* Meteor Challenge (Part 1)

Tasks:

1. Count the number of Stars
2. Count the number of Meteors
3. If the Meteors are falling perpendicularly to the Ground (Water level), count how many will fall on the Water
4. (optional) Find the phrase that is hidden in the dots in the sky.
    a. HINT 1: 177 Characters
    b. HINT 2: Most of the last tasks' code can be reused for this one

Please, send us the result and code you used to solve the tasks above. Explain how you achieved the results in each question. Good work!!

Subject: [CHALLENGE] [METEOR] Frederico de Rezende Casagrande

[Sample] Answers:

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

Pixel Ref:

(pure white) Stars
(pure red) Meteors
(pure blue) Water
(pure black) Ground

Como estava acontecendo um problema no carregamento da imagem por questão de segurança do cors do navegador, optei por instalar uma extensão do VS Code Live Server que permite simular um servidor, fazendo com que fosse possível rodar o software corretamente.

Explicando a lógica por trás do código

Primeiramente, fez se necessário encontrar alguma função que pudesse percorrer pixel a pixel da imagem e dizer a cor que aquele pixel representava. Para isso, o Typescript já tem funções inteligentes que nos garantem fazer isso, sendo que foi possível criar uma função que mapeou a imagem a partir de um desenho “drawImage”,após colocarmos ela inteiramente em 2D com a função getContext(‘2d’), a partir disso, foi criada uma constante para para alocar todos os dados referentes a essa imagem, que usou a função “getImageData” e tambem os pixels com a função “getPixels”, tudo isso foi para iniciarmos a classificação das cores da imagem.

Sendo assim, foi criada uma segunda função agora para coletar as cores desses pixels a partir de um for que percorre toda a imagem e aloca esses valores dentro de um array e, com isso, criou-se uma estrutura condicional para coletar o puro vermelho (rgb (255, 0, 0)), o puro branco (rgb(255, 255, 255)) e também o puro azul (rgb(0,0,255)) e quando esses cores eram encontradas, adicionava-se um novo valor para o meteoro no caso do vermelho, para a estrela no caso encontrasse branco e para a água caso fosse encontrado o azul, com isso foi possível coletar todas as estrelas, os meteoros e os pixels de água presentes na imagem.

Finalmente, para descobrir quais meteoros (red pixels) irão colidir com a água (blue pixels) foi necessário transformar a imagem em uma matriz de pixels 704 x 704, no qual caso a posição x do pixel vermelho, fosse igual a posição x do pixel azul no matriz ou no plano cartesiano, isso significa que haverá a colisão do meteoro com a água. Com isso, rodou-se um for dentro do outro para simularmos a matriz que condiz com as posições no eixo x de ambos, coletando cor por cor, além de fixar o eixo y dos pixels azuis no nível da água, já que para nossa sorte não havia água por baixo das pedras, o que dificultaria muito essa lógica. Após esses fors rodarem, foi se adicionando as coordenadas y que foram colidindo à variável “countWillFallOnTheWater”, fazendo assim com que fosse possível encontrar todos os meteoros que cairam na água.