Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Лабораторна роботи №5

з дисципліни «Спеціалізовані мови програмування»

на тему

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

Виконав:

Костюк В. Ю.

Перевірив:

Щербак С. С.

Львів 2024

**Мета:** Створення 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.

**Реалізація:**

**Файл file\_handler.py:**  
def save\_to\_file(filename, shape):

    with open(filename, 'w') as f:

        for line in shape.get\_2d\_representation():

            f.write(line + "\n")

**Файл shape3d.py:**

from colorama import Fore, Style, init

# Ініціалізація colorama

init()

class Shape3D:

    def \_\_init\_\_(self, color):

        self.color = color

    def get\_2d\_representation(self):

        raise NotImplementedError("This method should be overridden by subclasses")

    def display\_ascii\_art(self):

        representation = self.get\_2d\_representation()

        color\_code = self.get\_color\_code()

        for line in representation:

            print(color\_code + line + Style.RESET\_ALL)

    def get\_color\_code(self):

        return Fore.WHITE  # Можна реалізувати логіку для інших кольорів

class Cube(Shape3D):

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

        super().\_\_init\_\_(color)

        self.size = size

    def get\_2d\_representation(self):

        cube\_lines = [

            " " \* self.size + "+" + "-" \* self.size + "+",

            " " \* self.size + "/" + " " \* self.size + "/|",

            " " \* self.size + "+" + "-" \* self.size + "+ |",

            " " \* self.size + "|" + " " \* self.size + "|" + "+",

            " " \* self.size + "|" + " " \* self.size + "|/",

            " " \* self.size + "+" + "-" \* self.size + "+"

        ]

        return cube\_lines + self.get\_shadow\_representation()

    def get\_shadow\_representation(self):

        shadow\_lines = [

            " " \* (self.size + 1) + "#" \* self.size + " ",

            " " \* (self.size + 1) + "#" \* self.size + " ",

            " " \* (self.size + 1) + "#" \* self.size + " "

        ]

        return shadow\_lines

    def scale(self, factor):

        self.size = max(1, int(self.size \* factor))

    def translate(self, offset):

        pass  # Реалізуйте логіку для зберігання координат, якщо потрібно

**Папка functions, файл functions.py:**from AppSettings.AppSettings import COLORS

def get\_shape\_type():

shapes = ["cube", "pyramid"]

print("Доступні фігури:")

for i, shape in enumerate(shapes, 1):

print(f"{i}. {shape}")

while True:

choice = input("Виберіть фігуру: ")

try:

return shapes[int(choice) - 1]

except (IndexError, ValueError):

print("Помилка: Виберіть правильний номер фігури.")

def get\_shape\_dimension():

while True:

dimension = input("Якщо 3D - введіть 1, якщо 2D - введіть 2,: ").strip().lower()

if dimension in ['1', '2']:

return dimension

print("Неправильний ввід. Спробуйте ще раз.")

def get\_color\_choice(colors):

while True:

print("Виберіть колір:")

for num, color in colors.items():

print(f"{num}: {color}")

try:

color\_choice = int(input("Введіть номер кольору: "))

if color\_choice in colors:

return colors[color\_choice]

else:

print("Помилка: Виберіть номер кольору з наявних.")

except ValueError:

print("Помилка: Введіть число.")

def get\_validated\_custom\_char():

while True:

custom\_char = input("Введіть символ для ASCII-арту (Enter для використання #): ")

if not custom\_char:

return '#'

if len(custom\_char) == 1:

return custom\_char

print("Помилка: Ви повинні ввести рівно один символ.")

def get\_save\_choice():

while True:

save\_choice = input("Бажаєте зберегти ASCII-арт у файл? (так/ні): ").strip().lower()

if save\_choice in ["так", "ні"]:

return save\_choice

print("Помилка: Введіть 'так' або 'ні'.")

**Файл user\_input.py:**

def get\_user\_input():

    shape\_type = input("Введіть тип фігури (cube): ")

    size = int(input("Введіть розмір фігури: "))

    color = input("Введіть колір фігури (white, red, green, blue): ").lower()

    return shape\_type, size, color

**Файл main.py:**

from lab5.shape3d import Cube

from lab5.user\_input import get\_user\_input

from lab5.file\_handler import save\_to\_file

from colorama import init, Fore

# Ініціалізація colorama

init()

def get\_color\_code(color):

    color\_map = {

        'white': Fore.WHITE,

        'red': Fore.RED,

        'green': Fore.GREEN,

        'blue': Fore.BLUE,

        # Додайте більше кольорів за потреби

    }

    return color\_map.get(color, Fore.WHITE)  # За замовчуванням білий

def main():

    shape\_type, size, color = get\_user\_input()

    color\_code = get\_color\_code(color)

    if shape\_type == 'cube':

        shape = Cube(size, color\_code)

        shape.display\_ascii\_art()

        # Додайте можливість масштабування

        scale\_option = input("Бажаєте змінити розмір фігури? (yes/no): ")

        if scale\_option.lower() == 'yes':

            factor = float(input("Введіть коефіцієнт масштабування: "))

            shape.scale(factor)

            shape.display\_ascii\_art()

        save\_option = input("Чи хочете ви зберегти результат у файл? (yes/no): ")

        if save\_option.lower() == 'yes':

            filename = input("Введіть ім'я файлу: ")

            save\_to\_file(filename, shape)

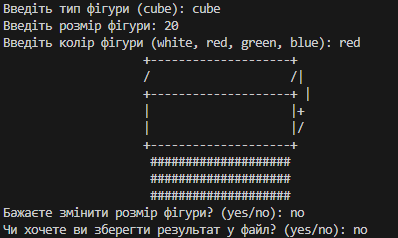
    else:

        print("Невідомий тип фігури.")

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

    main()

**Результат виконання:**



**Висновок:** на цій лабораторній роботі я створив високорівневий об'єктно-орієнтований генератор 3D ASCII-арту, який дозволить користувачам проектувати, відображати та маніпулювати 3D-фігурами в ASCII-арті