Міністерство освіти і науки України

Національний університет «Львівська політехніка»

лабораторної роботи №5

з дисципліни «Спеціалізовані мови програмування»

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

Виконав:

Фігура Ю. В.

Перевірив:

Щербак С.С.

Львів 2024

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

Git: [https://github.com/YuraFihura/-.git](https://github.com/Pivinter/-.git)

Завдання 1: Проектування класів

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

class Shape:

class Cube(Shape):

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).

if choice == '1':

print("Cube selected.")

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.

def draw(self, to\_file=False):

proj = self.project()

# Adding lines (edges) between vertices

for edge in self.edges:

start, end = proj[edge[0]], proj[edge[1]]

for i in range(1, 9):

x = start[0] + i \* (end[0] - start[0]) // 10

y = start[1] + i \* (end[1] - start[1]) // 10

proj.append((x, y))

Завдання 4: Проектування з 3D в 2D

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

def project(self):

# Projecting cube vertices onto 2D screen

return [(round(self.w / 2 + self.xm \* (x \* self.scale\_factor + self.shift[0]) / (z + 2 + self.shift[2])),

round(self.h / 2 + self.ym \* (y \* self.scale\_factor + self.shift[1]) / (z + 2 + self.shift[2])))

for x, y, z in self.cube]

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.

elif choice == '6':

shape.rotate()

shape.draw()

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.

def display\_menu():

print("\n--- Shape Rotation, Size, Color, and Save Menu ---")

print("1. Select Shape (Cube)")

print("2. Select Rotation Axis (X, Y, Z)")

print("3. Set Rotation Angle (in degrees)")

print("4. Choose Shape Color (Red, Green, Blue, etc.)")

print("5. Set Shape Size (Scale Factor)")

print("6. Display Shape")

print("7. Save Shape ASCII Art to a File")

print("8. Exit")

Завдання 7: Маніпуляція фігурою

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

def rotate(self):

# Rotating cube vertices based on set angles

self.cube = [(x, y \* math.cos(self.angle\_x) - z \* math.sin(self.angle\_x),

y \* math.sin(self.angle\_x) + z \* math.cos(self.angle\_x)) for x, y, z in self.cube]

self.cube = [(x \* math.cos(self.angle\_y) + z \* math.sin(self.angle\_y), y,

-x \* math.sin(self.angle\_y) + z \* math.cos(self.angle\_y)) for x, y, z in self.cube]

self.cube = [(x \* math.cos(self.angle\_z) - y \* math.sin(self.angle\_z),

x \* math.sin(self.angle\_z) + y \* math.cos(self.angle\_z), z) for x, y, z in self.cube]

Завдання 8: Варіанти кольорів

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

COLOR\_CODES = {

'red': '\033[31m',

'green': '\033[32m',

'yellow': '\033[33m',

'blue': '\033[34m',

'magenta': '\033[35m',

'cyan': '\033[36m',

'white': '\033[37m',

'reset': '\033[0m'

}

Завдання 9: Збереження та експорт

Додайте функціональність для зберігання згенерованого 3D ASCII-арту у текстовий файл

def save\_to\_file(filename, ascii\_art):

try:

with open(filename, 'w') as file:

file.write(ascii\_art)

print(f"ASCII art saved to {filename}")

except IOError as e:

print(f"Error saving to file: {e}")

Завдання 10: Розширені функції

Розгляньте можливість додавання розширених функцій, таких як тінь, освітлення та ефекти перспективи, для підвищення реалізму 3D ASCII-арту.

def set\_scale(self, scale):

try:

self.scale\_factor = float(scale)

except ValueError:

print("Invalid scale input! Using default scale.")

Висновок: Cтворив додаток для малювання 3D-фігур у ASCII-арті на основі об’єктно - орієнтованого підходу та мови Python.