 300)])
pygame.draw.polygon(screen, (139, 137, 137), [(200, 300), (300, 150), (400, 300)])
pygame.draw.polygon(screen, (34, 139, 34), [(0, 300), (500, 300), (500, 500), (0, 500)]
)  # Grass
# Trees
pygame.draw.rect(screen, (139, 69, 19), (50, 250, 20, 50))  # Trunk
pygame.draw.polygon(screen, (0, 100, 0), [(30, 250), (90, 250), (60, 200)])  # Leaves
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

26. WAP to draw moon and stars on a black background

```python
import pygame
import random
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((0, 0, 0))
# Moon
pygame.draw.circle(screen, (255, 255, 255), (100, 100), 50)
# Stars
for _ in range(100):
    x, y = random.randint(0, 500), random.randint(0, 500)
    pygame.draw.circle(screen, (255, 255, 255), (x, y), 1)
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

27. WAP to draw a model resembling a university building

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((255, 255, 255))
# Main building
pygame.draw.rect(screen, (70, 70, 70), (100, 200, 300, 150))
# Roof
pygame.draw.polygon(screen, (100, 100, 100), [(100, 200), (400, 200), (350, 150), (150,
150)])
# Entrance
pygame.draw.rect(screen, (0, 0, 0), (225, 250, 50, 100))
# Windows
for x in range(120, 380, 60):
    pygame.draw.rect(screen, (135, 206, 235), (x, 220, 40, 40))
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

28. WAP to draw an animated snowfall view

```python
import pygame
import random
pygame.init()
screen = pygame.display.set_mode((500, 500))
snowflakes = [[random.randint(0, 500), random.randint(0, 500)] for _ in range(100)]
clock = pygame.time.Clock()
while True:
    screen.fill((0, 0, 50))
    for flake in snowflakes:
        pygame.draw.circle(screen, (255, 255, 255), (flake[0], flake[1]), 2)
        flake[1] += 1
        if flake[1] > 500:
            flake[1] = 0
            flake[0] = random.randint(0, 500)
    pygame.display.flip()
    clock.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

29. WAP to draw an animated text "Hello"

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
font = pygame.font.SysFont(None, 48)
angle = 0
clock = pygame.time.Clock()
while True:
    screen.fill((255, 255, 255))
    text = font.render("Hello", True, (0, 0, 0))
    text = pygame.transform.rotate(text, angle)
    screen.blit(text, (200, 200))
    angle = (angle + 1) % 360
    pygame.display.flip()
    clock.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

30. WAP to design a simple computer game (Pong)

```python
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
paddle = pygame.Rect(200, 450, 100, 20)
ball = pygame.Rect(250, 250, 20, 20)
ball_dx, ball_dy = 3, 3
clock = pygame.time.Clock()
while True:
    screen.fill((0, 0, 0))
    pygame.draw.rect(screen, (255, 255, 255), paddle)
    pygame.draw.ellipse(screen, (255, 255, 255), ball)
    ball.x += ball_dx
    ball.y += ball_dy
    if ball.left <= 0 or ball.right >= 500: ball_dx *= -1
    if ball.top <= 0: ball_dy *= -1
    if ball.colliderect(paddle): ball_dy *= -1
    if ball.bottom >= 500: ball.x, ball.y = 250, 250
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT] and paddle.left > 0: paddle.x -= 5
    if keys[pygame.K_RIGHT] and paddle.right < 500: paddle.x += 5
    pygame.display.flip()
    clock.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

31. WAP to draw a line using DDA algorithm

```python
import pygame
def draw_line_dda(surface, color, start, end):
    x1, y1 = start
    x2, y2 = end
    dx, dy = x2 - x1, y2 - y1
    steps = max(abs(dx), abs(dy))
    if steps == 0: return
    x_inc, y_inc = dx / steps, dy / steps
    x, y = x1, y1
    for _ in range(steps + 1):
        surface.set_at((round(x), round(y)), color)
        x += x_inc
        y += y_inc

pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((255, 255, 255))
```

```
draw_line_dda(screen, (0, 0, 0), (100, 100), (400, 300))
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

32. WAP to draw a line using Bresenham's algorithm

python

```
import pygame
def draw_line_bresenham(surface, color, start, end):
    x1, y1 = start
    x2, y2 = end
    dx, dy = abs(x2 - x1), abs(y2 - y1)
    sx = 1 if x1 < x2 else -1
    sy = 1 if y1 < y2 else -1
    err = dx - dy
    while True:
        surface.set_at((x1, y1), color)
        if x1 == x2 and y1 == y2: break
        e2 = 2 * err
        if e2 > -dy:
            err -= dy
            x1 += sx
        if e2 < dx:
            err += dx
            y1 += sy

pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((255, 255, 255))
draw_line_bresenham(screen, (0, 0, 0), (100, 100), (400, 300))
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

33. WAP to draw a circle using midpoint algorithm

python

```
import pygame
```

```python
def draw_circle_midpoint(surface, color, center, radius):
    x0, y0 = center
    x, y = radius, 0
    err = 0
    while x >= y:
        surface.set_at((x0 + x, y0 + y), color)
        surface.set_at((x0 + y, y0 + x), color)
        surface.set_at((x0 - y, y0 + x), color)
        surface.set_at((x0 - x, y0 + y), color)
        surface.set_at((x0 - x, y0 - y), color)
        surface.set_at((x0 - y, y0 - x), color)
        surface.set_at((x0 + y, y0 - x), color)
        surface.set_at((x0 + x, y0 - y), color)
        y += 1
        err += 1 + 2 * y
        if 2 * (err - x) + 1 > 0:
            x -= 1
            err += 1 - 2 * x

pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((255, 255, 255))
draw_circle_midpoint(screen, (0, 0, 0), (250, 250), 100)
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

34. WAP to translate a given point by (4, 5)

python

```python
def translate_point(point, tx, ty):
    return (point[0] + tx, point[1] + ty)

original = (100, 100)
translated = translate_point(original, 4, 5)
print(f"Original: {original}, Translated: {translated}")
```

35. WAP to translate a given triangle by (-4, 5)

python

```python
def translate_triangle(triangle, tx, ty):
    return [(x + tx, y + ty) for x, y in triangle]
```

```
original = [(100, 100), (150, 150), (200, 100)]
translated = translate_triangle(original, -4, 5)
print(f"Original: {original}, Translated: {translated}")
```

36. WAP to translate a given polygon by (-4, -5)

python

```python
def translate_polygon(polygon, tx, ty):
    return [(x + tx, y + ty) for x, y in polygon]

original = [(100, 100), (150, 50), (200, 100), (175, 150), (125, 150)]
translated = translate_polygon(original, -4, -5)
print(f"Original: {original}, Translated: {translated}")
```

37. WAP to rotate a given point by -90 degrees

python

```python
import math
def rotate_point(point, angle, origin=(0, 0)):
    angle_rad = math.radians(angle)
    ox, oy = origin
    px, py = point
    qx = ox + math.cos(angle_rad) * (px - ox) - math.sin(angle_rad) * (py - oy)
    qy = oy + math.sin(angle_rad) * (px - ox) + math.cos(angle_rad) * (py - oy)
    return (round(qx), round(qy))

original = (100, 100)
rotated = rotate_point(original, -90)
print(f"Original: {original}, Rotated: {rotated}")
```

38. WAP to rotate a given triangle by 40 degrees

python

```python
import math
def rotate_triangle(triangle, angle, origin=(0, 0)):
    angle_rad = math.radians(angle)
    ox, oy = origin
    rotated = []
    for px, py in triangle:
        qx = ox + math.cos(angle_rad) * (px - ox) - math.sin(angle_rad) * (py - oy)
        qy = oy + math.sin(angle_rad) * (px - ox) + math.cos(angle_rad) * (py - oy)
```

```
        rotated.append((round(qx), round(qy)))
    return rotated

original = [(100, 100), (150, 150), (200, 100)]
rotated = rotate_triangle(original, 40)
print(f"Original: {original}, Rotated: {rotated}")
```

39. WAP to rotate a given polygon by 120 degrees

python

```
import math
def rotate_polygon(polygon, angle, origin=(0, 0)):
    angle_rad = math.radians(angle)
    ox, oy = origin
    rotated = []
    for px, py in polygon:
        qx = ox + math.cos(angle_rad) * (px - ox) - math.sin(angle_rad) * (py - oy)
        qy = oy + math.sin(angle_rad) * (px - ox) + math.cos(angle_rad) * (py - oy)
        rotated.append((round(qx), round(qy)))
    return rotated

original = [(100, 100), (150, 50), (200, 100), (175, 150), (125, 150)]
rotated = rotate_polygon(original, 120)
print(f"Original: {original}, Rotated: {rotated}")
```

40. WAP to reflect a given point over the x-axis

python

```
def reflect_point_x(point):
    return (point[0], -point[1])

original = (100, 100)
reflected = reflect_point_x(original)
print(f"Original: {original}, Reflected: {reflected}")
```

41. WAP to reflect a given triangle over the x-axis

python

```
def reflect_triangle_x(triangle):
    return [(x, -y) for x, y in triangle]

original = [(100, 100), (150, 150), (200, 100)]
```

```
reflected = reflect_triangle_x(original)
print(f"Original: {original}, Reflected: {reflected}")
```

42. WAP to reflect a given polygon over the x-axis

python

```
def reflect_polygon_x(polygon):
    return [(x, -y) for x, y in polygon]

original = [(100, 100), (150, 50), (200, 100), (175, 150), (125, 150)]
reflected = reflect_polygon_x(original)
print(f"Original: {original}, Reflected: {reflected}")
```

43. WAP to reflect a given point over the line x = y

python

```
def reflect_point_xy(point):
    return (point[1], point[0])

original = (100, 200)
reflected = reflect_point_xy(original)
print(f"Original: {original}, Reflected: {reflected}")
```

44. WAP to reflect a given triangle over the line x = y

python

```
def reflect_triangle_xy(triangle):
    return [(y, x) for x, y in triangle]

original = [(100, 100), (150, 150), (200, 100)]
reflected = reflect_triangle_xy(original)
print(f"Original: {original}, Reflected: {reflected}")
```

45. WAP to reflect a given polygon over the line x = y

python

```
def reflect_polygon_xy(polygon):
    return [(y, x) for x, y in polygon]
```

```python
original = [(100, 100), (150, 50), (200, 100), (175, 150), (125, 150)]
reflected = reflect_polygon_xy(original)
print(f"Original: {original}, Reflected: {reflected}")
```

46. WAP to shear a given figure

python

```python
def shear_polygon(polygon, shx, shy):
    return [(x + shx * y, y + shy * x) for x, y in polygon]

original = [(100, 100), (150, 150), (200, 100)]
sheared = shear_polygon(original, 0.5, 0)
print(f"Original: {original}, Sheared: {sheared}")
```

47. WAP to scale a given figure

python

```python
def scale_polygon(polygon, sx, sy):
    return [(x * sx, y * sy) for x, y in polygon]

original = [(100, 100), (150, 150), (200, 100)]
scaled = scale_polygon(original, 2, 0.5)
print(f"Original: {original}, Scaled: {scaled}")
```

48. WAP to perform translation and rotation on a given figure

python

```python
import math
def transform_polygon(polygon, tx, ty, angle):
    angle_rad = math.radians(angle)
    transformed = []
    for x, y in polygon:
        # Translate
        x += tx
        y += ty
        # Rotate around origin
        qx = math.cos(angle_rad) * x - math.sin(angle_rad) * y
        qy = math.sin(angle_rad) * x + math.cos(angle_rad) * y
        transformed.append((round(qx), round(qy)))
    return transformed

original = [(100, 100), (150, 150), (200, 100)]
```

```
transformed = transform_polygon(original, 10, 20, 30)
print(f"Original: {original}, Transformed: {transformed}")
```

49. WAP to perform scaling and rotation on a given figure

python

```python
import math
def transform_polygon(polygon, sx, sy, angle):
    angle_rad = math.radians(angle)
    transformed = []
    for x, y in polygon:
        # Scale
        x *= sx
        y *= sy
        # Rotate around origin
        qx = math.cos(angle_rad) * x - math.sin(angle_rad) * y
        qy = math.sin(angle_rad) * x + math.cos(angle_rad) * y
        transformed.append((round(qx), round(qy)))
    return transformed

original = [(100, 100), (150, 150), (200, 100)]
transformed = transform_polygon(original, 2, 0.5, 45)
print(f"Original: {original}, Transformed: {transformed}")
```

50. WAP to perform translation, rotation, and scaling on a given figure

python

```python
import math
def transform_polygon(polygon, tx, ty, sx, sy, angle):
    angle_rad = math.radians(angle)
    transformed = []
    for x, y in polygon:
        # Scale
        x *= sx
        y *= sy
        # Translate
        x += tx
        y += ty
        # Rotate around origin
        qx = math.cos(angle_rad) * x - math.sin(angle_rad) * y
        qy = math.sin(angle_rad) * x + math.cos(angle_rad) * y
        transformed.append((round(qx), round(qy)))
    return transformed

original = [(100, 100), (150, 150), (200, 100)]
```

```
transformed = transform_polygon(original, 10, 20, 2, 0.5, 30)
print(f"Original: {original}, Transformed: {transformed}")
```

51. WAP to perform shearing, translation, rotation, and scaling on a given figure

python

```python
import math
def transform_polygon(polygon, shx, shy, tx, ty, sx, sy, angle):
    angle_rad = math.radians(angle)
    transformed = []
    for x, y in polygon:
        # Shear
        x += shx * y
        y += shy * x
        # Scale
        x *= sx
        y *= sy
        # Translate
        x += tx
        y += ty
        # Rotate around origin
        qx = math.cos(angle_rad) * x - math.sin(angle_rad) * y
        qy = math.sin(angle_rad) * x + math.cos(angle_rad) * y
        transformed.append((round(qx), round(qy)))
    return transformed

original = [(100, 100), (150, 150), (200, 100)]
transformed = transform_polygon(original, 0.5, 0, 10, 20, 2, 0.5, 30)
print(f"Original: {original}, Transformed: {transformed}")
```

52. WAP to show flood fill algorithm

python

```python
import pygame
def flood_fill(surface, pos, target_color, fill_color):
    x, y = pos
    if surface.get_at((x, y)) != target_color:
        return
    queue = [(x, y)]
    while queue:
        x, y = queue.pop(0)
        if surface.get_at((x, y)) == target_color:
            surface.set_at((x, y), fill_color)
            for dx, dy in [(1, 0), (-1, 0), (0, 1), (0, -1)]:
                nx, ny = x + dx, y + dy
```

```
                if 0 <= nx < surface.get_width() and 0 <= ny < surface.get_height():
                    queue.append((nx, ny))

pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((255, 255, 255))
pygame.draw.rect(screen, (0, 0, 0), (100, 100, 200, 200), 2)
flood_fill(screen, (150, 150), (255, 255, 255), (255, 0, 0))
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

53. WAP to show boundary fill algorithm

python

```
import pygame
def boundary_fill(surface, pos, boundary_color, fill_color):
    x, y = pos
    if surface.get_at((x, y)) == boundary_color or surface.get_at((x, y)) == fill_color
:
        return
    queue = [(x, y)]
    while queue:
        x, y = queue.pop(0)
        if surface.get_at((x, y)) != boundary_color and surface.get_at((x, y)) != fill_
color:
            surface.set_at((x, y), fill_color)
            for dx, dy in [(1, 0), (-1, 0), (0, 1), (0, -1)]:
                nx, ny = x + dx, y + dy
                if 0 <= nx < surface.get_width() and 0 <= ny < surface.get_height():
                    queue.append((nx, ny))

pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((255, 255, 255))
pygame.draw.rect(screen, (0, 0, 0), (100, 100, 200, 200), 2)
boundary_fill(screen, (150, 150), (0, 0, 0), (0, 255, 0))
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

54. WAP to show Cohen-Sutherland Line Clipping algorithm

python

```
def compute_outcode(x, y, xmin, ymin, xmax, ymax):
    code = 0
    if x < xmin: code |= 1
    elif x > xmax: code |= 2
    if y < ymin: code |= 4
    elif y > ymax: code |= 8
    return code

def cohen_sutherland(x0, y0, x1, y1, xmin, ymin, xmax, ymax):
    outcode0 = compute_outcode(x0, y0, xmin, ymin, xmax, ymax)
    outcode1 = compute_outcode(x1, y1, xmin, ymin, xmax, ymax)
    while True:
        if not (outcode0 | outcode1): return (x0, y0, x1, y1)
        if outcode0 & outcode1: return None
        outcode = outcode0 if outcode0 else outcode1
        if outcode & 1: x, y = xmin, y0 + (y1 - y0) * (xmin - x0) / (x1 - x0)
        elif outcode & 2: x, y = xmax, y0 + (y1 - y0) * (xmax - x0) / (x1 - x0)
        elif outcode & 4: x, y = x0 + (x1 - x0) * (ymin - y0) / (y1 - y0), ymin
        elif outcode & 8: x, y = x0 + (x1 - x0) * (ymax - y0) / (y1 - y0), ymax
        if outcode == outcode0: x0, y0, outcode0 = x, y, compute_outcode(x, y, xmin, ym
in, xmax, ymax)
        else: x1, y1, outcode1 = x, y, compute_outcode(x, y, xmin, ymin, xmax, ymax)

result = cohen_sutherland(50, 50, 250, 250, 100, 100, 200, 200)
print(f"Clipped line: {result}")
```

55. WAP to show translation on a composite figure

python

```
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((255, 255, 255))
# Original composite figure (house)
pygame.draw.rect(screen, (139, 69, 19), (150, 200, 200, 150))
pygame.draw.polygon(screen, (128, 0, 0), [(100, 200), (400, 200), (250, 100)])
# Translated composite figure
pygame.draw.rect(screen, (139, 69, 19), (200, 250, 200, 150))
pygame.draw.polygon(screen, (128, 0, 0), [(150, 250), (450, 250), (300, 150)])
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
```

```
            pygame.quit()
            exit()
```

56. WAP to show rotation on a composite figure

python

```
import pygame
import math
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((255, 255, 255))
def rotate_point(point, angle, origin):
    angle_rad = math.radians(angle)
    ox, oy = origin
    px, py = point
    qx = ox + math.cos(angle_rad) * (px - ox) - math.sin(angle_rad) * (py - oy)
    qy = oy + math.sin(angle_rad) * (px - ox) + math.cos(angle_rad) * (py - oy)
    return (round(qx), round(qy))

# Original composite figure (house)
original_house = [(150, 200), (350, 200), (350, 350), (150, 350)]
original_roof = [(100, 200), (400, 200), (250, 100)]
origin = (250, 200)
# Rotated composite figure
rotated_house = [rotate_point(point, 30, origin) for point in original_house]
rotated_roof = [rotate_point(point, 30, origin) for point in original_roof]
pygame.draw.polygon(screen, (139, 69, 19), rotated_house)
pygame.draw.polygon(screen, (128, 0, 0), rotated_roof)
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

57. WAP to show reflection on a composite figure

python

```
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((255, 255, 255))
# Original composite figure (house)
pygame.draw.rect(screen, (139, 69, 19), (150, 200, 200, 150))
pygame.draw.polygon(screen, (128, 0, 0), [(100, 200), (400, 200), (250, 100)])
```

```
# Reflected composite figure (over x-axis)
pygame.draw.rect(screen, (139, 69, 19), (150, 500-200-150, 200, 150))
pygame.draw.polygon(screen, (128, 0, 0), [(100, 500-200), (400, 500-200), (250, 500-100
)])
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

58. WAP to show scaling on a composite figure

python

```
import pygame
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((255, 255, 255))
def scale_point(point, sx, sy, origin):
    ox, oy = origin
    px, py = point
    qx = ox + (px - ox) * sx
    qy = oy + (py - oy) * sy
    return (round(qx), round(qy))

# Original composite figure (house)
original_house = [(150, 200), (350, 200), (350, 350), (150, 350)]
original_roof = [(100, 200), (400, 200), (250, 100)]
origin = (250, 200)
# Scaled composite figure
scaled_house = [scale_point(point, 1.5, 1.5, origin) for point in original_house]
scaled_roof = [scale_point(point, 1.5, 1.5, origin) for point in original_roof]
pygame.draw.polygon(screen, (139, 69, 19), scaled_house)
pygame.draw.polygon(screen, (128, 0, 0), scaled_roof)
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

59. WAP to show shearing on a composite figure

python

```
import pygame
```

```python
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((255, 255, 255))
def shear_point(point, shx, shy, origin):
    ox, oy = origin
    px, py = point
    qx = px + shx * (py - oy)
    qy = py + shy * (px - ox)
    return (round(qx), round(qy))

# Original composite figure (house)
original_house = [(150, 200), (350, 200), (350, 350), (150, 350)]
original_roof = [(100, 200), (400, 200), (250, 100)]
origin = (250, 200)
# Sheared composite figure
sheared_house = [shear_point(point, 0.5, 0, origin) for point in original_house]
sheared_roof = [shear_point(point, 0.5, 0, origin) for point in original_roof]
pygame.draw.polygon(screen, (139, 69, 19), sheared_house)
pygame.draw.polygon(screen, (128, 0, 0), sheared_roof)
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

60. WAP to show composite operations on a composite figure

python

```python
import pygame
import math
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((255, 255, 255))
def transform_point(point, tx, ty, sx, sy, shx, shy, angle, origin):
    angle_rad = math.radians(angle)
    ox, oy = origin
    px, py = point
    # Translate to origin
    px -= ox
    py -= oy
    # Shear
    px += shx * py
    py += shy * px
    # Scale
    px *= sx
    py *= sy
    # Rotate
```

```
    qx = math.cos(angle_rad) * px - math.sin(angle_rad) * py
    qy = math.sin(angle_rad) * px + math.cos(angle_rad) * py
    # Translate back and add final translation
    qx += ox + tx
    qy += oy + ty
    return (round(qx), round(qy))

# Original composite figure (house)
original_house = [(150, 200), (350, 200), (350, 350), (150, 350)]
original_roof = [(100, 200), (400, 200), (250, 100)]
origin = (250, 200)
# Transformed composite figure
transformed_house = [transform_point(point, 50, 50, 1.2, 1.2, 0.2, 0, 30, origin) for p
oint in original_house]
transformed_roof = [transform_point(point, 50, 50, 1.2, 1.2, 0.2, 0, 30, origin) for po
int in original_roof]
pygame.draw.polygon(screen, (139, 69, 19), transformed_house)
pygame.draw.polygon(screen, (128, 0, 0), transformed_roof)
pygame.display.flip()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

61. WAP to show a 3-D object (cube)

python

```
import pygame
import math
pygame.init()
screen = pygame.display.set_mode((500, 500))
screen.fill((255, 255, 255))
def project_3d_to_2d(x, y, z):
    scale = 200 / (200 + z)
    return (250 + x * scale, 250 + y * scale)

# Cube vertices
vertices = [(-50, -50, -50), (50, -50, -50), (50, 50, -50), (-50, 50, -50),
            (-50, -50, 50), (50, -50, 50), (50, 50, 50), (-50, 50, 50)]
# Cube edges
edges = [(0, 1), (1, 2), (2, 3), (3, 0),
         (4, 5), (5, 6), (6, 7), (7, 4),
         (0, 4), (1, 5), (2, 6), (3, 7)]
# Rotate cube
angle = 0
clock = pygame.time.Clock()
while True:
```

```
    screen.fill((255, 255, 255))
    angle += 1
    rotated_vertices = []
    for x, y, z in vertices:
        # Rotate around Y axis
        rx = x * math.cos(math.radians(angle)) - z * math.sin(math.radians(angle))
        rz = x * math.sin(math.radians(angle)) + z * math.cos(math.radians(angle))
        ry = y
        # Project to 2D
        px, py = project_3d_to_2d(rx, ry, rz)
        rotated_vertices.append((px, py))
    # Draw edges
    for edge in edges:
        pygame.draw.line(screen, (0, 0, 0), rotated_vertices[edge[0]], rotated_vertices
[edge[1]]), 2)
    pygame.display.flip()
    clock.tick(60)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
```

Eac