ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ № 4

за курсом “Комп’ютерна графіка та анімація”

студента групи ПЗ-24у-1

Кондрачука Олександра Вадимовича

2024/2025 н.р.

1. **Постановка задачі**

На основі програми до лабораторної работи №3, розробити прикладну програму, яка переносе тіло у точку зі вказаними координатами. Також додати можливість рухати тіло за заданою траєкторією. Траєкторію (функцію) обрати самостійно. Передбачити окремі елементи керування для кожного перетворення. Додати можливість реагувати на досягнення країв екрану.

1. **Опис розв’язку**

Створив програмне вікно за допомогою мови програмування python та графічної бібліотеки pygame.

Задав тривимірну літеру «А» за допомогою масиву точок , що відповідають вершинам літери та масиву пар точок, що відповідають ребрам, що з’єднують вершини.

Щоб відобразити об’єм, додав фактор перспективи, яке відповідаю розміру одиниці об’єкта у пікселях. Потім ділив фактор на z координату, щоб дізнатися точку об’єкта на екрані.

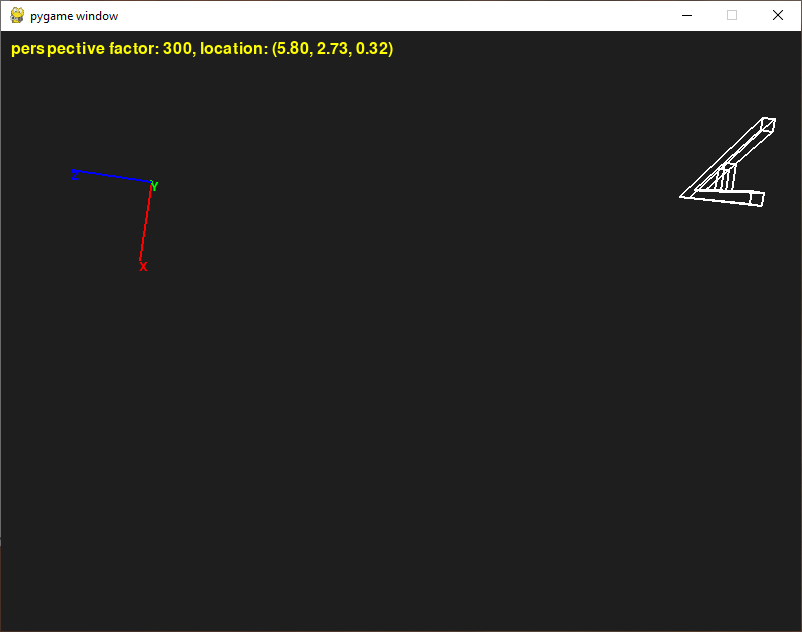
Для повороту по осям використовував множення матриць.

Для повороту об’єкта зчитував натискання клавіш з клавіатури.

Додав переміщення об’єкта за допомогою миші та новоствореної функції, що трансформує точку на екрані у точку на об’єкті.

Додав анімацію, що переміщує об’єкт та розвертає його у заздалегідь задані координати.

1. **Тестування програми**

****

1. **Додаток**

import pygame  
import numpy as np  
import math  
  
SCREEN\_WIDTH, SCREEN\_HEIGHT = 800, 600  
  
axes = [  
    ([0, 0, 0], [2, 0, 0]),   
    ([0, 0, 0], [0, 2, 0]),   
    ([0, 0, 0], [0, 0, 2])   
]  
  
vertices = [  
    [-0.5, -0.5, -0.1],     
    [0, 0.5, -0.1],      
    [0.5, -0.5, -0.1],     
       
    [-0.25, 0, -0.1],  
    [0.25, 0, -0.1],  
      
    [-0.7, -0.5, -0.1],     
    [0, 0.8, -0.1],      
    [0.7, -0.5, -0.1],     
       
    [-0.2, 0.1, -0.1],  
    [0.2, 0.1, -0.1],  
      
      
    [-0.5, -0.5, 0.1],     
    [0, 0.5, 0.1],      
    [0.5, -0.5, 0.1],     
       
    [-0.25, 0, 0.1],  
    [0.25, 0, 0.1],  
      
    [-0.7, -0.5, 0.1],     
    [0, 0.8, 0.1],      
    [0.7, -0.5, 0.1],     
       
    [-0.2, 0.1, 0.1],  
    [0.2, 0.1, 0.1]  
]  
  
edges = [  
    (0, 1),   
    (1, 2),       
    (3, 4),  
                   
    (5, 6),   
    (6, 7),       
    (8, 9),  
      
    (0, 5),  
    (2, 7),  
      
      
    (10, 11),   
    (11, 12),       
    (13, 14),  
                   
    (15, 16),   
    (16, 17),       
    (18, 19),  
      
    (10, 15),  
    (12, 17),  
      
      
    (0, 10),  
    (1, 11),  
    (2, 12),  
    (3, 13),  
    (4, 14),  
    (5, 15),  
    (6, 16),  
    (7, 17),  
    (8, 18),  
    (9, 19)  
]  
  
object\_center = [0, 0, 0]  
angles = [0, 0, 0]  
z\_offset = 5  
  
animation\_start\_point = [0, 0, 0]  
animation\_start\_rotation = [0, 0, 0]  
animation\_end\_point = [0.3, 0.1, 0.1]  
animation\_end\_rotation = [0, 0, 0]  
animation\_duration = 1.2  
animation\_elapsed\_time = animation\_duration  
  
def lerp(start, end, t):  
    result = [0, 0, 0]  
      
    result[0] = start[0] \* (1 - t) + end[0] \* t  
    result[1] = start[1] \* (1 - t) + end[1] \* t  
    result[2] = start[2] \* (1 - t) + end[2] \* t  
  
    return result  
  
def project\_point(point):  
    factor = perspective\_factor / (point[2] + z\_offset + object\_center[2])  
    x = (point[0] + object\_center[0]) \* factor + SCREEN\_WIDTH // 2  
    y = -(point[1] + object\_center[1]) \* factor + SCREEN\_HEIGHT // 2  
    return (int(x), int(y))  
  
def play\_move\_animation():  
    global object\_center, animation\_elapsed\_time, angles  
      
    animation\_elapsed\_time += 1/ 60  
      
    if animation\_elapsed\_time < animation\_duration:  
        if animation\_elapsed\_time + 1 / 60 > animation\_duration:  
            object\_center = animation\_end\_point.copy()  
            angles = animation\_end\_rotation.copy()  
        else:  
            object\_center = lerp(animation\_start\_point, animation\_end\_point, animation\_elapsed\_time / animation\_duration)  
            angles = lerp(animation\_start\_rotation, animation\_end\_rotation, animation\_elapsed\_time / animation\_duration)  
              
  
def inverse\_point(screen\_point):  
    factor = perspective\_factor / (z\_offset + object\_center[2])  
      
    object\_center[0] = (screen\_point[0] - SCREEN\_WIDTH // 2) / factor  
    object\_center[1] = -(screen\_point[1] - SCREEN\_HEIGHT // 2) / factor  
  
def rotate(point, angle, axis):  
    sin\_a = math.sin(angle)  
    cos\_a = math.cos(angle)  
      
    axis\_variants = [0, 1, 2]  
    axis\_variants.remove(axis)  
      
    first, second = axis\_variants  
    first\_transformed = point[first] \* cos\_a - point[second] \* sin\_a  
    second\_transformed = point[first] \* sin\_a + point[second] \* cos\_a  
      
    result = [point[0], point[1], point[2]]  
    result[first] = first\_transformed  
    result[second] = second\_transformed  
      
    return [result[0], result[1], result[2]]  
  
def axis\_draw():  
    axis\_origin = (150, 150)  
    axis\_factor = 40         
  
    for i, (start, end) in enumerate(axes):  
        rotated\_start = rotate(start, angles[0], 0)  
        rotated\_start = rotate(rotated\_start, angles[1], 1)  
        rotated\_start = rotate(rotated\_start, angles[2], 2)  
  
        rotated\_end = rotate(end, angles[0], 0)  
        rotated\_end = rotate(rotated\_end, angles[1], 1)  
        rotated\_end = rotate(rotated\_end, angles[2], 2)  
  
        projected\_start = (  
            int(rotated\_start[0] \* axis\_factor + axis\_origin[0]),  
            int(-rotated\_start[1] \* axis\_factor + axis\_origin[1])  
        )  
        projected\_end = (  
            int(rotated\_end[0] \* axis\_factor + axis\_origin[0]),  
            int(-rotated\_end[1] \* axis\_factor + axis\_origin[1])  
        )  
  
        color = [(255, 0, 0), (0, 255, 0), (0, 0, 255)][i]  
        pygame.draw.line(screen, color, projected\_start, projected\_end, 2)  
  
        axis\_label = ['X', 'Y', 'Z'][i]  
        font = pygame.font.SysFont(None, 20)  
        label = font.render(axis\_label, True, color)  
        screen.blit(label, projected\_end)  
  
  
pygame.init()  
  
screen = pygame.display.set\_mode((SCREEN\_WIDTH, SCREEN\_HEIGHT))  
clock = pygame.time.Clock()  
  
running = True  
dragging = False  
  
perspective\_factor = 300  
  
while running:  
    clock.tick(60)  
    screen.fill((30, 30, 30))  
  
    keys = pygame.key.get\_pressed()  
  
    if keys[pygame.K\_s]:  
        angles[0] += 0.02  
    if keys[pygame.K\_z]:  
        angles[1] += 0.02  
    if keys[pygame.K\_a]:  
        angles[2] += 0.02  
          
    if keys[pygame.K\_w]:  
        angles[0] -= 0.02  
    if keys[pygame.K\_x]:  
        angles[1] -= 0.02  
    if keys[pygame.K\_d]:  
        angles[2] -= 0.02  
  
    if keys[pygame.K\_UP]:  
        object\_center[2] += 0.02  
    if keys[pygame.K\_DOWN]:  
        object\_center[2] -= 0.02  
        object\_center[2] = max(0, object\_center[2])  
          
    if keys[pygame.K\_j]:  
        animation\_elapsed\_time = 0  
        animation\_start\_point = object\_center  
        animation\_start\_rotation = angles  
  
    for event in pygame.event.get():  
        if event.type == pygame.QUIT:  
            running = False  
        elif event.type == pygame.MOUSEBUTTONDOWN:  
            if event.button == 1:              
                dragging = True  
                inverse\_point(event.pos)  
        elif event.type == pygame.MOUSEBUTTONUP:  
            if event.button == 1:              
                dragging = False  
        elif event.type == pygame.MOUSEMOTION:  
            if dragging:  
                inverse\_point(event.pos)  
  
    play\_move\_animation()  
  
    transformed = []  
    for v in vertices:  
        rotated = rotate(v, angles[0], 0)  
        rotated = rotate(rotated, angles[1], 1)  
        rotated = rotate(rotated, angles[2], 2)  
          
        projected = project\_point(rotated)  
        transformed.append(projected)  
  
    for edge in edges:  
        pygame.draw.line(screen, (255, 255, 255), transformed[edge[0]], transformed[edge[1]], 2)  
  
    font = pygame.font.SysFont(None, 24)  
    info = font.render(f'perspective factor: {perspective\_factor}, location: ({object\_center[0]:.2f}, {object\_center[1]:.2f}, {object\_center[2]:.2f})', True, (255, 255, 0))  
    screen.blit(info, (10, 10))  
      
    axis\_draw()  
  
    pygame.display.flip()  
  
pygame.quit()