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

Виконала:

ст. гр. РІ-31

Оксана ЛЕСЮК

Прийняв:

Сергій Сергійович ЩЕРБАК

Львів-2024

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

**Завдання:**

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.

Виконуючи ці завдання, ви створите 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.

**Хід роботи**

**Cube.py:**

class Cube:

"""Class for rendering a cube."""

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

self.settings = settings

self.width = settings['cube\_width']

self.scale = settings['scale']

self.angles = settings['angles']

self.distance\_from\_viewer = settings['distance\_from\_viewer']

def render(self, char\_array, depth\_array):

"""Render cube surfaces."""

for x in range(-self.width, self.width, self.width // 10):

for y in range(-self.width, self.width, self.width // 10):

self.\_draw\_surface(x, y, -self.width, '@', char\_array, depth\_array)

self.\_draw\_surface(self.width, y, x, '$', char\_array, depth\_array)

def \_draw\_surface(self, x, y, z, char, char\_array, depth\_array):

"""Calculate and render a single surface."""

x\_proj, y\_proj, depth = self.\_project(x, y, z)

screen\_x = int(self.settings['screen']['width'] // 2 + self.scale \* depth \* x\_proj)

screen\_y = int(self.settings['screen']['height'] // 2 + self.scale \* depth \* y\_proj)

idx = screen\_x + screen\_y \* self.settings['screen']['width']

if 0 <= idx < len(char\_array) and depth > depth\_array[idx]:

char\_array[idx] = char

depth\_array[idx] = depth

def \_project(self, x, y, z):

"""Project 3D coordinates to 2D."""

A, B, C = self.angles['A'], self.angles['B'], self.angles['C']

z\_proj = z + self.distance\_from\_viewer

x\_proj = x \* cos(B) + z \* sin(B)

y\_proj = y \* cos(A) - z \* sin(A)

return x\_proj, y\_proj, 1 / z\_proj

def update\_angles(self):

"""Update rotation angles."""

self.angles['A'] += 0.05

self.angles['B'] += 0.05

self.angles['C'] += 0.01

**RendererBase.py**:

class RendererBase(ABC):

"""Abstract base class for renderers."""

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

self.settings = settings

self.width = settings['screen']['width']

self.height = settings['screen']['height']

self.char\_array = [' '] \* (self.width \* self.height)

self.depth\_array = [0] \* (self.width \* self.height)

@abstractmethod

def render\_frame(self):

"""Render a single frame."""

pass

def clear\_screen(self):

"""Clear terminal screen."""

os.system('clear')

def display\_frame(self):

"""Display the rendered frame on the screen."""

print("\033[H", end="")

for idx, char in enumerate(self.char\_array):

if idx % self.width == 0:

print()

print(self.settings['color'] + char, end="")

def initialize\_arrays(self):

"""Reset character and depth arrays."""

self.char\_array = [' '] \* (self.width \* self.height)

self.depth\_array = [0] \* (self.width \* self.height)

**RotatingRenderer.py:**

class RotatingRenderer(RendererBase):

"""Renderer for rotating 3D shapes."""

def render\_frame(self):

"""Render continuously rotating frames."""

cube = Cube(self.settings)

while True:

self.clear\_screen()

self.initialize\_arrays()

cube.render(self.char\_array, self.depth\_array)

self.display\_frame()

cube.update\_angles()

time.sleep(0.05)

**StaticRenderer.py:**

class StaticRenderer(RendererBase):

"""Renderer for static 3D shapes."""

def render\_frame(self):

"""Render a static frame."""

self.clear\_screen()

self.initialize\_arrays()

Cube(self.settings).render(self.char\_array, self.depth\_array)

self.display\_frame()

**SettingsManager.py:**

class SettingsManager:

"""Singleton class for managing settings."""

\_instance = None

def \_\_new\_\_(cls, file\_path):

if not cls.\_instance:

cls.\_instance = super().\_\_new\_\_(cls)

cls.\_instance.file\_path = file\_path

cls.\_instance.settings = {}

return cls.\_instance

def load\_settings(self):

"""Load settings from a JSON file."""

try:

with open(self.file\_path, 'r') as file:

self.settings = json.load(file)

except FileNotFoundError:

self.settings = self.default\_settings()

def save\_settings(self):

"""Save settings to a JSON file."""

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

json.dump(self.settings, file)

@staticmethod

def default\_settings():

"""Default settings."""

return {

'cube\_width': 20,

'scale': 40,

'color': '\x1b[39m',

'angles': {'A': 0, 'B': 0, 'C': 0},

'screen': {'width': 80, 'height': 24},

'horizontal\_offset': 0,

'distance\_from\_viewer': 100,

'increment\_speed': 1,

}

**Посилання на гіт-репозиторій:**

https://github.com/lesiukoksana/SMP-labs

**Висновок:**

У ході виконання лабораторної роботи було розроблено об'єктно-орієнтований додаток для малювання 3D-фігур у форматі ASCII-арту на мові Python. Основними завданнями були проектування класів, реалізація рендерингу 3D-об'єктів, створення можливості їх маніпулювання та відображення у вигляді ASCII-символів на екрані.