МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра інформаційних систем та мереж

Лабораторна робота №5

з курсу

СПЕЦІАЛІЗОВАНІ МОВИ ПРОГРАМУВАННЯ

Виконав студент

групи ІТ-21сп

**Петровський Т.О.**

Прийняв

**Щербак С.С.**

Львів - 2023

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

**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.

Код виконаних завдань представлено нижче.

from Lab5.cube import Cube

from Lab5.pyramid import Pyramid

from utils import UserInputHelper

from utils.data\_saver import FileHandler

class ArtInterface:

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

self.\_\_input\_helper = UserInputHelper()

def run(self):

while True:

print("Choose options")

print("1 - Draw 2D cube\n2 - Draw 3D cube\n3 - Draw 2D pyramid\n0 - Stop\n")

answer = self.\_\_input\_helper.get\_limited\_user\_input('', ['1', '2', '3', '0'])

if answer is '1':

art = self.\_\_get\_2d\_cube()

self.\_\_print\_art(art)

elif answer is '2':

art = self.\_\_get\_3d\_cube()

self.\_\_print\_art(art)

elif answer is '3':

art = self.\_\_get\_pyramid()

self.\_\_print\_art(art)

else:

break

self.\_\_save\_option(art)

def \_\_get\_2d\_cube(self):

x = self.\_\_input\_helper.get\_int\_user\_input("Enter x:\n")

y = self.\_\_input\_helper.get\_int\_user\_input("Enter y:\n")

return Cube.create\_2d(x, y)

def \_\_get\_3d\_cube(self):

x = self.\_\_input\_helper.get\_int\_user\_input("Enter x:\n")

y = self.\_\_input\_helper.get\_int\_user\_input("Enter y:\n")

z = self.\_\_input\_helper.get\_int\_user\_input("Enter z:\n")

return Cube().build\_3d(x, y, z)

def \_\_get\_pyramid(self):

x = self.\_\_input\_helper.get\_int\_user\_input("Enter x:\n")

y = self.\_\_input\_helper.get\_int\_user\_input("Enter y:\n")

return Pyramid.create\_pyramid(x, y)

# provide option to save object

def \_\_save\_option(self, content):

answer = self.\_\_input\_helper.get\_limited\_user\_input("Do you want to save art?(y,n)\n", ['y', 'n'])

if answer is 'y':

file\_name = self.\_\_input\_helper.get\_user\_input("Enter file name\n")

FileHandler().save\_to\_txt(file\_name, content)

@staticmethod

def \_\_print\_art(art):

for i in art:

print(i)

class Cube:

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

pass

# draw 2d representation of cube

@staticmethod

def create\_2d(x, y):

lst\_1 = []

for i in range(y):

line = ''

for j in range(x):

if i == 0 or i == y - 1:

if j == 0 or j == x - 1:

line += '+'

else:

line += '-'

else:

if j == 0 or j == x - 1:

line += '|'

else:

line += ' '

lst\_1 += [line]

return lst\_1

# Creates starting lines for the 3D cube drawing

@staticmethod

def \_\_create\_start\_line(length, size\_z):

lines = []

for index in range(length):

if index < size\_z:

line = ' ' \* (size\_z - index)

else:

line = ''

lines += [line]

lines += ['']

return lines

# Builds a single line for the 3D cube representation

@staticmethod

def \_\_build\_line(size\_x, index, sym, length):

line = ''

for size in range(size\_x + 1):

if index == 0 or index == length:

line += '+' if size == 0 or size == size\_x else '-'

else:

line += sym if size == 0 or size == size\_x else ' '

return line

# draw 3D-cube using ASCII art

def build\_3d(self, size\_x, size\_y, size\_z):

length\_x = size\_x + size\_z

length\_y = size\_y + size\_z

lines = self.\_\_create\_start\_line(length\_y, size\_z)

index = 0

i = 1

new\_lines = []

while index <= length\_y:

line = ''

if size\_z > index:

line = self.\_\_build\_line(size\_x, index, '/', length\_y)

elif size\_z == index:

line = self.\_\_build\_line(size\_x, 0, '|', length\_y)

elif size\_z < index:

line = self.\_\_build\_line(size\_x, index, '|', length\_y)

if size\_y > index != 0:

if length\_x - (index + len(line)) >= 0:

line += (' ' \* (index - 1)) + '|'

else:

line += (' ' \* (length\_x - len(line))) + '|'

elif size\_y == index:

if length\_x - (index + len(line)) >= 0:

line += (' ' \* (index - 1)) + '+'

else:

line += (' ' \* (length\_x - len(line))) + '+'

elif size\_y < index != length\_y:

if len(lines[index]) == 0:

line += (' ' \* (length\_y - index - 1)) + '/'

else:

i += 1

line += (' ' \* (index - i)) + '/'

new\_lines += [lines[index] + line]

index += 1

return new\_lines

class Pyramid:

@staticmethod

def create\_pyramid(x, y):

pyramid = []

for i in range(y):

line = ''

for j in range(x):

if i == 0: # for the top of the pyramid

line += '+' if j == x // 2 else ' '

elif i == y - 1: # drawing the base of the pyramid

if j == 0 or j == x - 1:

line += '+' # Place '+' at the beginning and end of the line

elif 0 < j < x - 1:

line += '-' # Fill the middle with '-'

elif j == x // 2 - i or j == x // 2 + i: # drawing the edges of the pyramid

line += '/' if j == x // 2 - i else '\\'

elif x // 2 - i < j < x // 2 + i: # drawing the sides of the pyramid

line += ' '

else:

line += ' '

pyramid.append(line)

return pyramid

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