МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра ІСМ



ЗВІТ

Про виконання лабораторної роботи №5

«Розробка ASCII ART генератора для візуалізації 3D-фігур»

З дисципліни

«Спеціалізовані мови програмування»

Студентки групи РІ – 31

Нащецької Яни Вікторівни

Прийняв викладач

Щербак С.С.

**Лабораторна робота № 5. Розробка ASCII ART генератора для візуалізації 3D-фігур**

**Мета:** Cтворення додатка для малювання 3D-фігур у ASCII-арті на основі об’єктно - орієнтованого підходу та мови Python

**План роботи**

Завдання 1: Проектування класів

Розробіть структуру класів для вашого генератора 3D ASCII-арту. Визначте основні компоненти, атрибути та методи, необхідні для програми.

Завдання 2: Введення користувача

Створіть методи у межах класу для введення користувача та вказання 3D-фігури, яку вони хочуть намалювати, та її параметрів (наприклад, розмір, кольори).

Завдання 3: Представлення фігури

Визначте структури даних у межах класу для представлення 3D-фігури. Це може включати використання списків, матриць або інших структур даних для зберігання форми фігури та її властивостей.

Завдання 4: Проектування з 3D в 2D

Реалізуйте метод, який перетворює 3D-представлення фігури у 2D-представлення, придатне для ASCII-арту.

Завдання 5: Відображення ASCII-арту

Напишіть метод у межах класу для відображення 2D-представлення 3D-фігури як ASCII-арту. Це може включати відображення кольорів і форми за допомогою символів ASCII.

Завдання 6: Інтерфейс, зрозумілий для користувача

Створіть зручний для користувача командний рядок або графічний інтерфейс користувача (GUI) за допомогою об'єктно-орієнтованих принципів, щоб дозволити користувачам спілкуватися з програмою.

Завдання 7: Маніпуляція фігурою

Реалізуйте методи для маніпулювання 3D-фігурою, такі масштабування або зміщення, щоб надавати користувачам контроль над її виглядом.

Завдання 8: Варіанти кольорів

Дозвольте користувачам вибирати варіанти кольорів для їхніх 3D ASCII-арт-фігур. Реалізуйте методи для призначення кольорів різним частинам фігури.

Завдання 9: Збереження та експорт

Додайте функціональність для зберігання згенерованого 3D ASCII-арту у текстовий файл

Завдання 10: Розширені функції

Розгляньте можливість додавання розширених функцій, таких як тінь, освітлення та ефекти перспективи, для підвищення реалізму 3D ASCII-арту.

**Основний код програми:**

import math

from shared.classes.art\_generator.ascii\_art\_generator import ArtGenerator

class Shape3D:

    def \_\_init\_\_(self, size=5):

        self.size = size

    def draw(self):

        raise NotImplementedError("Subclasses should implement this!")

class Cube(Shape3D):

    def \_\_init\_\_(self, size=5, symbol="#"):

        super().\_\_init\_\_(size)

        self.symbol = symbol

    def draw(self):

        width = self.size \* 2

        height = self.size + 5

        return draw\_cube(width, height, self.symbol)

def draw\_cube(width, height, symbol):

    cube = [[" "] \* width for \_ in range(height)]

    vertices = {

        "tc": (width // 2, 0),

        "tl": (0, int(0.25 \* height)),

        "tr": (width - 1, int(0.25 \* height)),

        "cc": (width // 2, int(0.5 \* height)),

        "bl": (0, int(0.75 \* height)),

        "br": (width - 1, int(0.75 \* height)),

        "bc": (width // 2, height - 1),

    }

    edges = (

        ("tc", "tl"),

        ("tc", "tr"),

        ("tl", "cc"),

        ("tr", "cc"),

        ("bl", "bc"),

        ("br", "bc"),

        ("cc", "bc"),

    )

    for edge in edges:

        v1 = vertices[edge[0]]

        v2 = vertices[edge[1]]

        x1, y1 = v1

        x2, y2 = v2

        if x1 > x2:

            x1, x2 = x2, x1

            y1, y2 = y2, y1

        try:

            m = (y2 - y1) / (x2 - x1)

        except ZeroDivisionError:

            for yy in range(min(y1, y2), max(y1, y2) + 1):

                cube[yy][x1] = symbol

        else:

            yy = y1

            for xx in range(x1, x2 + 1):

                cube[int(round(yy))][xx] = symbol

                yy += m

    x1, y1 = vertices["tl"]

    x2, y2 = vertices["bl"]

    for yy in range(min(y1, y2), max(y1, y2) + 1):

        cube[yy][x1] = "|"

    x1, y1 = vertices["tr"]

    x2, y2 = vertices["br"]

    for yy in range(min(y1, y2), max(y1, y2) + 1):

        cube[yy][x1] = "|"

    for xx in range(vertices["tl"][0], vertices["tr"][0] + 1):

        cube[vertices["tc"][1]][xx] = symbol

        cube[vertices["bc"][1]][xx] = symbol

    cube\_str = "\n".join("".join(row) for row in cube)

    return cube\_str

class Sphere(Shape3D):

    def \_\_init\_\_(self, size, symbol="\*", shadow\_symbol=".", shadow\_offset=2):

        super().\_\_init\_\_(size)

        self.symbol = symbol

        self.shadow\_symbol = shadow\_symbol

        self.shadow\_offset = shadow\_offset

    def draw(self):

        output = []

        for i in range(-self.size, self.size + 1):

            row = []

            for j in range(-self.size, self.size + 1):

                distance = math.sqrt(i\*\*2 + j\*\*2)

                if distance <= self.size:

                    shadow\_intensity = int((distance / self.size) \* 10)

                    if shadow\_intensity < 4:

                        row.append(self.symbol)

                    else:

                        if j > 0:

                            row.append(self.shadow\_symbol)

                        else:

                            row.append(self.shadow\_symbol)

                else:

                    row.append(" ")

            output.append("".join(row).rstrip())

        return "\n".join(output)

class Art3DGenerator(ArtGenerator):

    def \_\_init\_\_(

        self, shape: Shape3D, symbol="\*", width=40, height=10, color\_mode="bw"

    ):

        super().\_\_init\_\_("", symbol, width, height, "center", color\_mode)

        self.shape = shape

    def apply\_color(self, text):

        if self.color\_mode == "gray":

            return "\033[90m" + text + "\033[0m"

        return text

    def generate\_3d\_art(self):

        ascii\_art = self.apply\_color(ascii\_art)

        ascii\_art = self.shape.draw()

        self.text = ascii\_art

        return ascii\_art

**Висновок:** За допомогою цієї лабораторної роботи я створила високорівневий об'єктно-орієнтований генератор 3D ASCII-арту, який дозволить користувачам проектувати, відображати та маніпулювати 3D-фігурами в ASCII-арті. Цей проект надав мені глибоке розуміння об'єктно-орієнтованого програмування і алгоритмів графіки, та сприяв творчому підходу до створення ASCII-арту.