Міністерство освіти і науки України

Національний університет «Львівська політехніка»



**ЗВІТ**

про виконання лабораторної роботи № 5

з курсу:

«Спеціалізовані мови програмування»

**Виконав:**

студент гр. ІТ-31

Роман ХРАПЧУН

**Прийняв:**

Сергій ЩЕРБАК

Львів 2023

**Тема:** Розробка ASCII ART генератора для візуалізації 3D-фігур

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

**Хід 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.

**Реалізований код:**

from termcolor import colored

from abc import ABC, abstractmethod

import os

class Figure:

"""

Базовий клас для представлення геометричних фігур.

Attributes:

size (int): Розмір фігури.

color (str): Колір фігури.

Methods:

setSize(size: int): Встановлює розмір фігури.

setColor(color: str): Встановлює колір фігури.

generateFigure(): Абстрактний метод для генерації фігури.

"""

def \_\_init\_\_(self):

self.size = 1

self.color = 'green'

def setSize(self, size):

"""Встановлює розмір фігури."""

self.size = size

def setColor(self, color):

"""Встановлює колір фігури."""

self.color = color

def generateFigure(self):

"""Абстрактний метод для генерації фігури."""

return ""

def paintText(text, color):

"""

Функція для забарвлення тексту в заданий колір.

Args:

text (str): Вхідний текст.

color (str): Колір для забарвлення.

Returns:

str: Забарвлений текст.

"""

if len(color):

painted = colored(text, color)

return painted

else:

return text

def textFileSaver(filename, text):

"""Функція для збереження тексту у файл у папці Data."""

# Отримуємо шлях до поточної директорії, де виконується скрипт

current\_dir = os.path.dirname(os.path.abspath(\_\_file\_\_))

# Виходимо на рівень вище до папки Core

parent\_dir = os.path.dirname(current\_dir)

# Вказуємо шлях до папки Data, яка знаходиться на тому ж рівні, що й Core

data\_dir = os.path.join(parent\_dir, 'Data')

# Створюємо шлях до файлу в папці Data

file\_path = os.path.join(data\_dir, filename)

# Зберігаємо текст у файл

with open(file\_path, 'w') as file:

file.write(text)

print(f"Файл збережено у {file\_path}")

class Command(ABC):

"""

Абстрактний клас для представлення команди.

Methods:

execute(): Абстрактний метод для виконання команди.

"""

@abstractmethod

def execute(self):

pass

class Square(Figure):

"""

Клас для представлення квадрата.

Methods:

generateSquare(): Генерує квадрат з врахуванням розміру.

generateFigure(): Перевизначений метод для генерації квадрата з врахуванням кольору.

"""

def \_\_init\_\_(self):

super().\_\_init\_\_()

def generateSquare(self):

"""Генерує квадрат з врахуванням розміру."""

n = self.size

line = ''

if n > 1:

line += n \* 2 \* '\_'

else:

line += '\_'

top\_line = ' ' + line

bottom\_line = '|' + line + '|' + '\n'

inner\_spaces = n \* 2 \* ' '

inner\_fill = '|' + inner\_spaces + '|'

content = ''

i = 1

while (i < n):

content += inner\_fill + '\n'

i += 1

square = top\_line + '\n' + content + bottom\_line

return square

def generateFigure(self):

"""Перевизначений метод для генерації квадрата з врахуванням кольору."""

generated = self.generateSquare()

colored = paintText(generated, self.color)

return colored

class Cube(Figure):

"""

Клас для представлення куба.

Methods:

generateCube(): Генерує куб з врахуванням розміру.

generateFigure(): Перевизначений метод для генерації куба з врахуванням кольору.

"""

def \_\_init\_\_(self):

super().\_\_init\_\_()

def generateCube(self):

"""Генерує куб з врахуванням розміру."""

n = self.size

line = ''

if n > 1:

line += n \* 2 \* 'X'

else:

line += 'X'

top\_line = n \* ' ' + line

mid\_line\_space = 2 \* n - 2

mid\_line = '/' + line + '/' + mid\_line\_space \* ' ' + '\\'

bottom\_line = (n - 1) \* ' ' + '\\' + line + '\/'

inner\_spaces = n \* 2 \* ' '

inner\_top = '/' + inner\_spaces + '/'

inner\_bottom = '\\' + inner\_spaces + '\\'

top\_content = ''

bottom\_content = ''

for i in range(1, n):

top\_content += (n - i) \* ' ' + inner\_top + \

(i - 1) \* 2 \* ' ' + '\\' + '\n'

bottom\_content += (i - 1) \* ' ' + inner\_bottom + \

(n - i) \* 2 \* ' ' + '/' + '\n'

cube = top\_line + '\n' + top\_content + mid\_line + \

'\n' + bottom\_content + bottom\_line + '\n'

return cube

def generateFigure(self):

"""Перевизначений метод для генерації куба з врахуванням кольору."""

generated = self.generateCube()

colored = paintText(generated, self.color)

return colored

class Generate3DFigureCommand(Command):

"""

Команда для генерації 3D фігури через інтерфейс.

Attributes:

figure\_interface (FigureArtInterface): Інтерфейс фігури, через який відбувається генерація.

Methods:

execute(): Виконує команду генерації 3D фігури.

"""

def \_\_init\_\_(self, figure\_interface):

self.figure\_interface = figure\_interface

def execute(self):

"""Виконує команду генерації 3D фігури."""

print(self.figure\_interface.generate3dFigure())

class SetSizeCommand(Command):

"""

Команда для встановлення розміру фігури через інтерфейс.

Attributes:

figure\_interface (FigureArtInterface): Інтерфейс фігури, для якого встановлюється розмір.

new\_size (int): Новий розмір фігури.

Methods:

execute(): Виконує команду встановлення розміру фігури.

"""

def \_\_init\_\_(self, figure\_interface, new\_size):

self.figure\_interface = figure\_interface

self.new\_size = new\_size

def execute(self):

"""Виконує команду встановлення розміру фігури."""

self.figure\_interface.setSize(self.new\_size)

class SetColorCommand(Command):

"""

Команда для встановлення кольору фігури через інтерфейс.

Attributes:

figure\_interface (FigureArtInterface): Інтерфейс фігури, для якого встановлюється колір.

new\_color (str): Новий колір фігури.

Methods:

execute(): Виконує команду встановлення коліру фігури.

"""

def \_\_init\_\_(self, figure\_interface, new\_color):

self.figure\_interface = figure\_interface

self.new\_color = new\_color

def execute(self):

"""Виконує команду встановлення коліру фігури."""

self.figure\_interface.setColor(self.new\_color)

class FigureArtInterface(Figure):

"""

Клас інтерфейсу для взаємодії з геометричними фігурами.

Attributes:

type (str): Тип фігури.

left\_padding (int): Лівий відступ.

top\_padding (int): Верхній відступ.

bottom\_padding (int): Нижній відступ.

commands (dict): Словник команд для взаємодії з інтерфейсом.

Methods:

setType(type: str): Встановлює тип фігури.

setPaddings(left\_padding: int, top\_padding: int, bottom\_padding: int): Встановлює відступи фігури.

setPrimaryData(): Запитує користувача про основні дані фігури.

generateWithLeftPadding(text: str): Генерує текст з лівим відступом.

generateWithTopPadding(text: str): Генерує текст з верхнім відступом.

generateWithBottomPadding(text: str): Генерує текст з нижнім відступом.

generateWithPaddings(text: str): Генерує текст з врахуванням усіх відступів.

generate3dFigure(): Генерує 3D фігуру відповідно до вказаного типу.

generate2dFigure(): Генерує 2D фігуру відповідно до вказаного типу.

saveToFile2d(): Записує 2D фігуру в файл.

saveToFile3d(): Записує 3D фігуру в файл.

show\_menu(): Виводить меню для взаємодії з інтерфейсом.

loopMenu(): Запускає цикл взаємодії з інтерфейсом.

launch(): Запускає інтерфейс.

"""

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.type = "cube"

self.left\_padding = 5

self.top\_padding = 5

self.bottom\_padding = 5

self.commands = {

1: Generate3DFigureCommand(self),

2: SetSizeCommand(self, 0),

3: SetColorCommand(self, "")

}

def setType(self, type):

"""Встановлює тип фігури."""

self.type = type

def setPaddings(self, left\_padding, top\_padding, bottom\_padding):

"""Встановлює відступи фігури."""

self.left\_padding = left\_padding

self.top\_padding = top\_padding

self.bottom\_padding = bottom\_padding

def setPrimaryData(self):

"""Запитує користувача про основні дані фігури."""

size = int(input("Розмір фігури: "))

self.setSize(size)

color = input(

"Колір фігури (blue, green, red, magenta, yellow, white, cyan): ")

self.setColor(color)

type = input("Тип фігури (є тільки 'куб')(пропустити для типу за замовчуванням): ")

if len(type):

self.setType(type)

def generateWithLeftPadding(self, text):

"""Генерує текст з лівим відступом."""

lines = text.split('\n')

padded\_lines = [f"{' ' \* self.left\_padding}{line}" for line in lines]

return '\n'.join(padded\_lines)

def generateWithTopPadding(self, text):

"""Генерує текст з верхнім відступом."""

space = self.top\_padding \* '\n'

padded\_lines = space + text

return padded\_lines

def generateWithBottomPadding(self, text):

"""Генерує текст з нижнім відступом."""

space = self.bottom\_padding \* '\n'

padded\_lines = text + space

return padded\_lines

def generateWithPaddings(self, text):

"""Генерує текст з врахуванням усіх відступів."""

return self.generateWithBottomPadding(self.generateWithTopPadding(self.generateWithLeftPadding(text)))

def generate3dFigure(self):

"""Генерує 3D фігуру відповідно до вказаного типу."""

if self.type == 'куб':

cube = Cube()

cube.setSize(self.size)

cube.setColor(self.color)

return self.generateWithPaddings(cube.generateFigure())

return super().generateFigure()

def generate2dFigure(self):

"""Генерує 2D фігуру відповідно до вказаного типу."""

if self.type == 'куб':

square = Square()

square.setSize(self.size)

square.setColor(self.color)

return self.generateWithPaddings(square.generateFigure())

return super().generateFigure()

def saveToFile2d(self):

"""Записує 2D фігуру в файл."""

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

textFileSaver(

filename, self.generate2dFigure())

def saveToFile3d(self):

"""Записує 3D фігуру в файл."""

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

textFileSaver(

filename, self.generate3dFigure())

@staticmethod

def show\_menu():

"""Виводить меню для взаємодії з інтерфейсом."""

print("Виберіть пункт меню")

print("1 - Згенерувати 3D фігуру")

print("2 - Встановити розмір")

print("3 - Встановити колір (blue, green, red, magenta, yellow, white, cyan)")

print("4 - Встановити тип (куб)")

print("5 - Встановити відступи")

print("6 - Отримати 2D версію фігури")

print("7 - Зберегти у файл (3D)")

print("8 - Зберегти у файл (2D)")

print("0 - Вихід")

def loopMenu(self):

"""Запускає цикл взаємодії з інтерфейсом."""

while True:

self.show\_menu()

menu\_choice = int(input("Ключ меню: "))

if menu\_choice == 1:

print(self.generate3dFigure())

elif menu\_choice == 2:

new\_size = int(input("Введіть новий розмір: "))

self.setSize(new\_size)

elif menu\_choice == 3:

new\_color = input("Введіть новий колір: ")

self.setColor(new\_color)

elif menu\_choice == 4:

new\_type = input("Введіть новий тип: ")

self.setType(new\_type)

elif menu\_choice == 5:

left\_padding = int(input("Введіть лівий відступ: "))

top\_padding = int(input("Введіть верхній відступ: "))

bottom\_padding = int(input("Введіть нижній відступ: "))

self.setPaddings(left\_padding, top\_padding, bottom\_padding)

elif menu\_choice == 6:

print(self.generate2dFigure())

elif menu\_choice == 7:

self.saveToFile3d()

elif menu\_choice == 8:

self.saveToFile2d()

else:

break

def launch(self):

"""Запускає інтерфейс."""

self.setPrimaryData()

print(self.generate3dFigure())

self.loopMenu()

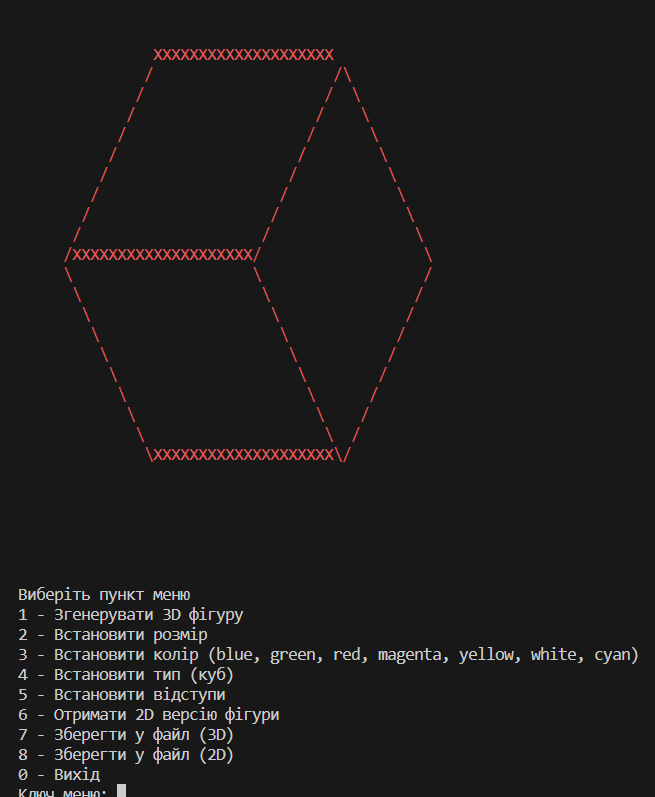
def run\_lab5():

interface = FigureArtInterface()

interface.launch()

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

run\_lab5()



*Рис. 1 – Результат виконання програми*

**Висновок:** На даній лабораторній роботі я створив універсальний Генератор 3D артів, який дозволить користувачам налаштовувати свої творіння з різними шрифтами, кольорами, розмірами та символами.