Desafio do meteoro da Stealth.Ag

[CHALLENGE] [METEOR] Claudio de Souza Brito

[Sample] Answers:

Number of Stars	315
Number of Meteors	328
Meteors falling on the Water	105
(optional) Hidden Phrase	???

Questão 1:

Para contar o número de estrelas, eu só preciso percorrer todos os pixels da imagem e contar aqueles que são branco puro (RGB = [1,1,1]).

Questão 2:

Para contar o número de meteoros, eu só preciso percorrer todos os pixels da imagem e contar aqueles que são vermelho puro (RGB = [1,0,0]).

Questão 3:

Se a queda do meteoro é perpendicular, eu só preciso saber se existem pixels vermelhos e azuis que possuem a mesma coordenada y (se estão na mesma coluna). Eu poderia percorrer os pixels uma vez para adquirir todas as colunas que existem água, e depois percorrer mais uma vez, dessa vez vendo se a coluna dos meteoros batem com alguma coluna da água, mas isso seria muito ineficiente. Então percebi que quando a água aparece na imagem, já não tem mais meteoros, os elementos estão bem separados, portanto, decidi percorrer a imagem de trás para frente, adquirindo as colunas que tem água primeiro, para quando chegar nos meteoros, já poder fazer a verificação.

É possível obter as 3 primeiras questões com apenas uma percorrida nos pixels:

```
for l in reversed(range(numLines)):     #Percorrendo a imagem de trás para frente
    for c in reversed(range(numColums)):#do último pixel ao primeiro
        pixel = data[l][c]
        if pixel[0] == 0.0 and pixel[1] == 0.0 and pixel[2] == 1.0:#Se for um pixel azul puro, então é água
            water.append(c)                                        #Adicionar a coluna desse pixel na lista de colunas com água

        elif pixel[0] == 1.0 and pixel[1] == 0.0 and pixel[2] == 0.0:#Se for um pixel vermelho puro
            numMeteor += 1                                           #Aumenta a contagem de meteoros em 1
            if c in water:                                              #Se a coluna do meteoro está na lista de colunas com água
                meteorInWater += 1                                      #Aumenta a contagem de mateoros caindo na água em 1

        elif pixel[0] == 1.0 and pixel[1] == 1.0 and pixel[2] == 1.0:#Se for um pixel branco puro
            numStars += 1                                            #Aumenta a contagem de estrelas em 1
```

Questão 4 opcional:

Mesmo não conseguindo fazer, queria compartilhar minha ideia. Para visualizar o que os pontos (meteoros e estrelas) estavam dizendo, irei juntar os pontos para formar letras. Para cada ponto, eu iria atrás do ponto mais próximo a ele (usando distância euclidiana[1]), e iria desenhar um traço verde puro (RGB = [0,1,0]) entre esses dois pixels usando o algoritmo completo de Bresenham[2].

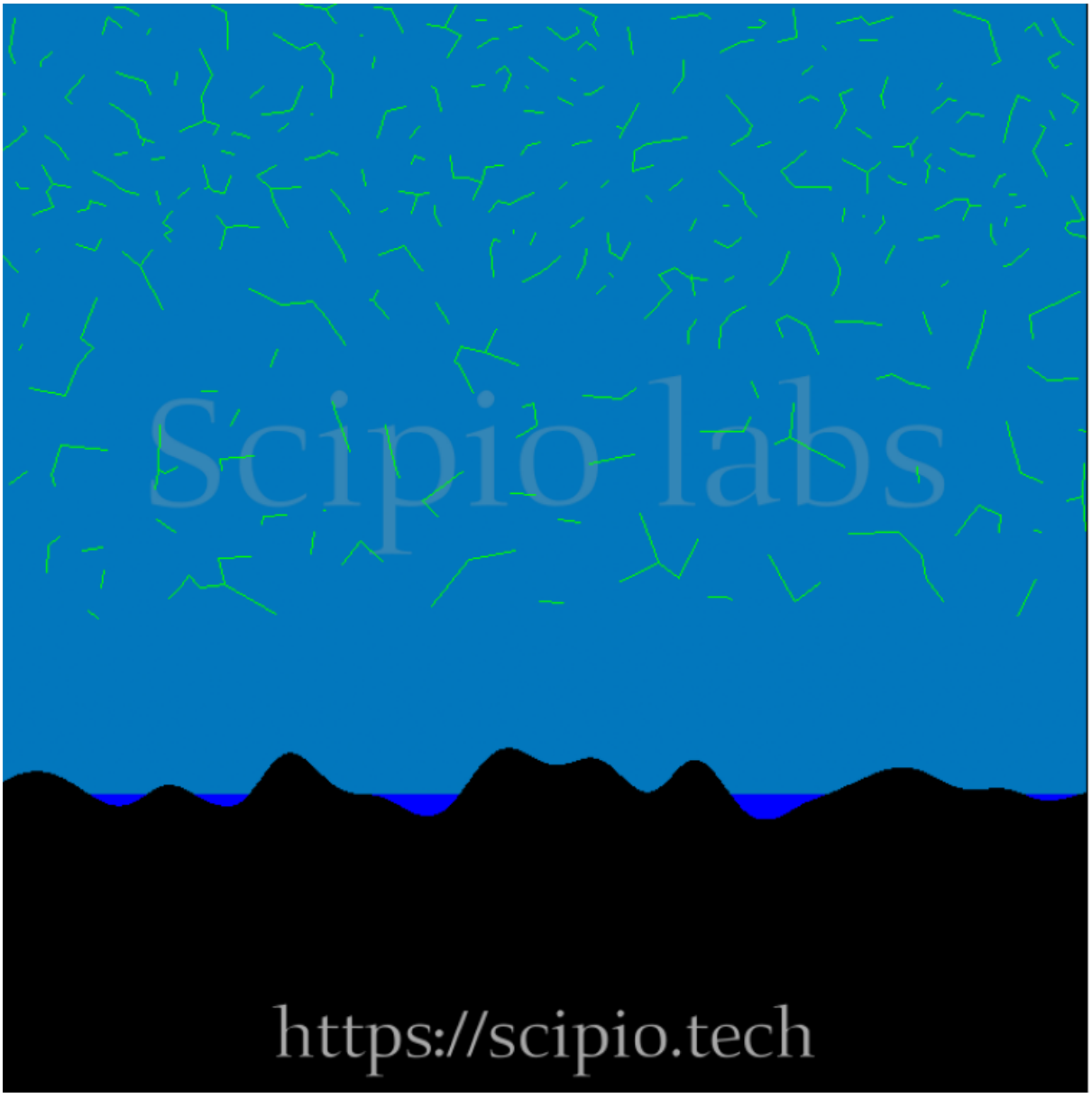
Mas antes eu precisava de uma lista com os pixels estrelas e meteoros e suas coordenadas, então criei uma classe para eles, e modifiquei os “for” das questões anteriores para não ter que percorrer novamente:

```
for l in reversed(range(numLines)):     #Percorrendo a imagem de trás para frente
    for c in reversed(range(numColums)):#do último pixel ao primeiro
        pixel = data[l][c]
        if pixel[0] == 0.0 and pixel[1] == 0.0 and pixel[2] == 1.0:#Se for um pixel azul puro, então é água
            water.append(c)                                        #Adicionar a coluna desse pixel na lista de colunas com água

        elif pixel[0] == 1.0 and pixel[1] == 0.0 and pixel[2] == 0.0:#Se for um pixel vermelho puro
            numMeteor += 1                                           #Aumenta a contagem de meteoros em 1
            if c in water:                                              #Se a coluna do meteoro está na lista de colunas com água
                meteorInWater += 1                                      #Aumenta a contagem de mateoros caindo na água em 1
            dots.append(Dot(l,c))                                    #Adiciona ponto na lista de pontos para desenhar linhas

        elif pixel[0] == 1.0 and pixel[1] == 1.0 and pixel[2] == 1.0:#Se for um pixel branco puro
            numStars += 1                                            #Aumenta a contagem de estrelas em 1
            dots.append(Dot(l,c))                                    #Adiciona ponto na lista de pontos para desenhar linhas
```

Acrescentando agora a parte exclusiva da questão 4:

```
data2 = copy.deepcopy(data)                                 #Copiando o array da imagem original

for pixel1 in dots:                                         #Para cada pixel na lista de pontos
    coord1 = np.array((pixel1.x, pixel1.y))                 #Pego suas coordenadas

    vizinhoProx = np.array((0,0))                           #Crio um vizinho com coordenadas zeradas
    menorDistancia = 9999999999                             #A menor distância é um número grande que será substituído logo na primeira iteração

    for pixel2 in dots:                                     #Para cada pixel na lista de pontos
        if pixel1 != pixel2:                                    #todos menos o primeiro pixel
            coord2 = np.array((pixel2.x, pixel2.y))                 #Pego as coordenadas dele
            dist = np.linalg.norm(coord1 - coord2)                  #Calculo a distância euclidiana entre ambos

            if dist < menorDistancia:                               #Se a distância for menor que a menor distância até agora
                menorDistancia = dist                                   #Atualiza a menor distância
                vizinhoProx[0] = coord2[0]                              #Atualiza as coordenadas do vizinho mais próximo
                vizinhoProx[1] = coord2[1]

    plotLine(data2, coord1[0], coord1[1], vizinhoProx[0], vizinhoProx[1])#Pinta o caminho entre os pixels
```

O resultado foi:



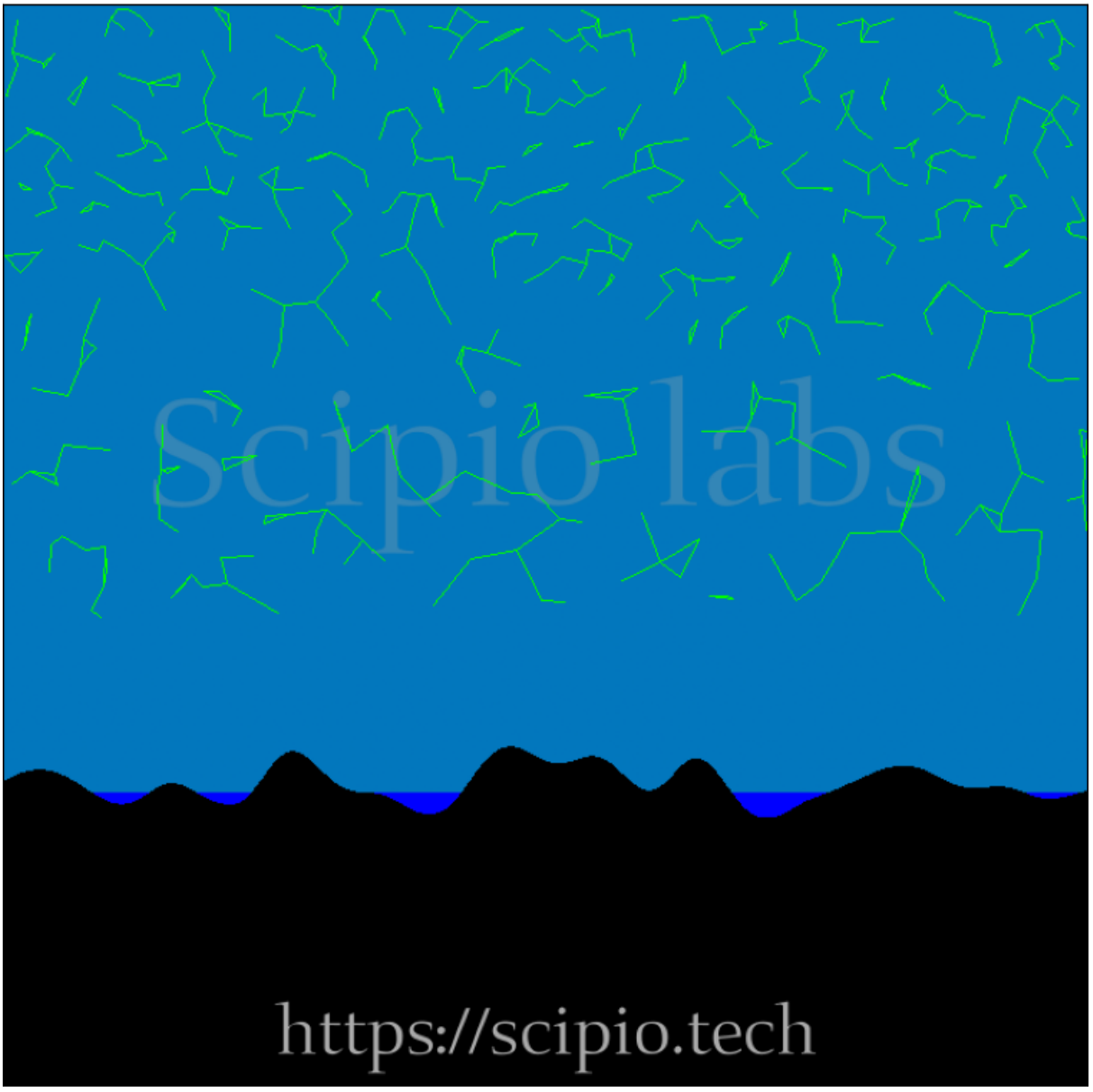
Além de ser ilegível, possui um problema, se o ponto A é o mais próximo de B e vice-versa, iremos traçar duas vezes a mesma coisa, desperdiçando alguns traços.

Então tive a ideia de colocar atributos na classe do ponto que armazena as coordenadas do ponto que foi considerado o mais próximo a ele. E o pixel A só é considerado para o cálculo de vizinho mais próximo de B se B não foi considerado o vizinho mais próximo de A antes. Isso força o programa a encontrar vizinhos mais longe formando mais traços

Sendo assim modificamos:

```
data2 = copy.deepcopy(data)                                         #Copiando o array da imagem original

for pixel1 in dots:                                                 #Para cada pixel na lista de pontos
    coord1 = np.array((pixel1.x, pixel1.y))                         #Pego suas coordenadas

    vizinhoProx = np.array((0,0))                                   #Crio um vizinho com coordenadas zeradas
    menorDistancia = 9999999999                                     #A menor distância é um número grande que será substituído logo na primeira iteração

    for pixel2 in dots:                                             #Para cada pixel na lista de pontos
        if pixel1 != pixel2 and (pixel2.duplaX != pixel1.x or pixel2.duplaY != pixel1.y):#Se os pixels são diferentes e se pixel2 não tem pixel1 como seu vizinho mais próximo
            coord2 = np.array((pixel2.x, pixel2.y))                         #Pego as coordenadas dele
            dist = np.linalg.norm(coord1 - coord2)                          #Calculo a distância euclidiana entre ambos

            if dist < menorDistancia:                                       #Se a distância for menor que a menor distância até agora
                menorDistancia = dist                                       #Atualiza a menor distância
                vizinhoProx[0] = coord2[0]                                  #Atualiza as coordenadas do vizinho mais próximo
                vizinhoProx[1] = coord2[1]

    plotLine(data2, coord1[0], coord1[1], vizinhoProx[0], vizinhoProx[1])   #Pinta o caminho entre os pixels
    pixel1.duplaX = vizinhoProx[0]                                          #Armazenando as informações do vizinho mais próximo no pixel1
    pixel1.duplaY = vizinhoProx[1]
```

E temos como resultado:



Temos mais traços, mas ainda é impossível de saber o que está escrito.

Referências:

[1] - Distância euclidiana: https://pt.wikipedia.org/wiki/Dist%C3%A2ncia_euclidiana
[2] - Algoritmo de Bresenham: https://en.wikipedia.org/wiki/Bresenham's_line_algorithm