МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра інформаційних систем та мереж

Лабораторна робота №5

з дисципліни

«Спеціалізовані мови приграмування»

на тему

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

Виконав:

ст. гр. РІ-32

Сергій БИХНЮК

Прийняв:

Сергій Щербак

Львів – 2024

**Мета роботи:**

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

**Завдання:**

**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.  
**Код програми:**

class AsciiArtRenderer:

ANSI\_COLORS = {

"black": "\033[30m", "red": "\033[31m", "green": "\033[32m",

"yellow": "\033[33m", "blue": "\033[34m", "magenta": "\033[35m",

"cyan": "\033[36m", "white": "\033[37m", "reset": "\033[0m"

}

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

self.shape = shape

self.color\_code = self.ANSI\_COLORS.get(self.shape.color, self.ANSI\_COLORS["white"])

def project\_to\_2d(self, point):

x, y, z = point

x\_2d = int((x - z) \* 0.5)

y\_2d = int((y - z) \* 0.5)

return x\_2d, y\_2d

def render\_ascii\_art(self):

points = self.shape.get\_points()

edges = self.shape.get\_edges()

canvas\_size = 40

canvas = [[' ' for \_ in range(canvas\_size)] for \_ in range(canvas\_size)]

projected\_points = []

for point in points:

x\_2d, y\_2d = self.project\_to\_2d(point)

x\_canvas = x\_2d + canvas\_size // 2

y\_canvas = y\_2d + canvas\_size // 2

if 0 <= x\_canvas < canvas\_size and 0 <= y\_canvas < canvas\_size:

canvas[y\_canvas][x\_canvas] = '#'

projected\_points.append((x\_canvas, y\_canvas))

for start, end in edges:

x1, y1 = projected\_points[start]

x2, y2 = projected\_points[end]

self.draw\_line(canvas, x1, y1, x2, y2)

print(self.color\_code)

for row in canvas:

print("".join(row))

print(self.ANSI\_COLORS["reset"])

def draw\_line(self, canvas, x1, y1, x2, y2):

dx = abs(x2 - x1)

dy = abs(y2 - y1)

sx = 1 if x1 < x2 else -1

sy = 1 if y1 < y2 else -1

err = dx - dy

while True:

if 0 <= x1 < len(canvas) and 0 <= y1 < len(canvas[0]):

canvas[y1][x1] = '#'

if x1 == x2 and y1 == y2:

break

e2 = err \* 2

if e2 > -dy:

err -= dy

x1 += sx

if e2 < dx:

err += dx

y1 += sy

from Shape3D import Shape3D

class Cube(Shape3D):

def \_\_init\_\_(self, size=5, position=(0, 0, 0), color="white"):

super().\_\_init\_\_(size, position, color)

def get\_points(self):

d = self.size / 2

x, y, z = self.position

return [

(x - d, y - d, z - d), (x + d, y - d, z - d),

(x + d, y + d, z - d), (x - d, y + d, z - d),

(x - d, y - d, z + d), (x + d, y - d, z + d),

(x + d, y + d, z + d), (x - d, y + d, z + d)

]

def get\_edges(self):

return [

(0, 1), (1, 2), (2, 3), (3, 0), # Нижній квадрат

(4, 5), (5, 6), (6, 7), (7, 4), # Верхній квадрат

(0, 4), (1, 5), (2, 6), (3, 7) # Вертикальні ребра

]

from UserInterface import UserInterface

def main5():

ui = UserInterface()

ui.get\_user\_input()

ui.display\_ascii\_art()

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

main5()

from Shape3D import Shape3D

class Pyramid(Shape3D):

def \_\_init\_\_(self, size=5, position=(0, 0, 0), color="white"):

super().\_\_init\_\_(size, position, color)

def get\_points(self):

d = self.size / 2

x, y, z = self.position

return [

(x - d, y - d, z), # Нижня ліва

(x + d, y - d, z), # Нижня права

(x + d, y + d, z), # Верхня права

(x - d, y + d, z), # Верхня ліва

(x, y, z + self.size) # Вершина піраміди

]

def get\_edges(self):

return [

(0, 1), (1, 2), (2, 3), (3, 0), # Основний квадрат

(0, 4), (1, 4), (2, 4), (3, 4) # Ребра до вершини

]

class Shape3D:

def \_\_init\_\_(self, size=5, position=(0, 0, 0), color="white"):

self.size = size

self.position = position

self.color = color

def get\_points(self):

raise NotImplementedError("Цей метод має бути реалізований у підкласі")

def get\_edges(self):

raise NotImplementedError("Цей метод має бути реалізований у підкласі")

from Cube import Cube

from Pyramid import Pyramid

from AsciiArtRenderer import AsciiArtRenderer

class UserInterface:

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

self.shape = None

def get\_user\_input(self):

shape\_type = input("Виберіть фігуру (cube/pyramid): ").strip().lower()

size = int(input("Введіть розмір фігури (рекомендується 5-10): ").strip())

color = input("Виберіть колір (black, red, green, yellow, blue, magenta, cyan, white): ").strip().lower()

if shape\_type == 'cube':

self.shape = Cube(size=size, color=color)

elif shape\_type == 'pyramid':

self.shape = Pyramid(size=size, color=color)

else:

print("Фігура не підтримується.")

def display\_ascii\_art(self):

if self.shape:

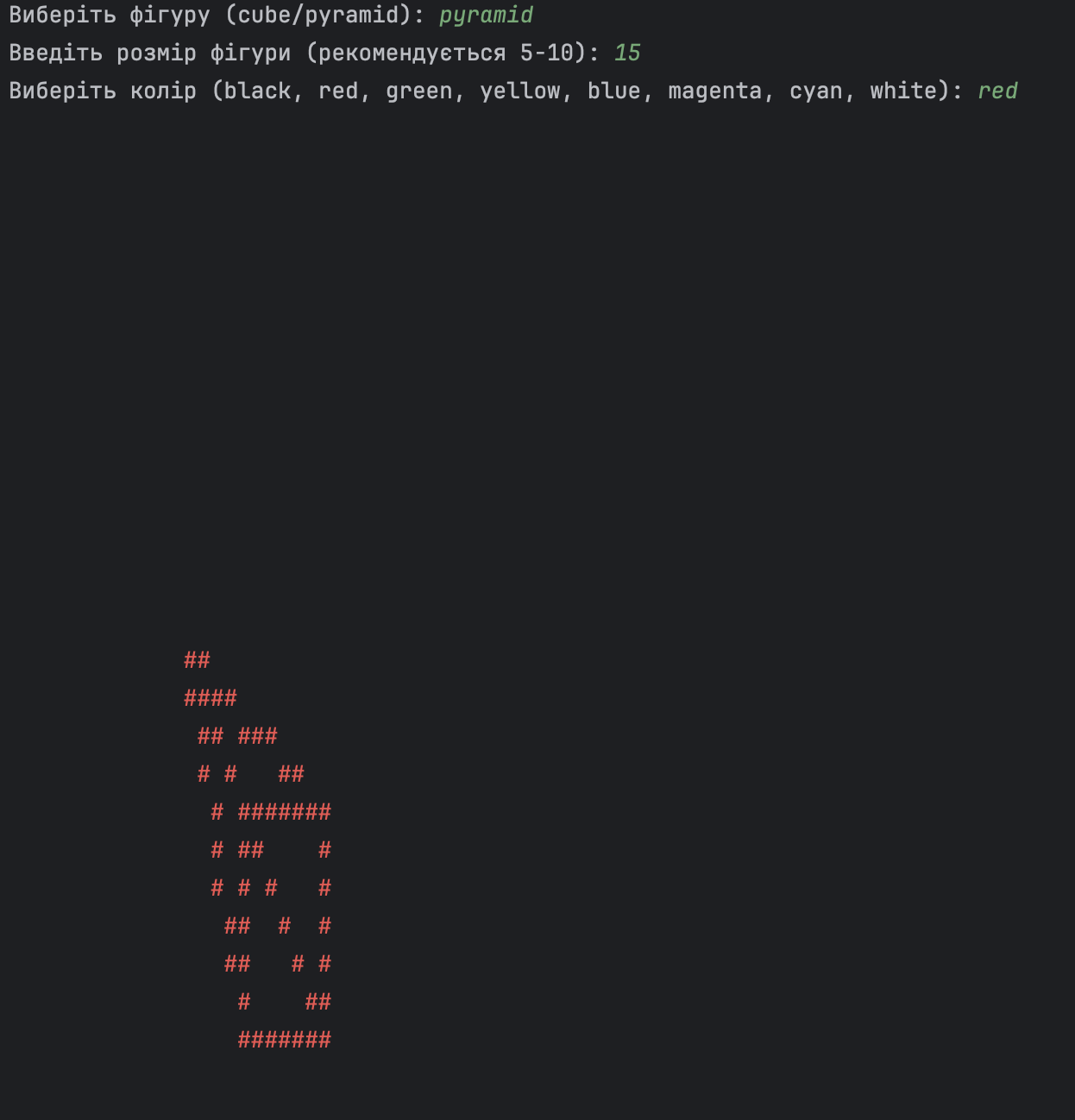
renderer = AsciiArtRenderer(self.shape)

renderer.render\_ascii\_art()

else:

print("Фігура не була налаштована.")

**Виконання:**



**Висновок:**

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 надасть вам глибоке розуміння об'єктно-орієнтованого програмування і алгоритмів графіки, сприятиме творчому підходу до створення ASCII-арту.