Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет Інформатики та Обчислювальної Техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №9

з дисципліни «Технології Computer Vision»

на тему

«СИНТЕЗ РЕАЛІСТИЧНИХ ОБ’ЄКТІВ ДОПОВНЕНОЇ РЕАЛЬНОСТІ»

Виконала:
студентка групи ІС-12
Павлова Софія

Перевірив:
Баран Д. Р.

Київ – 2024

1. Постановка задачі

Мета роботи:

Дослідити методологію і технології створення доповненої реальності.

Завдання І рівня:

З використанням методів бібліотеки OpenGL розробити скрипт, що реалізує реалістичну візуалізацію графічної сцени у композиції та відповідно до взаємовідносин об’єктів сцени:

в графічному вікні розташовано 3D багатокутник в аксонометричній проекції та 3D модель поверхні другого порядку – типи фігур та їх кількість, розмір, взаємне розташування обрати самостійно;

обрані фігури освітлюються точковим джерелом світла, модель світла, метод зафарбовування поверхонь для моделювання світло і тіні - обирається самостійно;

взаємне розташування «акторів» сцени: спостерігач, геометричні фігури, джерело світла – обрати самостійно;

передбачити анімацію сцени, шляхом обертання геометричних фігур відносно нерухомих спостеригача та джерела світла.

Порядок реалізації завдання відобразити у формі структурної схеми етапів конвеєру.

2. Виконання

2.1. Реалістична візуалізація графічної сцени

Композиція

Виберемо композицію, подібну до першої лабораторної роботи (лр 1). Візьмемо 2 фігури: паралелепіпед і додамо сферу.

Алгоритм програми

Представимо алгоритм програми у вигляді структурної схеми етапів конвеєру.

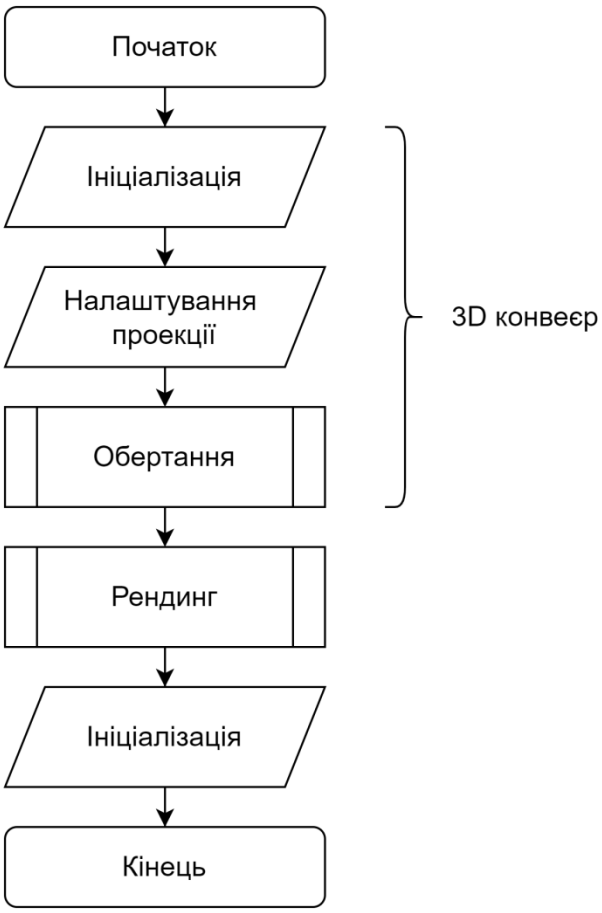

Рисунок 1 – Блок схема програми

Ініціалізація геометричних об’єктів

Візьмемо параметри паралелепіпеда з лр 1, колір – сірий. Додамо одиничну сферу, колір – жовтий.

Лістинг коду:

```
# Вершини паралелепіпеда
prlpd = np.array([
    [0, 0, 0],
    [2, 0, 0],
    [2, 1, 0],
    [0, 1, 0],
    [0, 0, 2],
    [2, 0, 2],
    [2, 1, 2],
    [0, 1, 2]
])
```

```
# Грані паралелепіпеда
faces = np.array([
    [0, 1, 2, 3],  # нижня грань
    [4, 5, 6, 7],  # верхня грань
    [0, 1, 5, 4],  # передня грань
    [2, 3, 7, 6],  # задня грань
    [0, 3, 7, 4],  # ліва грань
    [1, 2, 6, 5]   # права грань
])

# Нормалі до граней паралелепіпеда
normals = np.array([
    [0, 0, -1],  # нижня грань
    [0, 0, 1],   # верхня грань
    [0, -1, 0],  # передня грань
    [0, 1, 0],   # задня грань
    [-1, 0, 0],  # ліва грань
    [1, 0, 0]    # права грань
])
```

Ініціалізація сцени

Ініціалізуємо сцену білого кольору.

Лістинг коду:

```
# Ініціалізація сцени
def init():
    glClearColor(1.0, 1.0, 1.0, 1.0)
    glEnable(GL_DEPTH_TEST)
    glEnable(GL_LIGHTING)
    glEnable(GL_LIGHT0)
    glEnable(GL_COLOR_MATERIAL)
    glShadeModel(GL_FLAT)
```

Налаштування камери

Налаштуємо камеру збоку від сцени, щоб було добре видно обертання. Розташуємо її за 8 одиниць від центру сцени.

Лістинг коду:

```
# Налаштування камери
gluLookAt(8, 8, 8, 0, 0, 0, 0, 1, 0)  # Камера далі від сцени
```

Налаштування джерела світла

Розташуємо джерело світла на координатах (5, 5, 5). Джерело світла також має четвертий параметр, що дорівнює одиниці, що вказує, що це точкове джерело світла.

Лістинг коду:

```
# Визначення джерела світла
light_pos = [5, 5, 5, 1]
glLightfv(GL_LIGHT0, GL_POSITION, light_pos)
```

Візуалізація паралелепіпеда

Відобразимо паралелепіпед шляхом визначення його граней, вершин і нормалей.

Лістинг коду:

```
# Візуалізація паралелепіпеда
def draw_prlpd():
    glBegin(GL_QUADS)
    for i, face in enumerate(faces):
        glNormal3fv(normals[i])
        for vertex in face:
            glVertex3fv(prlpd[vertex])
    glEnd()
```

Рендеринг

Програма буде виконуватись у циклі, щоб утворити анімацію.

Установимо початкове значення кута, параметри камери та джерела освітлення. Виконаємо обертання композиції об’єктів навколо осі X і відобразимо паралелепіпед та сферу з відповідним зміщенням та центруванням. Збільшимо значення кута і додамо затримку для плавної візуалізації анімування.

Лістинг коду:

```
# Візуалізація
def display():
    global angle

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glLoadIdentity()

    # Налаштування камери
    gluLookAt(8, 8, 8, 0, 0, 0, 0, 1, 0)  # Камера далі від сцени

    # Визначення джерела світла
    light_pos = [5, 5, 5, 1]
    glLightfv(GL_LIGHT0, GL_POSITION, light_pos)

    # Обертання об'єктів навколо осі X
    glRotatef(angle, 1, 0, 0)

    # Відображення паралелепіпеда
    glPushMatrix()
    glColor3f(0.5, 0.5, 0.5)  # Сірий колір
```

```
    draw_prlpd()
    glPopMatrix()

    # Відображення сфери
    glPushMatrix()
    glColor3f(1.0, 0.5, 0.0)  # Помаранчевий колір
    glTranslatef(3, 0.5, 1)  # Центрування сфери навпроти однієї з граней паралелепіпеда
    glutSolidSphere(0.5, 50, 50)  # Зменшили розмір сфери
    glPopMatrix()

    glutSwapBuffers()

    angle += 1

    # Додаємо затримку в 0.01 секунди для плавної анімації
    time.sleep(0.01)

# Основний цикл
def main():
    glutInit(sys.argv)
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH)
    glutInitWindowSize(800, 600)
    glutInitWindowPosition(100, 100)
    glutCreateWindow(b"OpenGL Scene")
    glutDisplayFunc(display)
    glutIdleFunc(display)
    glutReshapeFunc(reshape)
    init()
    glutMainLoop()

# Головні виклики
if __name__ == "__main__":
    main()
```

Результат:

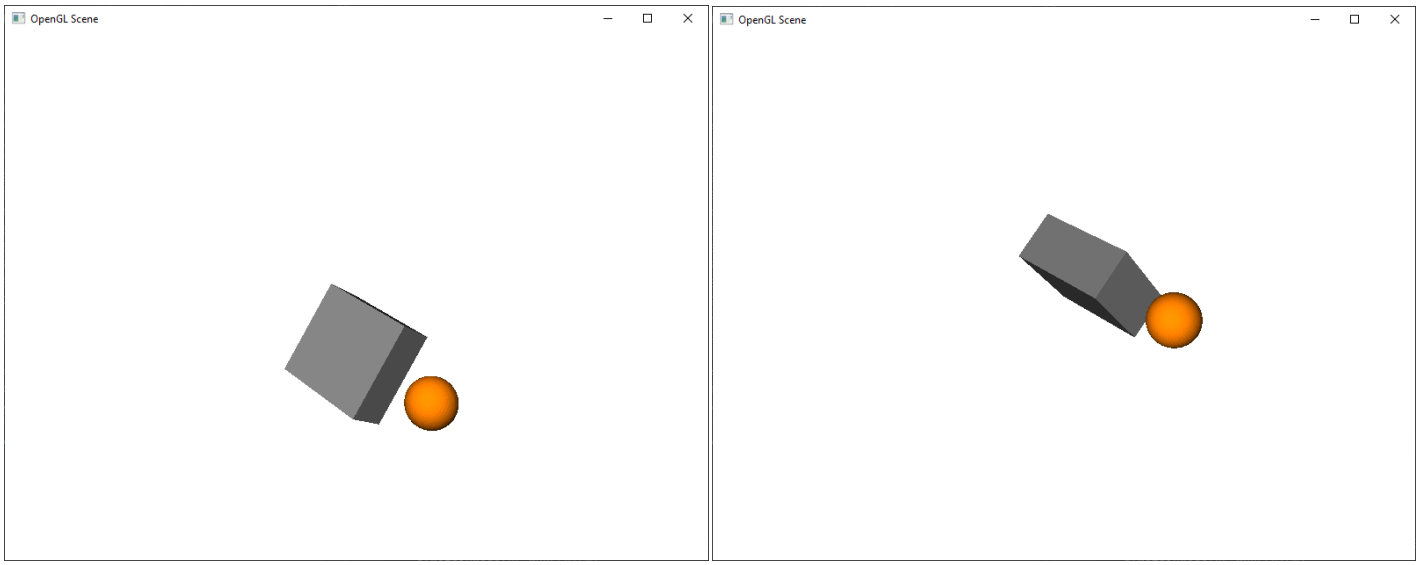

Рисунок 2 – Результат візуалізації сцени

У результаті програма **успішно візуалізує графічну сцену** з композицією паралелепіпеда та сфери.

2.2. Аналіз отриманих результатів

Обрано композицію: сірий паралелепіпед з лр 1 та помаранчева одинична сфера – підтверджено скриптом.

Розташовано джерело світла у точці (5, 5, 5) – підтверджено скриптом.

Розташовано камеру (спостерігача) за 8 одиниць від центру сцени – підтверджено скриптом.

Виконано анімацію обертання сцени – підтверджено рисунком.

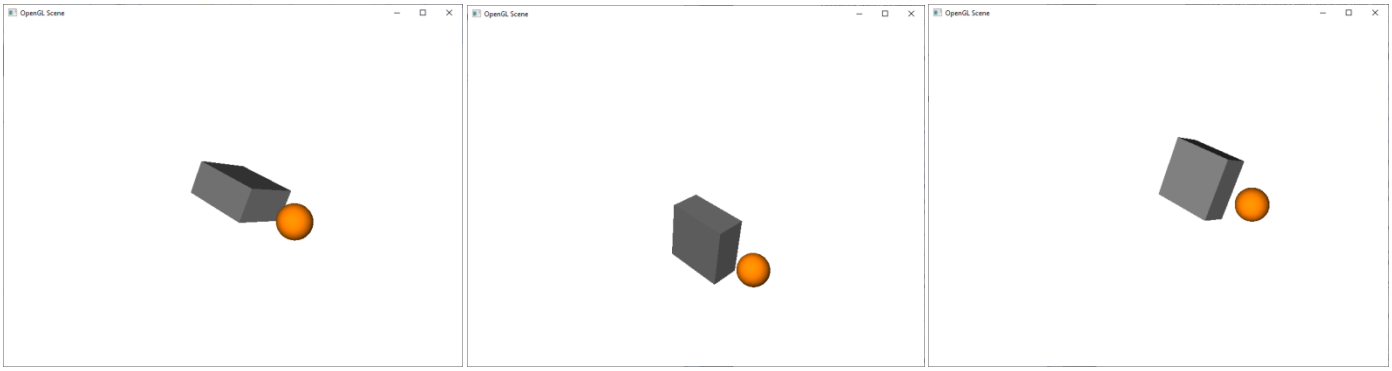

Рисунок 3 – Підтвердження виконання вимог 1

Висновок:

У результаті виконання лабораторної роботи отримано практичні та теоретичні навички роботи з бібліотекою OpenGL.

Реалізовано програмний скрипт, який реалізує реалістичну візуалізацію графічної сцени з композицією об’єктів, виконуючи анімацію обертання.

Результати роботи наведеного програмного скрипта в повній мірі задовольняють вимогам поставленого завдання та демонструють основні можливості OpenGL для створення графічних сцен.