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-21сп

Олександр КОЗАК

Прийняв

доцент каф. ІСМ:

Сергій ЩЕРБАК

|  |  |
| --- | --- |
| **Балів** | **Дата** |
|  |  |

Львів-2023

**Мета**: 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.

**art\_menu.py**

"""

A module that defines the ArtMenu class for creating and manipulating shapes.

"""

from UI.menu import Menu

from UI.menu\_item import Item

from shared.file\_handler import FileHandler

from shared.settings import get\_lab\_settings

from classes.lab5.shapes.cube import Cube

from classes.lab5.shapes.pyramid import Pyramid

settings = get\_lab\_settings("lab5")

ART\_2D\_PATH = settings["art\_2D\_path"]

ART\_3D\_PATH = settings["art\_3D\_path"]

class ArtMenu:

"""

A class representing an art menu for creating and manipulating shapes.

Attributes:

- \_\_shape: The currently selected shape.

Methods:

- \_\_init\_\_(self): Initializes an instance of the ArtMenu class.

- set\_shape(self): Sets the shape based on user input.

- get\_shape(self): Returns the currently selected shape.

- sub\_menu(self): Displays a sub-menu for viewing and manipulating the shape.

- view\_art\_2D(self): Prints the 2D representation of the shape.

- view\_art\_3D(self): Prints the 3D representation of the shape.

- change\_settings(self): Displays a menu for changing the settings of the shape.

- save\_menu(self): Displays a menu for saving and viewing the saved art.

- \_save\_to\_file\_2D(self): Saves the 2D representation of the shape to a file.

- \_save\_to\_file\_3D(self): Saves the 3D representation of the shape to a file.

- \_view\_saved\_2D(self): Reads and displays the saved 2D art from a file.

- \_view\_saved\_3D(self): Reads and displays the saved 3D art from a file.

"""

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

self.\_\_shape = None

# def main\_menu(self):

# art\_menu = Menu("\nArt Menu")

# art\_menu.add\_item(Item('1', 'Choose shape', self.set\_shape))

# art\_menu.add\_item(Item('0', 'Exit'))

# art\_menu.run()

def set\_shape(self):

"""

Sets the shape based on user input.

"""

shape = input("Choose shape [ cube, pyramid ]: ")

if shape not in ['cube', 'pyramid']:

print("Invalid shape")

return

if shape == 'cube':

self.\_\_shape = Cube()

if shape == 'pyramid':

self.\_\_shape = Pyramid()

self.sub\_menu()

def get\_shape(self):

"""

Returns the currently selected shape.

"""

return self.\_\_shape

def sub\_menu(self):

"""

Displays a sub-menu for viewing and manipulating the shape.

"""

sub\_menu = Menu("\nArt Menu")

sub\_menu.add\_item(Item('1', 'View 2D', self.view\_art\_2D))

sub\_menu.add\_item(Item('2', 'View 3D', self.view\_art\_3D))

sub\_menu.add\_item(Item('3', 'Change Settings', self.change\_settings))

sub\_menu.add\_item(Item('4', 'Save Art', self.save\_menu))

sub\_menu.add\_item(Item('0', 'Back'))

sub\_menu.run()

def view\_art\_2D(self):

"""

Prints the 2D representation of the shape.

"""

self.\_\_shape.print\_art\_2D()

def view\_art\_3D(self):

"""

Prints the 3D representation of the shape.

"""

self.\_\_shape.print\_art\_3D()

def change\_settings(self):

"""

Displays a menu for changing the settings of the shape.

"""

self.\_\_shape.get\_settings().settings\_menu()

self.\_\_shape.set\_settings(self.\_\_shape.get\_settings().get\_settings\_obj())

def save\_menu(self):

"""

Displays a menu for saving and viewing the saved art.

"""

save\_menu = Menu("\nSave Menu")

save\_menu.add\_item(Item('1', 'Save 2D to File', self.\_save\_to\_file\_2D))

save\_menu.add\_item(Item('2', 'Save 3D to File', self.\_save\_to\_file\_3D))

save\_menu.add\_item(Item('3', 'View Saved 2D', self.\_view\_saved\_2D))

save\_menu.add\_item(Item('4', 'View Saved 3D', self.\_view\_saved\_3D))

save\_menu.add\_item(Item('0', 'Back'))

save\_menu.run()

def \_save\_to\_file\_2D(self):

"""

Saves the 2D representation of the shape to a file.

"""

saved\_file = FileHandler(ART\_2D\_PATH)

saved\_file.write\_to\_file(self.\_\_shape.get\_2D())

def \_save\_to\_file\_3D(self):

"""

Saves the 3D representation of the shape to a file.

"""

saved\_file = FileHandler(ART\_3D\_PATH)

saved\_file.write\_to\_file(self.\_\_shape.get\_3D())

def \_view\_saved\_2D(self):

"""

Reads and displays the saved 2D art from a file.

"""

saved\_file = FileHandler(ART\_2D\_PATH)

saved\_file.read\_from\_file()

def \_view\_saved\_3D(self):

"""

Reads and displays the saved 3D art from a file.

"""

saved\_file = FileHandler(ART\_3D\_PATH)

saved\_file.read\_from\_file()

**art\_settings.py**

"""

A module that defines the PrintArt class for printing 2D and 3D art.

"""

from termcolor import colored

from classes.lab4.console\_reader.data\_from\_console import get\_console\_width

class PrintArt:

"""

A class that represents the printing of art.

Attributes:

- \_shape: The Shape object to be printed.

Methods:

- \_\_init\_\_(self, Shape): Initialize the PrintArt object.

- get\_shape(self): Get the Shape object associated with the PrintArt object.

- set\_shape(self, shape): Set the Shape object associated with the PrintArt object.

- print\_art\_2D(self): Print the 2D art of the Shape object.

- print\_art\_3D(self): Print the 3D art of the Shape object.

- \_\_justify\_art(self, art): Justify the art by adding padding based on the justification settings.

- \_\_get\_padding(self, art): Calculate the padding based on the console width and justification settings.

"""

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

"""

Initialize the PrintArt object.

Parameters:

- Shape: The Shape object to be printed.

"""

self.\_shape = Shape

def get\_shape(self):

"""

Get the Shape object associated with the PrintArt object.

Returns:

- The Shape object.

"""

return self.\_shape

def set\_shape(self, shape):

"""

Set the Shape object associated with the PrintArt object.

Parameters:

- shape: The new Shape object.

"""

self.\_shape = shape

def print\_art\_2D(self):

"""

Print the 2D art of the Shape object.

"""

art = self.\_\_justify\_art(self.\_shape.get\_2D())

print(colored("\nArt 2D\n"))

print(colored(art, self.\_shape.get\_settings\_obj().get\_color()))

def print\_art\_3D(self):

"""

Print the 3D art of the Shape object.

"""

art = self.\_\_justify\_art(self.\_shape.get\_3D())

print(colored("\nArt 3D\n"))

print(colored(art, self.\_shape.get\_settings\_obj().get\_color()))

def \_\_justify\_art(self, art):

"""

Justify the art by adding padding based on the justification settings.

Parameters:

- art: The art to be justified.

Returns:

- The justified art.

"""

padding = self.\_\_get\_padding(art)

art\_lines = art.split('\n')

aligned\_lines = [" " \* padding + line for line in art\_lines]

art = '\n'.join(aligned\_lines)

return art

def \_\_get\_padding(self, art):

"""

Calculate the padding based on the console width and justification settings.

Parameters:

- art: The art to be justified.

Returns:

- The padding value.

"""

console\_width = get\_console\_width()

art\_len = len(art)//self.\_shape.get\_settings\_obj().get\_size()

justify = self.\_shape.get\_settings\_obj().get\_justify()

if justify == "center":

return (console\_width - art\_len) // 2

if justify == "right":

return console\_width - art\_len

return 0

**lab\_menu.py**

"""

Art Menu Module (Lab 5)

This module provides a menu function 'lab\_menu' that creates and runs an Art Menu (Lab 5).

The Art Menu allows users to choose a shape and provides an option to exit the menu.

Classes:

- ArtMenu: A class that implements the Art Menu functionality.

Functions:

- lab\_menu: Creates and runs a menu for the Art Menu (Lab 5). It adds menu items for

choosing a shape and exiting the menu.

Usage:

Import this module and call the 'lab\_menu' function to start the Art Menu. The program

will display a menu with options for choosing a shape or exiting.

"""

from UI.menu import Menu

from UI.menu\_item import Item

from classes.lab5.art.art\_menu import ArtMenu

def lab\_menu():

"""

This function creates and runs a menu for the Art Menu (Lab 5).

It adds menu items for choosing a shape and exiting the menu.

"""

art = ArtMenu()

art\_menu = Menu("\nArt Menu (Lab 5)")

art\_menu.add\_item(Item('1', 'Choose shape', art.set\_shape))

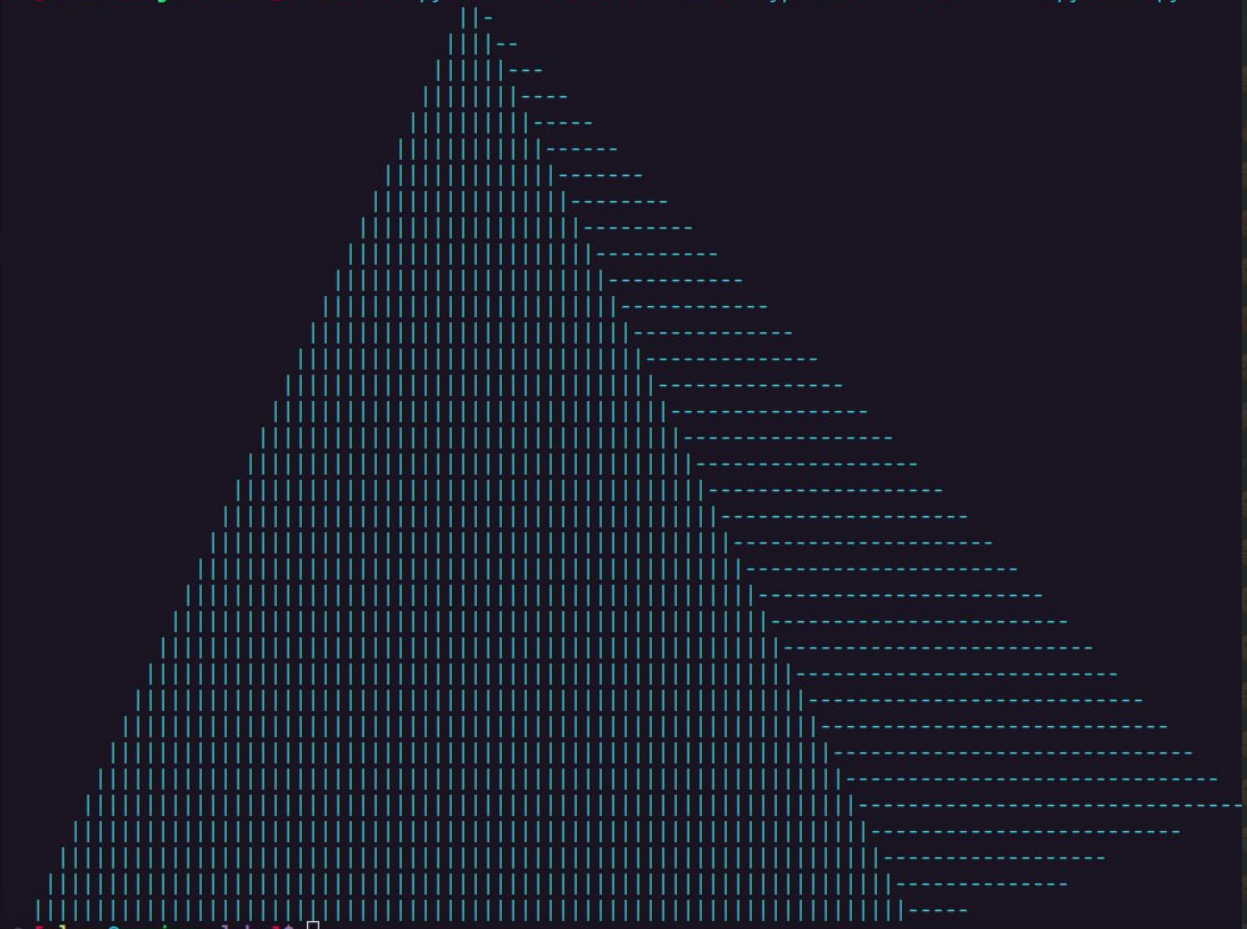
art\_menu.add\_item(Item('0', 'Exit'))

art\_menu.run()

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

*Рис. 1 Результат виконання завдання*



*Рис. 2 Результат виконання завдання*

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