Meteor Challenge

Enzo Magalhães

Answers:

Number of Stars	315
Number of Meteors	328
Meteors falling on the Water	105
(optional) Hidden Phrase	“It's not about how hard you hit. It's about how hard you can get hit and keep moving forward. How much you can take and keep moving forward" - Rocky Balboa/Sylvester Stallone

1. Introdução

- Linguagem de programação utilizada: Python

1.1 Pré-Processamento: Carregando a imagem

Antes de resolver as questões do desafio, foi necessário carregar a imagem como uma matriz de pixels RGBA. Foi utilizada a biblioteca PIL para isso:

```
from PIL import Image

img = Image.open("meteor_challenge_01.png")
matrix = img.load()
WIDTH, HEIGHT = img.size

print(f"W: {WIDTH}, H: {HEIGHT}")
print(f"Example pixel on position (0,0) in RGBA format: {matrix[0,0]}")
```

✓ 0.0s

```
W: 704, H: 704
Example pixel on position (0,0) in RGBA format: (2, 119, 189, 255)
```

2. Resultados

2.1 Número de Estrelas e Meteoros

Como as estrelas são mapeadas como pixels brancos puros (255,255,255,255 em RGBA) e os meteoros como pixels vermelhos puros (255,0,0,255 em RGBA) na imagem, o número de estrelas e meteoros pode ser facilmente extraído contando o número de pixels com essas cores. O seguinte código em Python faz exatamente isso:

```python
stars = 0
meteors = 0
STAR = (255,255,255,255)
METEOR = (255,0,0,255)

for c in range(WIDTH):
    for r in range(HEIGHT):
        pixel = matrix[c, r]
        if pixel == STAR:
            stars += 1
        elif pixel == METEOR:
            meteors += 1

print(f"Number of stars: {stars}")
print(f"Number of meteors: {meteors}")
```

✓ 0.0s

```
Number of stars: 315
Number of meteors: 328
```

Então cheguei ao resultado de que o número de estrelas é igual a 315, e o número de meteoros é igual a 328.

2.2 Número de Meteoros Caindo na Água

Se os meteoros estão caindo perpendicularmente ao solo, então aqueles que estão caindo na água são aqueles que estão diretamente acima dos pixels de água, ou, em outras palavras, meteoros que têm o mesmo valor de coluna que os pixels de água existentes.

Resolvi este problema inicialmente extraindo todas as posições horizontais (colunas) dos pixels de água e verificando, para cada meteoro, se sua posição horizontal é a mesma de qualquer pixel de água. Essa abordagem pode ser facilmente implementada percorrendo cada pixel na imagem de baixo para cima, para que os pixels de água no nível do solo sejam visitados e tenham suas posições de coluna armazenadas antes que as estrelas e os meteoros sejam visitados no loop. O código a seguir implementa isso:

```python
stars = 0
meteors = 0
water_meteors = 0
STAR = (255,255,255,255)
METEOR = (255,0,0,255)
WATER = (0,0,255,255)

water_cols = set()
for c in range(WIDTH):
    for r in reversed(range(HEIGHT)):
        pixel = matrix[c, r]
        if pixel == WATER and c not in water_cols:
            water_cols.add(c)
        elif pixel == STAR:
            stars += 1
        elif pixel == METEOR:
            meteors += 1
            if c in water_cols:
                water_meteors+=1

print(f"Number of stars: {stars}")
print(f"Number of meteors: {meteors}")
print(f"Number of meteors falling on the Water: {water_meteors}")
```

✓ 0.0s

```
Number of stars: 315
Number of meteors: 328
Number of meteors falling on the Water: 105
```

Então o número de meteoros caindo na água é de 105 meteoros.

2.3 Frase Escondida

Depois de explorar várias possibilidades de padrões nos pontos no céu, descobri que, para cada coluna na imagem, o número de estrelas e meteoros é de um ou zero. Então, coletei as contagens de estrelas e meteoros para cada coluna, seguindo um padrão binário:

```python
star_binary_phrase = []
meteor_binary_phrase = []

for c in range(WIDTH):
    col_star_count = 0
    col_meteor_count = 0
    for r in reversed(range(HEIGHT)):
        pixel = matrix[c, r]
        if pixel == STAR:
            col_star_count += 1
        elif pixel == METEOR:
            col_meteor_count += 1

    star_binary_phrase.append(str(col_star_count))
    meteor_binary_phrase.append(str(col_meteor_count))


star_binary_phrase = "".join(star_binary_phrase)
meteor_binary_phrase = "".join(meteor_binary_phrase)
print(f"star column counts: {star_binary_phrase}")
print(f"meteor column counts: {meteor_binary_phrase}")
```

✓ 0.0s

```
star column counts: 00100010010010010111010000100111011100110010000001101110011011101110100001000000110000101100010011011
meteor column counts: 01110010011101110110000101110010011001000010111000100000010010000110111101110110011000000110110101110
```

Depois disso, descobri que cada sequência de 8 bits representa um número que é um identificador de letra ASCII, então o código binário poderia ser traduzido convertendo cada sequência em sua letra correspondente para encontrar a mensagem escondida:

```python
star_binary_phrase = []
meteor_binary_phrase = []

for c in range(WIDTH):
    col_star_count = 0
    col_meteor_count = 0
    for r in reversed(range(HEIGHT)):
        pixel = matrix[c, r]
        if pixel == STAR:
            col_star_count += 1
        elif pixel == METEOR:
            col_meteor_count += 1

    star_binary_phrase.append(str(col_star_count))
    meteor_binary_phrase.append(str(col_meteor_count))

def get_phrase(binary_sequence):
    binary_sequence = "".join(binary_sequence)
    phrase = []
    for i in range(0, len(binary_sequence), 8):
        phrase.append(chr(int(binary_sequence[i : i + 8], 2)))

    return "".join(phrase)

print(f"star phrase: {get_phrase(star_binary_phrase)}")
print(f"meteor phrase: {get_phrase(meteor_binary_phrase)}")
```

✓ 0.0s

```
star phrase: "It's not about how hard you hit. It's about how hard you can get hit and keep moving fo
meteor phrase: rward. How much you can take and keep moving forward"  - Rocky Balboa/Sylvester Stallone
```

Com isso, podemos concluir que a frase final é:

> **"It's not about how hard you hit. It's about how hard you can get hit and keep moving forward. How much you can take and keep moving forward" - Rocky Balboa/Sylvester Stallone**

2.4 Código final e análise

Já que todos os problemas exigem percorrer os pixels da imagem, eles podem ser resolvidos em um único loop de altura * largura. O código final faz exatamente isso:

```
from PIL import Image

img = Image.open("meteor_challenge_01.png")
matrix = img.load()
WIDTH, HEIGHT = img.size
STAR = (255, 255, 255, 255)
METEOR = (255, 0, 0, 255)
WATER = (0, 0, 255, 255)

def get_phrase(star_binary_phrase, meteor_binary_phrase):
  star_binary_phrase = "".join(star_binary_phrase)
  meteor_binary_phrase = "".join(meteor_binary_phrase)

  star_phrase = []
  for i in range(0, len(star_binary_phrase), 8):
      star_phrase.append(chr(int(star_binary_phrase[i : i + 8], 2)))

  meteor_phrase = []
  for i in range(0, len(meteor_binary_phrase), 8):
      meteor_phrase.append(chr(int(meteor_binary_phrase[i : i + 8], 2)))

  return "".join(star_phrase) + "".join(meteor_phrase)


stars = 0
meteors = 0
water_cols = set()
water_meteors = 0

star_binary_phrase = []
```

```
meteor_binary_phrase = []

for c in range(WIDTH):
  col_star_count = 0
  col_meteor_count = 0
  for r in reversed(range(HEIGHT)):
      pixel = matrix[c, r]
      if pixel == WATER and c not in water_cols:
          water_cols.add(c)
      elif pixel == STAR:
          stars += 1
          col_star_count += 1
      elif pixel == METEOR:
          meteors += 1
          col_meteor_count += 1
          if c in water_cols:
              water_meteors += 1

  star_binary_phrase.append(str(col_star_count))
  meteor_binary_phrase.append(str(col_meteor_count))

hidden_phrase = get_phrase(star_binary_phrase, meteor_binary_phrase)
print(
  f"Number of stars: {stars}\nNumber of meteors: {meteors}\nNumber of
water        meteors: {water_meteors}\nHidden phrase: {hidden_phrase}"
)
```

A complexidade de tempo deste algoritmo é O(WH), e a complexidade de espaço é O(W), onde W = largura e H = altura da imagem.