МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра інформаційних систем та мереж



Лабораторна робота №5

з дисципліни Спеціалізовані мови програмування

на тему

Розробка ASCII ART генератора для візуалізації 3D-фігур

Виконав:

студент групи РІ-21сп

Владислав ДМИТРЕНКО

Львів – 2024

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

**План роботи**

**Завдання 1:** Розробіть структуру класів для вашого генератора 3D ASCII-арту. Визначте основні компоненти, атрибути та методи, необхідні для програми.

**Завдання 2:** Створіть методи у межах класу для введення користувача та вказання 3D-фігури, яку вони хочуть намалювати, та її параметрів (наприклад, розмір, кольори).

**Завдання 3:** 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:** Реалізуйте метод, який перетворює 3D-представлення фігури у 2D-представлення, придатне для ASCII-арту.

**Завдання 5:** 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:** 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:** Реалізуйте методи для маніпулювання 3D-фігурою, такі масштабування або зміщення, щоб надавати користувачам контроль над її виглядом.

**Завдання 8:** Дозвольте користувачам вибирати варіанти кольорів для їхніх 3D ASCII-арт-фігур. Реалізуйте методи для призначення кольорів різним частинам фігури.

**Завдання 9:** Додайте функціональність для зберігання згенерованого 3D ASCII-арту у текстовий файл

**Завдання 10:** Розгляньте можливість додавання розширених функцій, таких як тінь, освітлення та ефекти перспективи, для підвищення реалізму 3D ASCII-

**Результати тестування:**

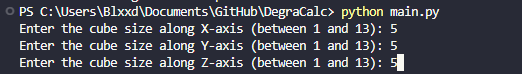
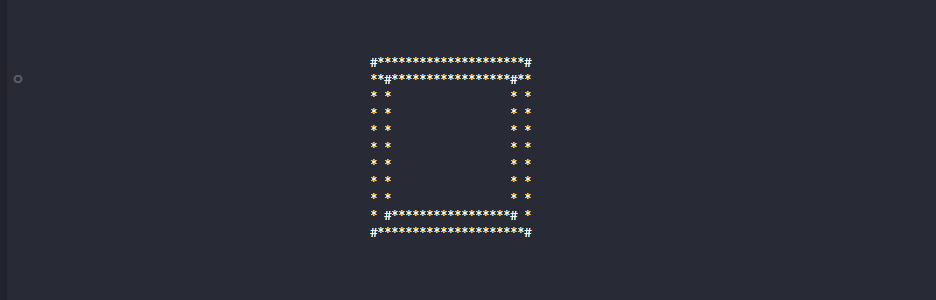
****

Рис. 1. Задання розміру 3Д-кубу

  
Рис. 2. Результат генерації арту

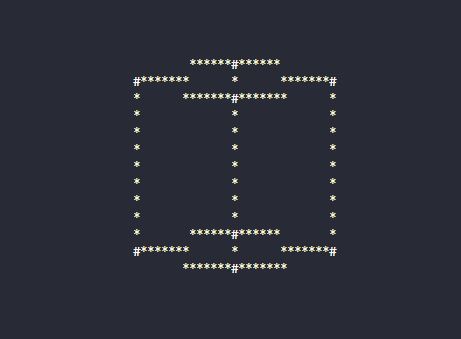
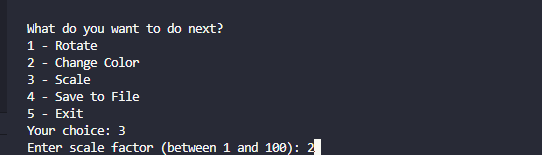
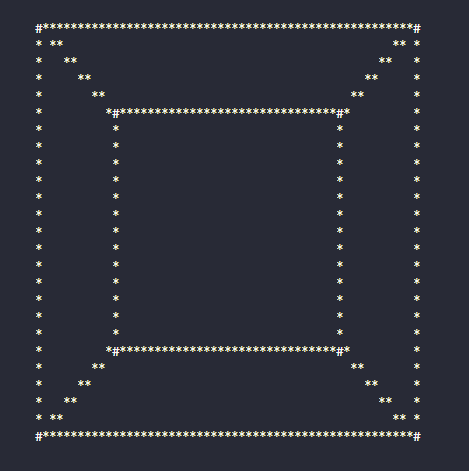


Рис. 3. Обертання куба на 45град.

  
Рис. 4. Інтерфейс користувача

  
Рис. 5. Збільшення куба в 2 рази

Текст рендер функції

from utils.color\_config import ColorConfig

class Renderer:

def \_\_init\_\_(self, resolution, foco, y\_distorter, left\_right, up\_down):

try:

self.resolution = resolution

self.foco = foco

self.y\_distorter = y\_distorter

self.left\_right = left\_right

self.up\_down = up\_down

except Exception as e:

print(f"Error initializing renderer: {e}")

def project(self, cube):

try:

return [(round(2 \* point[0] \* self.foco / (self.foco + point[2])),

round(point[1] \* self.foco / ((self.foco + point[2]) \* self.y\_distorter)))

for point in cube]

except Exception as e:

print(f"Error during projection: {e}")

return []

def get\_lines(self, proj):

try:

connected\_points = [(0, 1), (0, 2), (0, 3), (1, 4), (1, 5), (2, 4), (2, 6),

(3, 5), (3, 6), (7, 6), (7, 5), (7, 4)]

return [self.interpolate(proj[point0][0], proj[point0][1], proj[point1][0], proj[point1][1])

for point0, point1 in connected\_points]

except Exception as e:

print(f"Error during line calculation: {e}")

return []

def interpolate(self, x0, y0, x1, y1):

try:

alpha = (y1 - y0 + 0.001) / (x1 - x0 + 0.001)

beta = y0 - (alpha \* x0)

if alpha > 1 or alpha < -1:

return [((round((y - beta) / (alpha + 0.001))), y) for y in range(int(min(y1, y0)), int(max(y1, y0) + 1))]

else:

return [(x, (round(x \* alpha + beta))) for x in range(int(min(x1, x0)), int(max(x1, x0)) + 1)]

except Exception as e:

print(f"Error during interpolation: {e}")

return []

def render(self, proj, lins, cube\_color):

ascii\_art = []

color\_code = ColorConfig.get\_color\_code(cube\_color)

reset\_code = ColorConfig.COLORS["Reset"]

for j in range(self.resolution):

line = ""

for i in range(self.resolution \* 3):

if (i - self.resolution \* self.left\_right,

j - self.resolution \* self.up\_down) in proj:

line += f"{color\_code}#{reset\_code}"

elif any((i - self.resolution \* self.left\_right,

j - self.resolution \* self.up\_down) in lin for lin in lins):

line += f"{color\_code}\*{reset\_code}"

else:

line += " "

ascii\_art.append(line)

print(line)

return "\n".join(ascii\_art)

**Висновки:** В ході виконання лабораторної роботи було створено високорівневий об'єктно-орієнтований генератор 3D ASCII Куба, який дозволяє користувачу проєктувати, маніпулювати та переглядати фігуру у вигляді ASCII- арту.