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:
ст. гр. ІТ-32
Шоха А.А.
Прийняв:
Щербак С.С.

Львів-2023

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

Завдання на лабораторну роботу

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.

Хід роботи

Код виконання завдання:

cube.py

```
import math
import sys
import time
import random
import numpy as np
from colorama import Fore

characters = [Fore.RED + '/', Fore.BLUE + '.', Fore.YELLOW +
'-', Fore.MAGENTA + ',', Fore.GREEN + '+', Fore.WHITE + '*']

class Cube:
     def __init__(self, width: int, bg: str = ' ', distance:
int = 100, speed: float = 0.6):
              self.__angles: np.array = np.array([0, 0, 0],
dtype=float)

        self.__width: int = width
        self.__screen_width: int = 160
        self.__screen_height: int = 44

        self.__bg: str = bg
                     self.__z_buffer: list[float] = [0] *
self.__screen_width * self.__screen_height
                  self.__buffer: list[str] = [self.__bg] *
self.__screen_width * self.__screen_height

        self.__distance: int = distance
        self.__h_offset: float = -2 * self.__width
        self.__k1: float = 40

        self.__speed: float = speed

    def euler_to_rotation_matrix(self, angles):
        # Convert angles to radians
        angles = np.radians(angles)

        # Extract individual angles
        alpha, beta, gamma = angles

        # Rotation matrix for rotation about x-axis
        R_x = np.array([[1, 0, 0],
                                          [0, math.cos(alpha),
-math.sin(alpha)],
```

```
                                            [0,  math.sin(alpha),
math.cos(alpha)]])

        # Rotation matrix for rotation about y-axis
        R_y = np.array([[math.cos(beta), 0, math.sin(beta)],
                        [0, 1, 0],
                        [-math.sin(beta), 0, math.cos(beta)]])

        # Rotation matrix for rotation about z-axis
           R_z = np.array([[math.cos(gamma), -math.sin(gamma),
0],
                        [math.sin(gamma), math.cos(gamma), 0],
                        [0, 0, 1]])

        # Combined rotation matrix
        R = np.dot(R_z, np.dot(R_y, R_x))

        return R

     def __rotate_point(self, x: float, y: float, z: float) ->
np.array:
                                                rot_matrix     =
self.euler_to_rotation_matrix(self.__angles)
        return np.dot(rot_matrix, [x, y, z])

       def __rotate_face(self, c_x: float, c_y: float, c_z:
float, ch: str) -> None:
        x, y, z = self.__rotate_point(c_x, c_y, c_z)
        z += self.__distance

         xp = int(self.__screen_width / 2 + self.__h_offset + 2
* self.__k1 * x / z)
        yp = int(self.__screen_height / 2 + self.__k1 * y / z)

        ooz = 1 / z
        i = xp + yp * self.__screen_width
              if  0  <=  i  <  len(self.__z_buffer)  and  ooz  >
self.__z_buffer[i]:
            self.__z_buffer[i] = ooz
            self.__buffer[i] = ch

    def update(self) -> None:
                      self.__z_buffer:  list[float]   =   [0]   *
self.__screen_width * self.__screen_height
                   self.__buffer:  list[str]  =  [self.__bg]  *
self.__screen_width * self.__screen_height

        x = -self.__width
        while x < self.__width:
            y = -self.__width
            while y < self.__width:
```

```
                        self.__rotate_face(x, y, -self.__width,
Fore.RED + '/')
                         self.__rotate_face(self.__width, y, x,
Fore.BLUE + '.')
                        self.__rotate_face(-self.__width, y, -x,
Fore.YELLOW + '-')
                        self.__rotate_face(-x, y, self.__width,
Fore.MAGENTA + ',')
                        self.__rotate_face(x, -self.__width, -y,
Fore.GREEN + '+')
                        self.__rotate_face(x, self.__width, -y,
Fore.WHITE + '*')

                y += self.__speed

            x += self.__speed

        self.__angles += np.array([0.5, 0.5, 0.15])

    def draw(self) -> None:
        for n, c in enumerate(self.__buffer):
            sys.stdout.write(c if n % self.__screen_width else
'\n')
```

main.py

```
from cube import Cube
import sys

def main():
    cube = Cube(width=10, speed=2)

    try:
        sys.stdout.write('\x1b[2J')
        while True:
            sys.stdout.write('\x1b[H')

            cube.update()
            cube.draw()
    except KeyboardInterrupt:
        sys.stdout.write('\x1b[2J')
        sys.stdout.write('\x1b[H')

if __name__ == '__main__':
    main()
```

Приклад виконання програми:

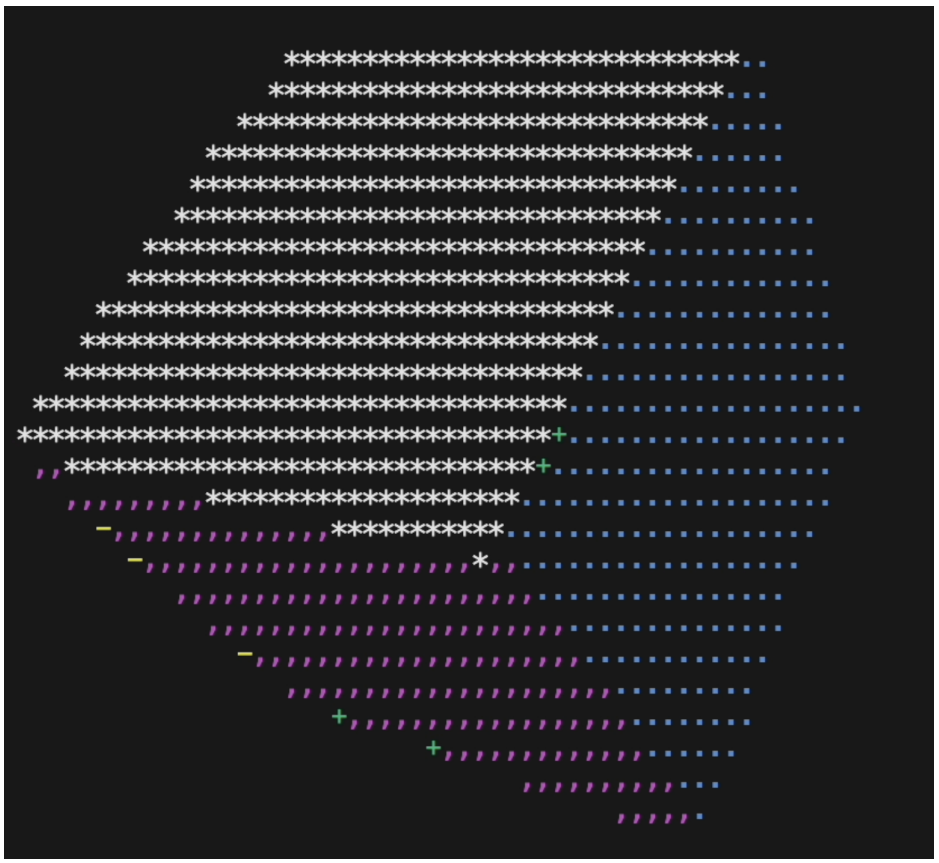

Рис. 1. Анімація обертання кубу

Висновок: під час виконання лабораторної роботи було створено дотаток для малювання 3D-фігур у ASCI-арті на основі об’єктно - орієнтованого підходу та мови Python