최근에 헝그리앱의 우마무스메 캐릭터 뽑기 시뮬레이터(링크)를 클론 코딩해 보고 있다.

좌측이 헝그리앱 원본, 우측이 클론 코딩한 화면

관련하여 찾아본 가챠 시뮬레이터 중 가장 퀄리티가 괜찮아 보이는데, 미호노 부르봉 픽업 이후 업데이트가 멈춘 것 같았다.
그래서 최신 픽업까지 업데이트해보고, 서포트 카드 뽑기와 무료 뽑기 기능까지 추가하는 것을 목표로 수정 중이다.

주된 구현 내용을 두 가지로 요약해 보면 다음과 같다.
1. 기존 PHP 서버 응답 코드를 로컬 자바스크립트 코드로 수정(추후 파이썬 API 형태로 변경해야 할 듯)
2. 서버에 저장된 정적 데이터 이용하던 것을 전부 로컬에 저장하여 사용하기 위해 경로를 적절히 수정

좌측이 헝그리앱 원본 이미지, 우측이 구현 중인 화면에서 사용한 인 게임 이미지

코드 구현은 의외로 크게 어렵지 않았는데, 이미지 파일 다듬는 것이 생각보다 고역이었다.
기존 이미지 파일은 화소가 적어 화질이 떨어지고, 1성과 2성 이미지의 경우 실제 가챠에서 나오는 이미지와 달랐다.
게다가 배너도 일본어로 되어 있어 전부 고화질인 인 게임 파일로 수정했다.

화질은 개선됐지만 화질을 얻게 되면서 놓치게 된 2성, 3성 캐릭터 프레임을 구현하기가 상당히 까다로웠다.
2성부터 프레임에 별 반짝임이 추가되고, 3성부터 무지갯빛이 도는데 이게 없으니 가챠가 주는 재미가 이전보다 떨어진다는 느낌이 들었다.
애초에 실제 인게임 내 가챠에서는 프레임에 애니메이션이 있어서 완전히 같게 따라 하기는 힘들더라도, 비슷하게나마 웹상에서 흉내 내보고 싶었다.

2성, 3성 프레임 이미지만 구하면 우마무스메 이미지의 등급에 따라 알맞은 프레임만 위에 얹어주면 되니 쉽지만, 그걸 구할 수가 없는 상황.
그래서 어떻게 하면 좋을까 고민하다가, 문득 파이썬을 이용할 수 있지 않을까 생각을 했다.
위처럼 같은 등급의 이미지들을 모아놓고 보면 프레임의 형태가 모두 같으니, 같은 RGBA 값을 가진 픽셀만 살리고 나머지는 투명화하면 프레임 이미지를 얻을 수 있지 않을까 생각한 것이다.

그렇게 작성하게 된 코드는 다음과 같다.

```python
# Python 3.10.0
from PIL import Image
import numpy as np

# open PNG file(RGBA)
img1 = Image.open('./1.png')
img2 = Image.open('./2.png')
# open PNG file(RGB)
# img1 = Image.open('./1.png').convert('RGB')
# img2 = Image.open('./2.png').convert('RGB')

# PIL image to NumPy array
# pix[h][w]: [R G B A]
pix1 = np.array(img1)
pix2 = np.array(img2)
pix3 = pix1

for h in range(img1.height):
    for w in range(img1.width):
        if np.array_equal(pix1[h][w], pix2[h][w]) == False:
            pix3[h][w][:] = 0 # pix[h][w] = [0 0 0 0]

# NumPy array to PIL image
img3 = Image.fromarray(pix3)
# Show image
img3.show()

# Ref
# https://supermemi.tistory.com/entry/Python-PIL-PIL-%EC%9D%B4%EB%AF%B8%EC%A7%80-Numpy-%
# http://www.tcpschool.com/css/css3_module_colors
```

코드와 같은 경로에 있는 1.png, 2.png 파일을 열고 이미지 파일의 rgba 값을 numpy array로 변환한 뒤, 각각 픽셀에서 동일한 것만 남기고 나머지는 색상의 투명도를 나타내는 알파 채널(alpha channel)을 비롯한 rgba 값을 0으로 만들어 투명하게 처리해 준다.
투명해야 우마무스메 이미지 위에 프레임을 얹을 수 있을 테니까.
그렇게 만들어진 픽셀 값이 담긴 numpy array를 다시 PIL image로 변환해 주고, 이를 보여주는 코드이다.
여기서 이미지를 보여주는 것에서 끝내지 않고 저장까지 하려면 show 대신 urllib.request.urlretrieve를 이용하면 된다.

이렇게 어찌어찌 프레임을 구하긴 했는데 결과물이 생각보다 구렸다.

상단 코드를 이용해 첫 번째 이미지와 두 번째 이미지의 공통된 픽셀만을 구한 것이 세 번째 이미지, 수제 3성 프레임이다.

뭔가... 뭔가 조금 아쉬웠다.
심지어 그림판 3D로 이미지 파일 하나에서 구한 프레임보다도 별로였다.

그래서 결국 그림판 3D로 따낸 프레임을 사용했는데, 맨 처음 사진에 있는 클론 코딩한 화면에 반영된 프레임이 그림판 3D로 만든 것이다.
보기에 그저 그래서, 그냥 둘 다 안 쓰기로 했다.

이미지 여러 장을 이용해 투명화 조건을 적당히 조정해가며 구해보면 좀 더 나은 프레임을 구할 수도 있었겠지만 그렇게까지 하기는 귀찮았다.
결국 프레임을 쪼개어 하단 별은 우마무스메 인벤 캐릭터 페이지처럼 CSS로 처리하고, 무지갯빛 테두리는 인 게임 이미지를 사용, 별 반짝임은 포기하기로 했다.

구현 결과물만 놓고 보면 코드를 써먹지 못했으니 헛수고가 됐지만, 누군가는 한 번쯤 이렇게 이미지 간 공통된 부분을 찾는 상황이 오지 않을까 해서 내용을 정리하여 공유해 본다.