МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ



Лабораторна робота №5

з дисципліни “Спеціалізовані мови програмування”

Виконала:

студентка групи ІТ-31

Юлія ДОБОШ

Прийняв:

доцент кафедри ІСМ

Сергій ЩЕРБАК

Львів-2023

**Тема роботи:** Розробка ASCII ART генератора для візуалізації 3D-фігур

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

Хід 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.

Детально код можна переглянути тут : <https://github.com/YuliiaDobosh/Python.git>

Код програми :

import logging

from .file\_saver import FileSaver

from .graphic.graphic import Graphic, Figure, Color

from .interfaces.art\_generator import ArtGenerate

from consolemenu import ConsoleMenu

from consolemenu.items import FunctionItem

import curses

import curses.ascii

from .graphic.point\_3d import Point3D

from .graphic.figure\_factory import FigureFactory

class ArtGenerate(ArtGenerate):

"""Concrete implementation of the ArtGenerate abstract class.

This class provides methods for generating and displaying 3D ASCII art.

"""

def \_\_init\_\_(self, graphic: Graphic, file\_saver: FileSaver) -> None:

"""Initialize the ArtGenerate object."""

super().\_\_init\_\_(graphic, file\_saver)

self.menu = ConsoleMenu()

self.menu.append\_item(FunctionItem("Generate 3D Art", self.generate\_art))

self.menu.append\_item(FunctionItem("Show 3D Arts", self.show\_arts))

self.arts: list[list[Figure]] = []

def \_\_generate\_art(self, window) -> str:

"""Generate 3D ASCII art with user interaction."""

logging.info("Початок генерації 3D арту")

def get\_user\_input(stdscr, prompt\_string, height=0):

"""Get user input from the console."""

stdscr.addstr(height, 0, prompt\_string)

curses.echo()

stdscr.refresh()

input\_str = stdscr.getstr(height, len(prompt\_string))

curses.noecho()

return input\_str.decode('utf-8')

while True:

window.clear()

symbol = get\_user\_input(window, "Enter symbol: ", 0)

prmt = f'Enter color({", ".join([c.name for c in Color])}): '

c = get\_user\_input(window, prmt, 1).upper()

color = Color[c]

figure\_type = get\_user\_input(window, "Enter figure type(Cube, Piramid, Sphere): ", 2).lower()

figure\_graphic = FigureFactory()

if figure\_type == "cube":

fig = figure\_graphic.create\_cube(Point3D(0, 0, 0))

elif figure\_type == 'piramid':

fig = figure\_graphic.create\_pyramid(Point3D(0, 0, 0))

elif figure\_type == 'sphere':

radius = float(get\_user\_input(window, "Enter radius: ", 3))

fig = figure\_graphic.create\_sphere(Point3D(0, 0, 0), radius)

fig.set\_symbol(symbol)

fig.set\_color(color)

self.graphic.add\_figure(fig)

dx, dy, dz = 0, 0, 0

self.graphic.set\_window(window)

while True:

window.clear()

self.graphic.last\_figure.translate(dx, dy, dz)

self.graphic.render\_figures()

window.addstr(0, 0, f"{self.graphic.last\_figure.local\_transform}")

window.addstr(self.graphic.height - 5, 0, "x - a/d")

window.addstr(self.graphic.height - 4, 0, "y - w/s")

window.addstr(self.graphic.height - 3, 0, "z - q/e")

window.addstr(self.graphic.height - 2, 0, "ESC - Finish")

key = window.getch()

if key == curses.ascii.ESC:

window.clear()

break

elif key == ord('a'):

dy, dz, dx = 0, 0, -0.08

elif key == ord('d'):

dy, dz, dx = 0, 0, 0.08

elif key == ord('w'):

dx, dz, dy = 0, 0, -0.08

elif key == ord('s'):

dx, dz, dy = 0, 0, 0.08

elif key == ord('q'):

dx, dy, dz = 0, 0, -0.08

elif key == ord('e'):

dx, dy, dz = 0, 0, 0.08

window.refresh()

window.clear()

self.graphic.render\_figures()

window.addstr(self.graphic.height - 3, 0, "Enter - Add figure")

window.addstr(self.graphic.height - 2, 0, "ESC - Finish")

key = window.getch()

if key == curses.ascii.ESC:

self.arts.append([f.copy() for f in self.graphic.figures])

self.graphic.figures.clear()

break

logging.info("3D арт сгенеровано")

def generate\_art(self):

"""Wrapper function to initiate the generation of 3D ASCII art."""

curses.wrapper(self.\_\_generate\_art)

def \_\_show\_arts(self, window):

"""Display saved 3D ASCII arts."""

logging.info("Відображення збережених 3D артів")

self.graphic.is\_init = False

self.graphic.set\_window(window)

while True:

window.clear()

index = 0

self.graphic.figures = self.arts[index]

self.graphic.render\_figures()

window.addstr(self.graphic.height - 3, 0, "a/d - Move")

window.addstr(self.graphic.height - 2, 0, "ESC - Finish")

window.refresh()

key = window.getch()

if key == curses.ascii.ESC:

break

elif key == ord('a'):

index -= 1 if index == 0 else 0

elif key == ord('d'):

index += 1 if index < len(self.arts) - 1 else 0

self.graphic.figures = []

logging.info("Відображення завершено")

def show\_arts(self):

"""Wrapper function to initiate the display of saved 3D ASCII arts."""

curses.wrapper(self.\_\_show\_arts)

def start(self):

"""Start the ArtGenerate application."""

logging.info("Старт ArtGenerate")

self.menu.show()

logging.info("ArtGenerate завершено")

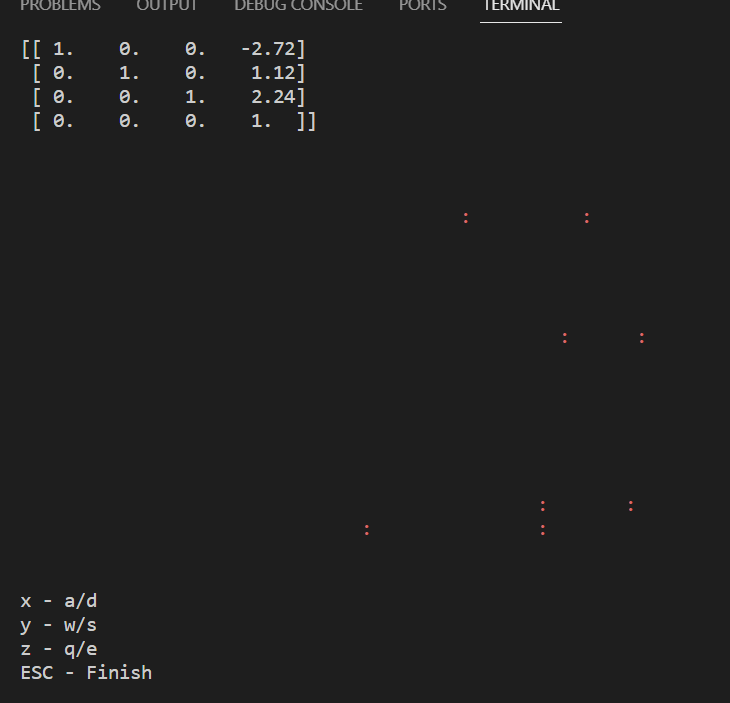


Рис.1 Результат виконання роботи

**Висновок**: Виконуючи лабораторну роботу я створила високорівневий об'єктно-орієнтований генератор 3D ASCII-арту, який дозволить користувачам проектувати, відображати та маніпулювати 3D-фігурами в ASCII-арті. Цей проект надав мені глибоке розуміння об'єктно-орієнтованого програмування і алгоритмів графіки, сприятиме творчому підходу до створення ASCII-арту.