МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра ІСМ

******

Звіт

до лабораторної роботи №5

На тему “Розробка ASCII ART генератора для візуалізації 3D-фігур”

З дисципліни “Спеціалізовані мови програмування”

*Виконав:*

*ст. гр. ІТ-31*

*Володимир Лисецький*

*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.

**Хід роботи**

**Програмний код:**

/figures/Kube.py

from termcolor import colored, COLORS

class Kube:

def \_\_init\_\_(self, x, y, z, color):

self.x = x

self.y = y

self.z = z

self.color = color

def print(self):

kube = self.draw()

for item in kube:

print(item)

def write(self):

kube = self.draw()

with open("output.txt", "w") as file:

for item in kube:

file.write(item + "\n")

def generate\_kube(self, n, x, y, cde):

output = colored(

f"{str(cde[0]).rjust(n + 1)}" +

f"{cde[1] \* (9 \* x - 1)}" +

f"{cde[0]}" +

f"{str(cde[2]).rjust(y + 1) if len(cde) > 2 else ''}"

, self.color)

return output

def draw(self):

result = [self.generate\_kube(self.y + 1, self.x, 0, '+-')]

for i in range(1, self.y):

result.append(self.generate\_kube(self.y - i + 1, self.x, i - 1, '/ |'))

result.append(self.generate\_kube(0, self.x, self.y, '+-|'))

for \_ in range(4 \* self.z - self.y - 3):

result.append(self.generate\_kube(0, self.x, self.y, '| |'))

result.append(self.generate\_kube(0, self.x, self.y, '| +'))

for i in range(self.y - 1, 0, -1):

result.append(self.generate\_kube(0, self.x, i, '| /'))

result.append(self.generate\_kube(0, self.x, 0, "+-\n"))

return result

/figures/Sphere.py

from helpers.render import Renderer

import math

from termcolor import COLORS, colored

class Sphere:

def \_\_init\_\_(self, size, shade, color,ambient, vector):

renderer\_init = Renderer()

self.size = size

self.shade = shade

self.ambient = ambient

self.light = renderer\_init.normalize(vector)

self.render = renderer\_init.return\_shades()

self.color = color

def draw\_sphere(self):

size, shade, ambient, light, renders = self.size, self.shade, self.ambient, self.light, self.render

result = []

renderer\_init = Renderer()

for i in range(int(math.floor(-size)),int(math.ceil(size)+1)):

x = i + 0.5

line = ''

for j in range(int(math.floor(-2\*size)),int(math.ceil(2\*size)+1)):

y = j/2 + 0.5

if x\*x + y\*y <= size\*size:

vec = renderer\_init.normalize((x,y,math.sqrt(size\*size - x\*x - y\*y)))

b = renderer\_init.dot(light,vec)\*\*shade + ambient

intensity = int((1-b)\*(len(renders)-1))

line += renders[intensity] if 0 <= intensity < len(renders) else renders[0]

else:

line += ' '

result.append(line)

return result

def print(self):

sphere = self.draw\_sphere()

for item in sphere:

print(colored(item, self.color))

def write(self):

sphere = self.draw\_sphere()

with open("output.txt", "w") as file:

for item in sphere:

file.write(item + "\n")

/helpers/render.py

import math

class Renderer:

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

self.shades = ('.',':','!','\*','o','e','&','#','%','@')

def return\_shades(self):

return self.shades

def normalize(self, v):

len = math.sqrt(v[0]\*\*2 + v[1]\*\*2 + v[2]\*\*2)

return (v[0]/len, v[1]/len, v[2]/len)

def dot(self, x,y):

d = x[0]\*y[0] + x[1]\*y[1] + x[2]\*y[2]

return -d if d < 0 else 0

/helpers/menu.py

from figures.sphere import Sphere

from figures.kube import Kube

class Menu:

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

# Default figure = sphere

self.size = 20

self.shade = 3

self.color = "red"

self.ambient = 0.2

self.vector = (30, 40, -50)

def display\_menu(self):

print("1. Create Sphere")

print("2. Create Cube")

def get\_user\_choice(self):

return int(input("Enter your choice (1 for Sphere, 2 for Cube): "))

def ask\_write\_to\_file(self, figure):

answer = input("Do you want to write this figure to a file? (yes/no): ").lower()

if answer == 'yes':

figure.write()

def create\_shape(self):

self.display\_menu()

choice = self.get\_user\_choice()

if choice == 1:

sphere = self.create\_sphere()

sphere.print()

self.ask\_write\_to\_file(sphere)

return sphere

elif choice == 2:

kube = self.create\_kube()

kube.print()

self.ask\_write\_to\_file(kube)

return

else:

print("Invalid choice. Please choose 1 or 2.")

return None

def create\_sphere(self):

self.size = float(input("Enter the size of the sphere: "))

self.shade = int(input("Enter the shade level (integer): "))

self.color = input("Enter the color of the sphere: ")

self.ambient = float(input("Enter the ambient light level (float): "))

self.vector = tuple(map(float, input("Enter the light direction vector (comma-separated, e.g., 1,0,0): ").split(',')))

return Sphere(self.size, self.shade, self.color, self.ambient, self.vector)

def create\_kube(self):

x = int(input("Enter x: "))

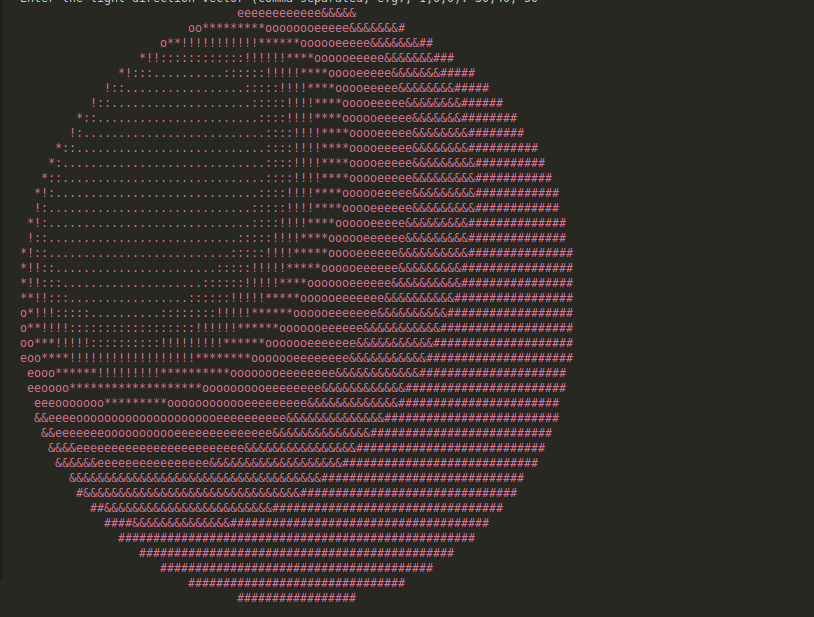
y = int(input("Enter y: "))

z = int(input("Enter z: "))

color = input("Enter color: ")

return Kube(x, y, z, color)

Результат виконання програми:



**Висновок:** Під час виконання даної лабораторної роботи було створено додаток для малювання 3D-фігур у ASCII-арті на основі об’єктно - орієнтованого підходу та мови Python.