**Міністерство освіти і науки України**

**Національний технічний університет України «КПІ» імені Ігоря Сікорського**

**Кафедра обчислювальної техніки ФІОТ**

**ЗВІТ**

**з лабораторної роботи №9**

**з навчальної дисципліни «Computer Vision»**

**Тема:**

**СИНТЕЗ РЕАЛІСТИЧНИХ ОБ’ЄКТІВ ДОПОВНЕНОЇ РЕАЛЬНОСТІ**

**Виконав:**

Студент 3 курсу кафедри ІПІ ФІОТ,

Навчальної групи ІП-11

Трикош І. В.

**Перевірив:**

Професор кафедри ОТ ФІОТ

Писарчук О.О.

**Київ 2024**

## **І. Мета:**

Дослідити методологію і технології створення доповненої реальності.

## **ІІ. Завдання:**

Розробити програмний скрипт мовою Python що реалізує створення об’єктів доповненої реальності:

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.

## **ІІІ. Результати виконання лабораторної роботи.**

* 1. **Синтезована математична модель**

Відповідно до умов задачі синтезовано математичну модель, що реалізує обертання та переміщення 3D фігур: куба і параболоїда при активному джерелі світла.

Еліптичний параболоїд описується формулою:

де та – коефіцієнти кривизни у площинах xz та yz відповідно.

Освітлення реалізується з допомогою моделі відбиття Фонга:

де – інтенсивність джерела випромінювання світла, – константа відбиття світла, – кут між відбитим від точки поверхні променем та вектором спостерігача, – степінь дзеркального відображення матеріалу.

Модель трьохвимірного переносу (переміщення) у точку :

де – точка, – матриця перетворення.

Обертання об’єкта навколо осі X:

де – кут повороту (в радіанах).

Обертання об’єкта навколо осі Y:

де – кут повороту (в радіанах).

* 1. **Результати архітектурного проектування та їх опис**

На рис. 1 зображена блок-схема до моделі до програмного скрипту:



Рисунок 1 – Блок схема алгоритму

Робота алгоритму розпочинається з ініціалізації параметрів програми. Сюди входить визначання розміру вікна, кольору фону, координат джерела світла тощо. Ці дії виконуються у блоці 1 блок-схеми алгоритму рис.1.

У блоці 2 виконується переміщення (перенос) куба з допомогою матриці перетворення.

У блоці 3 виконується обертання куба навколо осей X та Y.

У блоці 4 виконується відображення куба у вікні програми.

У блоці 5 відбувається переміщення (перенос) параболоїда.

У блоці 6 виконується обертання параболоїда навколо осі Y.

У блоці 7 відбувається відображення параболоїда у вікні програми. На цьому виконання алгоритму завершене.

* 1. **Опис структури проекту програми**

Для реалізації розробленого алгоритму мовою програмування Python з використанням можливостей інтегрованого середовища PyCharm сформовано проєкт.

Проєкт базується на лінійній бізнес-логіці функціонального програмування та має таку структуру.

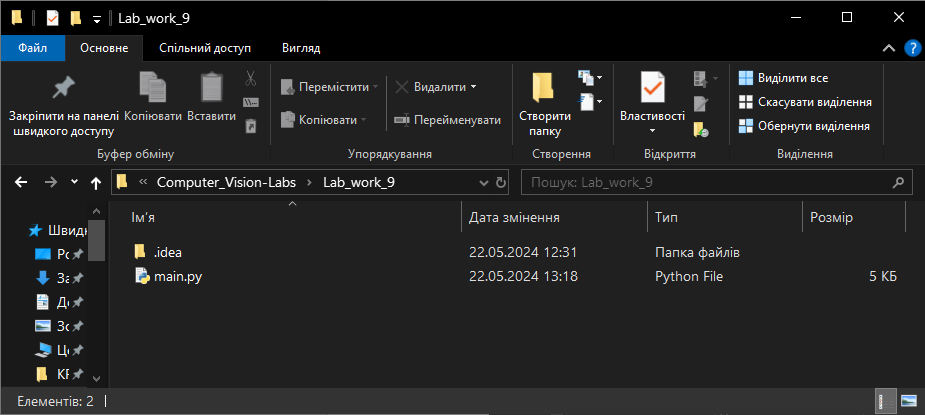


Рисунок 2 – Структура проєкту

Lab\_work\_9 – головний каталог проєкту

main.py – файл програмного коду

* 1. **Результати роботи програми відповідно до завдання (допускається у формі скриншотів)**

Результатом роботи програми є сукупність послідовності графічних вікон, що реалізують умови завдання лабораторної роботи.

1. На рисунках 3-6 зображено вікно виконання програми у різні періоди часу та різним рівнем освітленості:

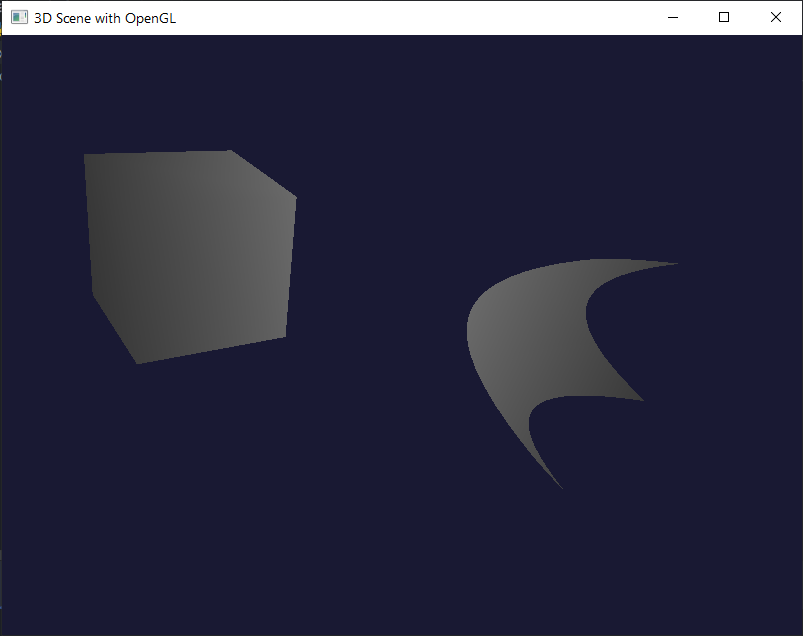


Рисунок 3 – Вікно виконання програми (обидва об’єкти освілені)

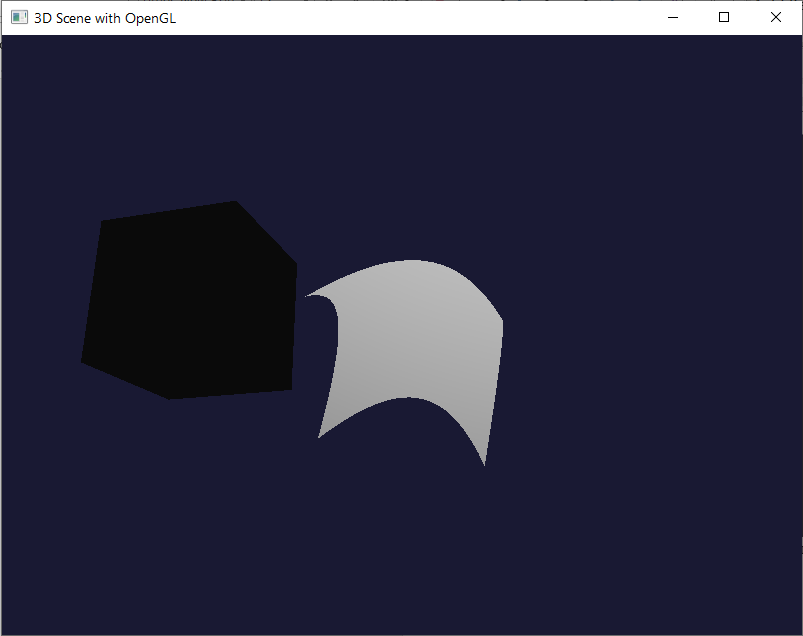


Рисунок 4 – Вікно виконання програми (освітлений лише параболоїд)

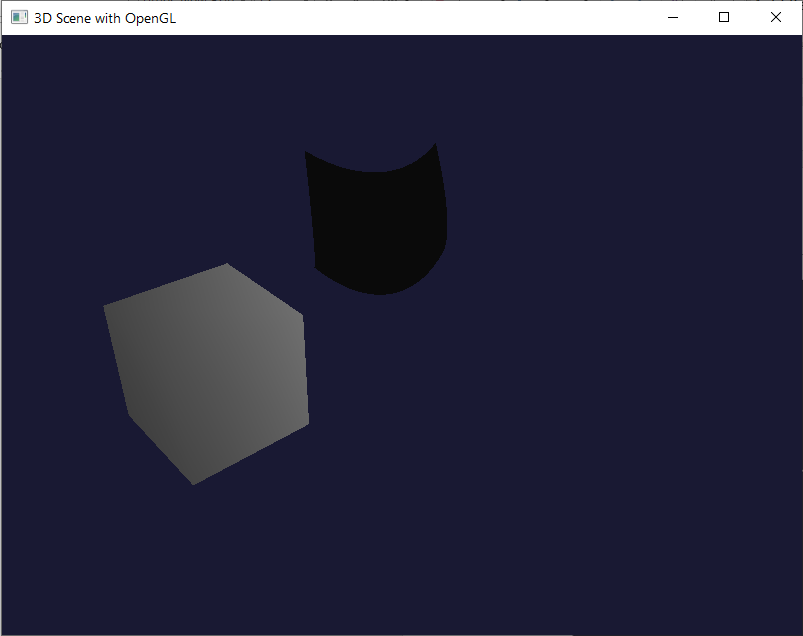


Рисунок 5 – Вікно виконання програми (освітлений лише куб)

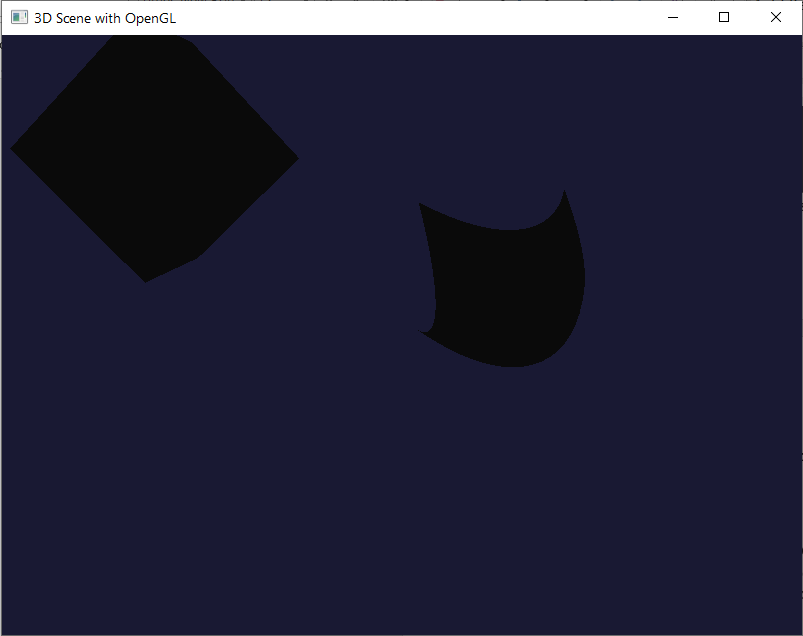


Рисунок 6 – Вікно виконання програми (обидва об’єкти неосвілені)

Бачимо, що обидва об’єкти добре переміщуються, обертаються і освітлюються в залежності від кута до джерела освітлення. Представлені результати відповідають завданню лабораторної роботи.

* 1. **Програмний код, що забезпечує отримання результату (допускається у формі скриншотів)**

Програмний код послідовно реалізує алгоритм на рис.1 та спрямований на отримання результатів, поданих на рис.3-6.

Для спрощення програмного коду і раціоналізації обчислень застосовано функціональні механізми створення підпрограм.

При цьому використано можливості Python бібліотек: numpy, OpenGL та sys.

Контекстні коментарі пояснюють сутність окремих скриптів наведеного коду програми.

main.py

*"""  
Переміщення та обертання з допомогою OpenGL:  
 3D багатокутника (куб)  
 3D моделі поверхні другого порядку (параболоїд)  
"""  
  
from* OpenGL.GL *import* \*  
*from* OpenGL.GLU *import* \*  
*from* OpenGL.GLUT *import* \*  
*import* sys  
*import* numpy *as* np  
  
*# Розмір вікна*width, height = 800, 600  
  
*# Параметри для повороту фігур*angle\_cube = 0  
angle\_surface = 0  
  
*# Параметр для руху об'єктів*time = 0.0  
  
  
*def* init():  
 *""" Ініціалізація параметрів OpenGL """* glClearColor(0.1, 0.1, 0.2, 1.0) *# Колір фону* glEnable(GL\_DEPTH\_TEST) *# Тест глибини* glEnable(GL\_LIGHTING) *# Освітлення* glEnable(GL\_LIGHT0) *# Джерело світла  
  
 # Налаштування джерела світла* light\_pos = [-10, -10, 10, 100]  
 glLightfv(GL\_LIGHT0, GL\_POSITION, light\_pos)  
  
 *# Налаштування матеріалів об'єктів* glMaterialfv(GL\_FRONT, GL\_AMBIENT, [0.2, 0.2, 0.2, 1.0])  
 glMaterialfv(GL\_FRONT, GL\_DIFFUSE, [0.8, 0.8, 0.8, 1.0])  
 glMaterialfv(GL\_FRONT, GL\_SPECULAR, [1.0, 1.0, 1.0, 1.0])  
 glMaterialf(GL\_FRONT, GL\_SHININESS, 50)  
  
  
*def* draw\_polygon():  
 *""" Малювання куба по гранях """* glBegin(GL\_QUADS)  
  
 glVertex3f(-1.0, -1.0, 1.0)  
 glVertex3f(1.0, -1.0, 1.0)  
 glVertex3f(1.0, 1.0, 1.0)  
 glVertex3f(-1.0, 1.0, 1.0)  
  
 glVertex3f(-1.0, -1.0, -1.0)  
 glVertex3f(1.0, -1.0, -1.0)  
 glVertex3f(1.0, 1.0, -1.0)  
 glVertex3f(-1.0, 1.0, -1.0)  
  
 glVertex3f(-1.0, -1.0, 1.0)  
 glVertex3f(-1.0, 1.0, 1.0)  
 glVertex3f(-1.0, 1.0, -1.0)  
 glVertex3f(-1.0, -1.0, -1.0)  
  
 glVertex3f(1.0, -1.0, 1.0)  
 glVertex3f(1.0, 1.0, 1.0)  
 glVertex3f(1.0, 1.0, -1.0)  
 glVertex3f(1.0, -1.0, -1.0)  
  
 glVertex3f(-1.0, -1.0, -1.0)  
 glVertex3f(1.0, -1.0, -1.0)  
 glVertex3f(1.0, -1.0, 1.0)  
 glVertex3f(-1.0, -1.0, 1.0)  
  
 glVertex3f(-1.0, 1.0, -1.0)  
 glVertex3f(1.0, 1.0, -1.0)  
 glVertex3f(1.0, 1.0, 1.0)  
 glVertex3f(-1.0, 1.0, 1.0)  
 glEnd()  
  
  
*def* draw\_surface():  
 *""" Малювання параболоїда """* glBegin(GL\_QUADS)  
 *for* x *in* np.arange(-1.0, 1.0, 0.1):  
 *for* y *in* np.arange(-1.0, 1.0, 0.1):  
 z = x \*\* 2 + y \*\* 2  
 glVertex3f(x, y, z)  
 glVertex3f(x + 0.1, y, (x + 0.1) \*\* 2 + y \*\* 2)  
 glVertex3f(x + 0.1, y + 0.1, (x + 0.1) \*\* 2 + (y + 0.1) \*\* 2)  
 glVertex3f(x, y + 0.1, x \*\* 2 + (y + 0.1) \*\* 2)  
 glEnd()  
  
  
*def* display():  
 *""" Відображення сцени """  
 global* angle\_cube, angle\_surface, time  
 glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)  
 glLoadIdentity()  
  
 *# Камера* gluLookAt(5, 5, 5, 0, 0, 0, 0, 1, 0)  
  
 *# Переміщення та обертання куба* glPushMatrix()  
 cube\_position = 2.0 \* np.sin(time)  
 glTranslatef(-3.0, cube\_position, 1.0)  
 glRotatef(angle\_cube, 1, 1, 0)  
 draw\_polygon()  
 glPopMatrix()  
  
 *# Переміщення та обертання параболоїда* glPushMatrix()  
 surface\_position = 1.5 \* np.sin(time)  
 glTranslatef(surface\_position, 0.0, -1.0)  
 glRotatef(angle\_surface, 0, 1, 0)  
 draw\_surface()  
 glPopMatrix()  
  
 glutSwapBuffers()  
 angle\_cube += 0.3 *# Швидкість обертання куба* angle\_surface += 0.5 *# Швидкість обертання поверхні другого порядку* time += 0.01 *# Зміна часу для руху об'єктів  
  
  
def* reshape(w, h):  
 *""" Налаштування сцени у вікні """* glViewport(0, 0, w, h)  
 glMatrixMode(GL\_PROJECTION)  
 glLoadIdentity()  
 gluPerspective(45, w / h, 1, 50)  
 glMatrixMode(GL\_MODELVIEW)  
 glLoadIdentity()  
  
  
*def* main():  
 *""" Запуск програми """* glutInit(sys.argv)  
 glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH)  
 glutInitWindowSize(width, height)  
 glutInitWindowPosition(100, 100)  
 glutCreateWindow(b"3D Scene with OpenGL")  
 init()  
 glutDisplayFunc(display)  
 glutIdleFunc(display)  
 glutReshapeFunc(reshape)  
 glutMainLoop()  
  
  
*if* \_\_name\_\_ == "\_\_main\_\_":  
 main()

## **IV. Висновки.**

У ході виконання лабораторної роботи було проведено дослідження методології і технології створення доповненої реальності. Для цього було реалізовано циклічне переміщення та обертання куба та параболоїда при активному джерелі світла з допомогою бібліотеки OpenGL.

Виконав: студент Трикош І. В.