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»

Студента групи РІ – 31

Тураша Івана Павловича

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:**

from colorama import Fore

class Cube:

    def \_\_init\_\_(self, size, color):

        self.size = size

        self.color = color

    def draw(self):

        colors = {

            'red': Fore.RED,

            'green': Fore.GREEN,

            'blue': Fore.BLUE,

            'yellow': Fore.YELLOW,

            'magenta': Fore.MAGENTA,

            'cyan': Fore.CYAN,

            'white': Fore.WHITE,

            'black': Fore.BLACK

        }

        cube\_color = colors.get(self.color.lower(), Fore.WHITE)

        # ASCII Cube drawing logic

        t = v = h = int(self.size / 2)

        s, p, b, f, n = " ", cube\_color + "+", cube\_color + "|", cube\_color + "/", "\n"

        l = p + (cube\_color + "-") \* (t \* 4) + p

        S = s \* (4 \* t)

        k = s \* h

        K = b + S + b

        r = (s \* t) + s + l + n

        while t:

            r += (s \* t) + f + (S + f + s \* (h - t) + b) + n

            t -= 1

        r += l + (k + b) + n + ((K + k + b + n) \* (v - 1)) + K + k + p + n

        while v:

            v -= 1

            r += K + (s \* v) + f + n

        r += l

        return r

    def resize(self, scaling\_factor):

        self.size = int(self.size \* scaling\_factor)

from colorama import Fore

class Square:

    def \_\_init\_\_(self, size, color):

        self.size = size  # Висота квадрата

        self.width = size \* 2  # Ширина, яка відповідає передній грані куба

        self.color = color

    @staticmethod

    def from\_cube(cube):

        # Створюємо квадрат, який має ту ж висоту та подвійну ширину як передня грань куба

        return Square(size=cube.size, color=cube.color)

    def draw(self):

        colors = {

            'red': Fore.RED,

            'green': Fore.GREEN,

            'blue': Fore.BLUE,

            'yellow': Fore.YELLOW,

            'magenta': Fore.MAGENTA,

            'cyan': Fore.CYAN,

            'white': Fore.WHITE,

            'black': Fore.BLACK

        }

        square\_color = colors.get(self.color.lower(), Fore.WHITE)

        square = ""

        # Верхній рядок квадрата

        square += square\_color + "+" + "-" \* self.width + "+" + "\n"

        # Бокові сторони квадрата

        for \_ in range(self.size):

            square += square\_color + "|" + " " \* self.width + "|" + "\n"

        # Нижній рядок квадрата

        square += square\_color + "+" + "-" \* self.width + "+"

        return square

import sys

from business\_layer.cube import Cube

from business\_layer.square import Square

from persistent\_layer.file\_storage import FileStorage

class CommandLineInterface:

    def \_\_init\_\_(self):

        self.scene = []

        self.last\_cube = None

    def run(self):

        print("Welcome to the ASCII Art Generator!")

        while True:

            command = input("Enter a command (create/render/save/resize/exit): ").lower()

            if command == "exit":

                print("Exiting the program.")

                sys.exit(0)

            elif command == "create":

                self.create\_shape()

            elif command == "render":

                self.render\_scene()

            elif command == "save":

                self.save\_to\_file()

            elif command == "resize":

                self.resize\_shape()

            else:

                print("Invalid command. Please enter 'create', 'render', 'save', 'resize', or 'exit'.")

    def create\_shape(self):

        shape\_type = input("Enter shape type (cube/square): ").lower()

        if shape\_type == "cube":

            self.create\_cube()

        elif shape\_type == "square":

            self.create\_square()

        else:

            print("Invalid shape type. Please enter 'cube' or 'square'.")

    def create\_cube(self):

        try:

            size = int(input("Enter the size of the cube: "))

            color = input("Enter the color of the cube (e.g., red, green, blue): ").lower()

            cube = Cube(size, color)

            self.scene.append(cube)

            self.last\_cube = cube

            print("Cube created.")

        except ValueError:

            print("Invalid input for size. Please enter a numeric value.")

    def create\_square(self):

        if not self.last\_cube:

            print("No cube exists to create a square from. Please create a cube first.")

            return

        square = Square.from\_cube(self.last\_cube)

        self.scene.append(square)

        print("Square created as the front face of the last cube.")

    def render\_scene(self):

        if not self.scene:

            print("No shapes to render. Please create shapes first.")

            return

        for shape in self.scene:

            print(shape.draw())

            print("\n")

    def resize\_shape(self):

        if not self.scene:

            print("No shape to resize. Please create a shape first.")

            return

        try:

            scaling\_factor = float(input("Enter the scaling factor for the current shape: "))

            current\_shape = self.scene[-1]

            current\_shape.resize(scaling\_factor)

            print("Shape resized.")

        except ValueError:

            print("Invalid input. Please enter a numeric scaling factor.")

    def save\_to\_file(self):

        file\_name = input("Enter the file name to save the ASCII art (include .txt extension): ")

        FileStorage.save\_to\_file(self.scene, file\_name)

**Висновок:** За допомогою цієї лабораторної роботи я створив 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.