학과 : 기계시스템디자인공학과

학번 : 19100064

이름 : 김정효

제출일 : 2024/11/13

결과 보고서:  
그래픽 화투 게임

1. 요구사항

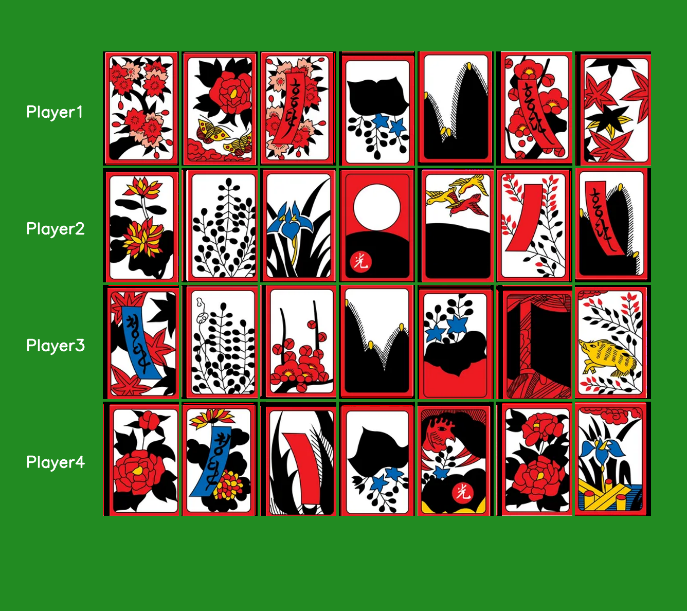
다음 그림과 같은 1188x1337 크기의 화투 이미지(flower\_cards.png 강의 자료실에 있음.)를 사용하여 프로그램을 진행한다.

* **실습 사이트에서 텍스트기반의 Card 예제를 해결한 적이 있다.**
* **Whatoo 클래스를 만들고 내부에 속성 deck 리스트를 만든다.**
* **클래스의 생성자는 카드 이미지를 읽어 들이고 numpy를 이용하여 그림을 48장으로 자르고 deck를 초기화하는 기능을 한다.**
* **shuffle 메서드는 카드를 섞는 것이다. deal 메서드는 4명의 선수에 7장씩 패를 나눠주는데 OpenCv 창에 아래 그림과 같이 카드들을 표시한다. 당연이 각 선수의 화투패는 중복이 되지 않도록 한다.**
* **기타 내용은 스스로 알아서 한다.**

2. 코드 실행 결과

<원본 이미지> <결과 이미지>

그래픽 디자인, 소설, 만화 영화, 일러스트레이션이(가) 표시된 사진

자동 생성된 설명

3. 코드 구조

코드 개요

Whatoo 클래스는 OpenCV(cv2)와 Numpy를 활용하여 화투 게임의 카드 이미지를 처리하고 시뮬레이션하는 기능을 제공합니다. 아래의 주요 메서드는 다음과 같은 역할을 수행합니다.

**\_init\_\_(self, img)**: 생성자 메서드로, 클래스 인스턴스가 생성될 때 호출됩니다. 입력으로 받은 이미지(img)를 기반으로 카드를 로드하고, 이를 self.deck 리스트에 저장합니다.

 **load\_cards(self, img)**: 전체 카드 이미지에서 개별 카드를 추출하여 self.deck에 저장합니다. 이미지의 크기를 1188x1337로 조정한 후, 각 카드를 8x6 그리드로 분할하여 리스트에 추가합니다.

 **shuffle\_deck(self)**: random.shuffle 함수를 사용하여 self.deck에 저장된 카드들의 순서를 무작위로 섞습니다.

 **deal\_cards(self)**: 섞인 덱에서 각 플레이어에게 7장씩 카드를 분배합니다. 4명의 플레이어를 가정하여, 각 플레이어의 카드 리스트를 반환합니다.

 **display\_player\_cards(self, players)**: 각 플레이어의 카드를 화면에 표시합니다. NumPy를 사용하여 배경 이미지를 생성하고, 각 플레이어의 카드를 해당 위치에 배치한 후, OpenCV의 cv2.imshow 함수를 통해 결과를 출력합니다.

 **write\_userName(self, posX, posY, playerName, background)**: 지정된 위치에 플레이어의 이름을 표시합니다. OpenCV의 cv2.putText 함수를 사용하여 배경 이미지 위에 텍스트를 추가합니다.

 **run\_game(self)**: 게임의 전체 흐름을 제어하는 메서드로, 덱을 섞고, 카드를 분배하며, 결과를 화면에 표시합니다.

 **save\_image(self, result\_image)**: 결과 이미지를 파일로 저장합니다. 현재 스크립트가 위치한 디렉토리에 'output.png' 파일로 저장하며, 저장 성공 여부를 출력합니다.

4. 전체 코드

import cv2

import numpy as np

import random

from PIL import Image

class Whatoo:

    def \_\_init\_\_(self, img):

        # 카드 이미지를 읽어오고 48장으로 분할

        self.deck = []

        self.load\_cards(img)

    def load\_cards(self, img):

        # 이미지 크기를 정확히 맞춤 (1188x1337으로 고정)

        img\_width = img.shape[0]

        img\_height = img.shape[1]

        print(img\_width, img\_height)

        img = cv2.resize(img, (1188, 1337))  # 이미지 크기를 1188x1337로 고정

        card\_height = img\_height // 8

        card\_width = img\_width // 6  # 각 카드의 크기: 167x198

        print(card\_width, card\_height)

        for col in range(8):

            for row in range(6):

                # 각 카드를 분할하여 정확한 크기로 deck에 추가

                card = img[row \* card\_width: (row + 1) \* card\_width,

                           col \* card\_height: (col + 1) \* card\_height]

                # 모든 카드를 정확히 167x198 크기로 조정

                #card = cv2.resize(card, (card\_width, card\_height))

                self.deck.append(card)

        print("Length: ", self.deck.\_\_len\_\_())

    def shuffle\_deck(self):

        random.shuffle(self.deck)

    def deal\_cards(self):

        players = [[] for \_ in range(4)]

        for i in range(4):

            players[i] = self.deck[i \* 7:(i + 1) \* 7]

        return players

    def display\_player\_cards(self, players):

        # 모든 카드가 동일한 크기인지 확인하고 배경 크기 설정

        card\_height, card\_width = players[0][0].shape[:2]  # 모든 카드가 167x198인지 확인

        background\_height = card\_height \* 4 + 300  # 4명의 플레이어, 세로 여백 포함

        background\_width = card\_width \* 7 + 300    # 각 플레이어당 7장의 카드, 가로 여백 포함

        # 배경을 정확히 계산된 크기로 설정

        background = np.zeros((background\_height, background\_width, 3), dtype=np.uint8)

        print("background : ",background\_height, background\_width)

        background[:] = (34, 139, 34)  # 녹색 배경 설정 (RGB 색상)

        x\_min = 200

        y\_min = 100

        for i, player\_cards in enumerate(players):

            for j, card in enumerate(player\_cards):

                y\_offset = i \* (card\_height + 5)

                x\_offset = j \* (card\_width + 5)

                background[y\_min + y\_offset : y\_min + y\_offset + card\_height, x\_min + x\_offset : x\_min + x\_offset + card\_width] = card

            self.write\_userName(50, (i+1) \* (card\_height + 5), f"Player{i+1}", background)

        cv2.imshow("Game Result", background)

        # 현재 폴더에 이미지 저장

        self.save\_image(background)

    def write\_userName(self, posX, posY, playerName, background):

        # 텍스트 설정

        text = playerName

        position = (posX, posY)  # 텍스트 위치 (x, y)

        font = cv2.FONT\_HERSHEY\_SIMPLEX  # 폰트 설정

        font\_scale = 1  # 폰트 크기 (크기 10에 맞도록 크기 조정 필요)

        color = (255, 255, 255)  # 흰색 텍스트 (BGR 형식)

        thickness = 2  # 텍스트 두께

        # 이미지에 텍스트 추가

        cv2.putText(background, text, position, font, font\_scale, color, thickness, cv2.LINE\_AA)

    def run\_game(self):

        self.shuffle\_deck()

        players = self.deal\_cards()

        self.display\_player\_cards(players)

        cv2.waitKey(0)

        cv2.destroyAllWindows()

    def save\_image(self, result\_image):

        import os

        # 현재 코드가 존재하는 폴더 경로

        current\_folder = os.path.dirname(os.path.abspath(\_\_file\_\_))

        # 저장 경로 설정

        output\_path = os.path.join(current\_folder, 'output.png')

        # 이미지 확장자 추출

        extension = os.path.splitext(output\_path)[1]

        # 이미지 인코딩

        result, encoded\_img = cv2.imencode(extension, result\_image)

        if result:

            with open(output\_path, mode='w+b') as f:

                encoded\_img.tofile(f)

            print(f"이미지가 {output\_path}에 저장되었습니다.")

        else:

            print("이미지 인코딩에 실패했습니다.")

# 한글 경로를 Pillow로 읽기

card\_image\_path = "c:/Users/aloho/문서/Github/openCV-SeoultechMSD/\_opencv\_example/00\_d/Flower\_cards.png"

img\_pil = Image.open(card\_image\_path)

# Pillow 이미지를 OpenCV 형식으로 변환

img = cv2.cvtColor(np.array(img\_pil), cv2.COLOR\_RGB2BGR)

# 게임 실행

game = Whatoo(img)

game.run\_game()