МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра інформаційних систем та мереж

Звіт

до лабораторної роботи №4

Розробка ASCII ART генератора для візуалізації 2D-фігур

Виконала:

ст. гр. PI-32
Довгошия А.А

Балів	**Дата**

Прийняв:

асис. каф. ІСМ

Щербак С.С

Львів — 2024

Мета: Створення Генератора ASCII-арту без використання зовнішніх бібліотек

Хід роботи

Завдання 1: Введення користувача

Створіть програму Python, яка отримує введення користувача щодо слова або фрази, яку вони хочуть перетворити в ASCII-арт.

Завдання 2: Набір символів

Визначте набір символів (наприклад, '@', '#', '*', тощо), які будуть використовуватися для створення ASCII-арту. Ці символи будуть відображати різні відтінки.

Завдання 3: Розміри Art-у

Запитайте у користувача розміри (ширина і висота) ASCII-арту, який вони хочуть створити. Переконайтеся, що розміри в межах керованого діапазону

Завдання 4: Функція генерації Art-у

Напишіть функцію, яка генерує ASCII-арт на основі введення користувача, набору символів та розмірів. Використовуйте введення користувача, щоб визначити, які символи використовувати для кожної позиції в Art-у.

Завдання 5: Вирівнювання тексту

Реалізуйте опції вирівнювання тексту (ліво, центр, право), щоб користувачі могли вибирати, як їх ASCII-арт розміщується на екрані.

Завдання 6: Відображення мистецтва

Відобразіть створений ASCII-арт на екрані за допомогою стандартних функцій друку Python.

Завдання 7: Збереження у файл

Додайте можливість зберігати створений ASCII-арт у текстовий файл, щоб користувачі могли легко завантажувати та обмінюватися своїми твореннями.

Завдання 8: Варіанти кольорів

Дозвольте користувачам вибирати опції кольорів (чорно-білий, відтінки сірого) для свого ASCII-арту.

Завдання 9: Функція попереднього перегляду

Реалізуйте функцію попереднього перегляду, яка показує користувачам попередній перегляд їх ASCII-арту перед остаточним збереженням

Завдання 10: Інтерфейс, зрозумілий для користувача

Створіть інтерфейс для користувача у командному рядку, щоб зробити програму легкою та інтуїтивно зрозумілою для використання.

Generate_art.py

```
from src.const.const import REPLACE_CHARACTER_PROMPT,
ONE_CHARACTER_MESSAGE
from src.ui.lab3.art_ui import ArtUI

class AsciiArtGeneratorLab4:
    def __init__(self, config):
```

```
        self.config = config
        self.art_ui = ArtUI(self.config)

    def generate_ascii_art(self):
        text = self.art_ui.get_text_input()
        font_name = self.art_ui.get_font_choice()
        color = self.art_ui.get_color_choice()
        justify = self.art_ui.get_justification_choice()

        font = self.config.custom_font if font_name == "custom" else
self.config.star_font

        replacement_char = self.get_replacement_char()

        ascii_art = self.create_custom_ascii_art(text, font, justify, color,
replacement_char)

print(f"{self.config.colors.get(color)}{ascii_art}{self.config.colors.get('reset')}")

        self.art_ui.prompt_to_save_ascii_art(ascii_art)



    def start_ui(self):
        self.generate_ascii_art()

    def get_replacement_char(self):
        while True:
            replacement_char = input(REPLACE_CHARACTER_PROMPT)
```

```
            if len(replacement_char) <= 1:
                return replacement_char
            else:
                print(ONE_CHARACTER_MESSAGE)

    def create_custom_ascii_art(self, text, font, justify, color='reset',
replacement_char=None):
        ascii_art_lines = []

        for char in text.upper():
            if char in font:
                char_art = font[char]
                for i in range(len(char_art)):
                    if i >= len(ascii_art_lines):
                        ascii_art_lines.append([''])
                    line = char_art[i]
                    ascii_art_lines[i].append(line)
            else:
                for i in range(len(ascii_art_lines)):
                    if i >= len(ascii_art_lines):
                        ascii_art_lines.append([''])
                    ascii_art_lines[i].append(' ' * (len(next(iter(font.values()))[0]) + 1))

        for i in range(len(ascii_art_lines)):
            ascii_art_lines[i] = ''.join(ascii_art_lines[i])

        ascii_art = '\n'.join(ascii_art_lines)
```

```
    ascii_art = self.replace_characters_in_art(ascii_art, replacement_char)

    justified_art = self.apply_justification(ascii_art, justify)

    return self.apply_color(justified_art, color)

def replace_characters_in_art(self, ascii_art, replacement_char):
    if replacement_char:
        ascii_art = ascii_art.replace('x', replacement_char)
        ascii_art = ascii_art.replace('☆', replacement_char)
    return ascii_art

def apply_justification(self, ascii_art, justify):
    justified_lines = []
    for line in ascii_art.splitlines():
        if justify == 'center':
            justified_lines.append(line.center(self.config.fixed_width))
        elif justify == 'right':
            justified_lines.append(line.rjust(self.config.fixed_width))
        else:
            justified_lines.append(line.ljust(self.config.fixed_width))
    return '\n'.join(justified_lines)

def apply_color(self, ascii_art, color):
    color_code = self.config.colors.get(color, self.config.colors['reset'])
    return f"{color_code}{ascii_art}{self.config.colors['reset']}"
```

main.py

```
from src.classes.lab4.generate_art import AsciiArtGeneratorLab4
from src.config.lab4_config import Lab4Config

def main():
    config = Lab4Config()
    generator = AsciiArtGeneratorLab4(config)
    generator.start_ui()

if __name__ == "__main__":
    main()
```

art_ui.py

```
"""
This module defines the ArtUI class, which provides an interface for
generating and saving ASCII art based on user preferences.
Users can select text, font, color, justification, and width for the ASCII art.
"""

from src.classes.lab3.generate_art import AsciiArtGenerator
from src.const.const import (
    FONT_CHOICE_PROMPT, COLOR_CHOICE_PROMPT,
JUSTIFY_CHOICE_PROMPT,
    WIDTH_CHOICE_PROMPT, SAVE_CHOICE_PROMPT,
FILE_NAME_PROMPT,
    USER_ASCII_TEXT_PROMPT
)
```

```
from lib import FileOperations, UserInput


class ArtUI:
    """
    Interface class for managing user input, generating ASCII art,
    and providing options to save the artwork.
    """

    def __init__(self, config):
        """Initialize ArtUI with a configuration and ASCII art generator."""
        self.config = config
        self.art_generator = AsciiArtGenerator(self.config)

    def get_text_input(self):
        """Prompt the user to input text for ASCII art generation."""
        return input(USER_ASCII_TEXT_PROMPT)

    def get_font_choice(self):
        """Prompt the user to choose a font from available options."""
        return UserInput.get_input(FONT_CHOICE_PROMPT, self.config.fonts)

    def get_color_choice(self):
        """Prompt the user to choose a color from available options."""
        return UserInput.get_input(COLOR_CHOICE_PROMPT,
list(self.config.colors.keys()))

    def get_justification_choice(self):
```

```
        """Prompt the user to select text justification."""
        return UserInput.get_input(JUSTIFY_CHOICE_PROMPT,
self.config.justify)

    def get_width_input(self):
        """Prompt the user to input a width for the ASCII art."""
        return UserInput.get_integer_input(WIDTH_CHOICE_PROMPT, 10, 175)

    def generate_ascii_art(self):
        """Generate ASCII art based on user preferences for text, font, color, and
alignment."""
        text = self.get_text_input()
        font = self.get_font_choice()
        color = self.get_color_choice()
        justify = self.get_justification_choice()
        width = self.get_width_input()

        ascii_art = self.art_generator.generate_art(text, color, font, width, justify)

print(f"{self.config.colors.get(color)}{ascii_art}{self.config.colors.get('reset')}")

        self.prompt_to_save_ascii_art(ascii_art)

    def prompt_to_save_ascii_art(self, ascii_art):
        """Prompt the user to decide if the ASCII art should be saved."""
        save_choice = input(SAVE_CHOICE_PROMPT).strip().lower()
        if save_choice in ('yes', 'y'):
            self.save_ascii_art(ascii_art)
```

```
    def save_ascii_art(self, ascii_art):
        """Save ASCII art to a specified file."""
        filename = input(FILE_NAME_PROMPT).strip()
        if not filename.endswith('.txt'):
            filename += '.txt'

        FileOperations.save_art(ascii_art, self.config.file_path, filename)

    def start_ui(self):
        """Initiate the ASCII art generation process."""
        self.generate_ascii_art()
```

lab3_config.py

```
from src.classes.lab3.config import BaseConfig
from src.classes.lab4.custom_font import font
from src.classes.lab4.star_font import star_font
from src.const.path_constants import FOLDER_PATH_RESULT_LAB4


class Lab4Config(BaseConfig):
    def __init__(self):
        super().__init__()
        self.fonts.extend(["star", "custom"])
        self.custom_font = font
        self.star_font = star_font
        self.colors.update({
            'black': "\033[30m",
```

```
    'white': "\033[37m",
    'light_gray': "\033[37m",
    'dark_gray': "\033[90m",
    'reset': "\033[0m"
})
self.fixed_width = 175
self.file_path = FOLDER_PATH_RESULT_LAB4
```

Рис.1. Результат програми.

Висновок: на цій лабораторній роботі я вивчила створення додатка Генератора ASCII-арту без використання зовнішніх бібліотек