Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра ІСМ



**Звіт**

до лабораторної роботи № 5

з дисципліни

​*Спеціалізовані мови програмування*

на тему:

“**Розробка ASCII ART генератора**

**для візуалізації 3D-фігур**”

Виконав студент РІ-31

**Лазенко Юрій**

Прийняв: Щербак С.С.

Львів – 2024

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

**Завдання лабораторної роботи**

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.

Файл запуску:

import sys

from cube\_renderer import CubeRenderer

from scene import Scene

from square import Square

from colorama import Fore,Style

class CommandLineInterface:

    def \_\_init\_\_(self):

        self.scene = Scene()

    def run(self):

        print("Welcome to the 3D ASCII Art Generator!")

        while True:

            Scene.display\_menu()

            command = input(f"{Style.BRIGHT}{Fore.YELLOW}Enter a command: {Style.RESET\_ALL}").lower()

            if command == "exit":

                sys.exit(0)

            elif command == "render":

                self.scene.render\_scene()

            elif command == "cube":

                self.create\_cube()

            elif command == "square":

                self.create\_square()

                self.scene.render\_one()

            elif command == "size":

                self.set\_size()

                self.scene.render\_one()

            elif command == "resize":

                self.resize\_shape()

                self.scene.render\_one()

            elif command == "color":

                self.set\_color()

                self.scene.render\_one()

            elif command == "save":

                self.save\_to\_file()

            else:

                print("Invalid command. Please try again.")

    def create\_cube(self):

        size\_input = input("Enter the size of the cube: ")

        try:

            size = float(size\_input)

        except ValueError:

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

            return

        color = input("Enter the color of the cube (e.g., red, green, blue, yellow, magenta, cyan, white, black ): ")

        cube = CubeRenderer(size, color)

        self.scene.add\_shape(cube)

        print("Cube created.")

        print(cube.start\_rendering())

    def create\_square(self):

        size\_input = input("Enter the size of the square: ")

        try:

            size = float(size\_input)

        except ValueError:

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

            return

        color = input("Enter the color of the square (e.g., red, green, blue, yellow, magenta, cyan, white, black ): ")

        square = Square(size, color)

        self.scene.add\_shape(square)

        print("Square created.")

    def set\_size(self):

        size = float(input("Enter the size for the current shape: "))

        if not self.scene.shapes:

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

            return

        current\_shape = self.scene.shapes[-1]

        current\_shape.size = size

    def resize\_shape(self):

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

        if not self.scene.shapes:

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

            return

        current\_shape = self.scene.shapes[-1]

        current\_shape.size \*= scaling\_factor

    def set\_color(self):

        if not self.scene.shapes:

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

            return

        color = input(

            "Enter the color for the current shape (e.g., red, green, blue, yellow, magenta, cyan, white, black): ")

        current\_shape = self.scene.shapes[-1]

        current\_shape.color = color

        print(f"Color set to {color} for the current shape.")

    def save\_to\_file(self):

        if not self.scene.shapes:

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

            return

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

        try:

            with open(file\_name, "w") as file:

                for shape in self.scene.shapes:

                    ascii\_art = shape.draw()

                    file.write(ascii\_art + "\n\n")

            print(f"ASCII art saved to {file\_name}.")

        except Exception as e:

            print(f"Error saving to file: {e}")

def main():

    cli = CommandLineInterface()

    cli.run()

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**Висновок**

Виконавши ці завдання, 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, сприяв творчому підходу до створення ASCII-арту.

GitHub: [smp3/SMP-main/Lab5 at main · 1azenkoSS/smp3](https://github.com/1azenkoSS/smp3/tree/main/SMP-main/Lab5)