МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ



Лабораторна робота №5

з дисципліни “Спеціалізовані мови програмування”

на тему

“ Розробка ASCII ART генератора для візуалізації 3D-фігур ”

Виконав:

студент групи РІ-32

Мар’ян МІЛЯНЕЦЬ

Прийняв:

к.т.н.,

доц. кафедри ІСМ

Сергій ЩЕРБАК

Львів-2023

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

**План роботи**

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.

Програмний код:

import sys

from colorama import Fore, init

init(autoreset=True)

class Command:

def execute(self):

pass

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

}

# Set the selected color or default to white if the color is not found

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

t = v = h = int(self.size / 2) # top, vertical, horizontal, ініціалізація змінних

s, p, b, f, n = " ", cube\_color + "+", cube\_color + "|", cube\_color + "/", "\n" # s - space, p - plus, b - bar, f - forward slash, n - new line,

# символьні змінні для відображення куба

l = p + (cube\_color + "-") \* (t \* 4) + p # l - line, t - top, контур верхньої межі куба

S = s \* (4 \* t) # S - space, верхня частина куба

k = s \* h # k - space, вертикальна частина куба

K = b + S + b # K - bar, вертикальна частина куба

r = (s \* t) + s + l + n # r - result, t - top, l - line, n - new line, перший рядок куба

while t:

# t - top, v - vertical, h - horizontal, s - space, f - forward slash, b - bar, n - new line

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

t -= 1

# t - top, v - vertical, h - horizontal, s - space, f - forward slash, b - bar, n - new line

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

class ASCIIArtRenderer:

def render(self, command):

return command.execute()

class DrawCubeCommand(Command):

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

self.cube = cube

def execute(self):

return self.cube.draw()

class Scene:

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

self.shapes = []

def add\_shape(self, shape):

self.shapes.append(shape)

def render\_scene(self):

for shape in self.shapes:

print(shape.draw())

class CommandLineInterface:

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

self.scene = Scene()

def run(self):

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

while True:

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

if command == "exit":

sys.exit(0)

elif command == "create":

self.create\_cube()

elif command == "render":

self.scene.render\_scene()

elif command == "cube":

self.create\_cube()

elif command == "size":

self.set\_size()

elif command == "resize":

self.resize\_shape()

elif command == "color":

self.set\_color()

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 = Cube(size, color)

self.scene.add\_shape(cube)

print("Cube 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()

Висновки: Виконуючи 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.