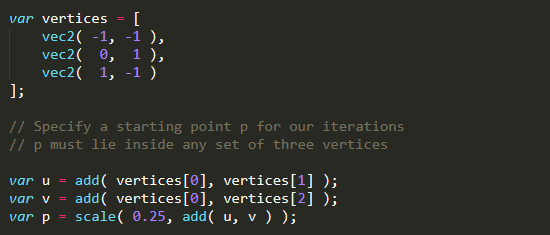
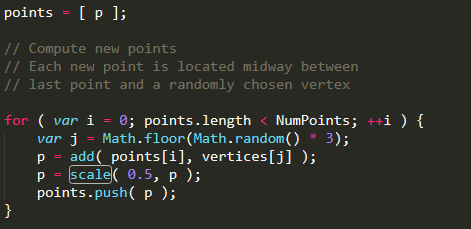
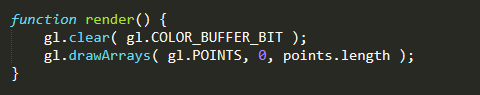
Gasket1



Cria um vetor vértices com os pontos. Os três pontos delimitam o triângulo externo. Para encontrar um ponto p dentro desse triângulo delimitado, somam-se os pontos e multiplica-se as coordenadas dos pontos por 0.25

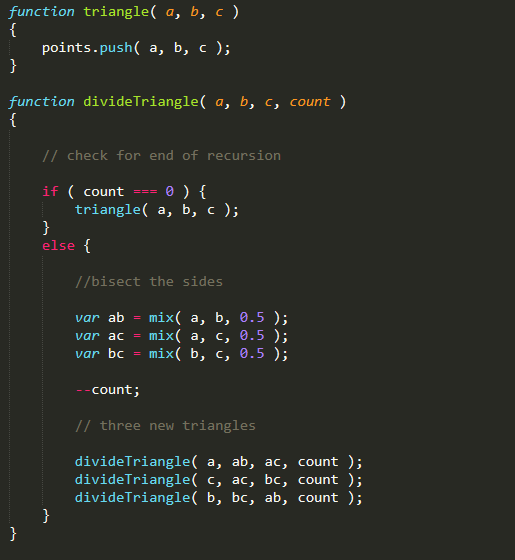


Cria-se um array e adiciona-se nele o ponto p. Cria-se um loop, seguindo o tamanho do vetor p até alcançar-se o número de pontos NumPoints definidos no início do programa. Pega-se aleatoriamente um dos pontos delimitadores do triângulo listados em vertices e soma-se com um elemento de points. Divide-se esse ponto por 0.5 (para que o ponto seja localizado entre o último ponto e o vértice aleatório) e ele é adicionado a points.



A função render é chamada e todos os pontos em points são renderizados usando o gl.POINTS

Gasket2

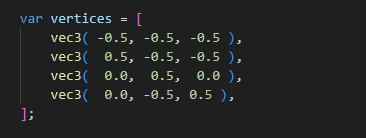


Funções do gasket2:

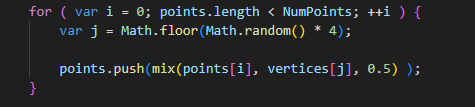
Triangle: adiciona à lista points três pontos que determinam um novo triângulo

divideTriangle: os parâmetros da função são os 3 pontos que determinam um triângulo e a contagem de vezes que os triângulos serão subdivididos. Conforme são encontrados os pontos que determinam os triângulos internos ao triângulo inicial e os triângulos gerados a partir dele, a contagem é decrescida. Assim como no gasket1 a função render é chamada após os shaders serem configurados, seus dados enviados à GPU, e os shaders associados aos buffers.

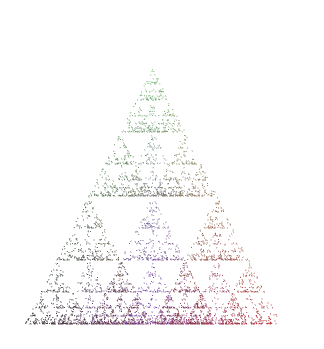
Gasket3:



Aqui são inicializados os vértices, e cada um com 3 dimensões, pois a intenção é que o gasket seja gerado em 3d. Nesse gasket os vértices são utilizados para delimitar os pontos obtidos posteriormente.

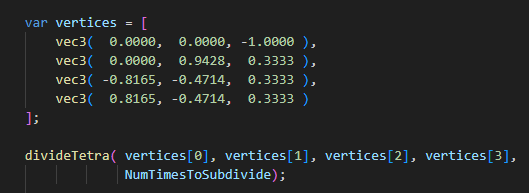


São adicionados pontos até que seja atingido o número de pontos determinados. Obtém-se uma variável j que vai de 0 a 4, arrendodada para o seu piso. É gerado um novo ponto realizando-se a soma do ponto e de um dos 4 vértices determinados inicialmente e dividindo-se essa soma por 2.

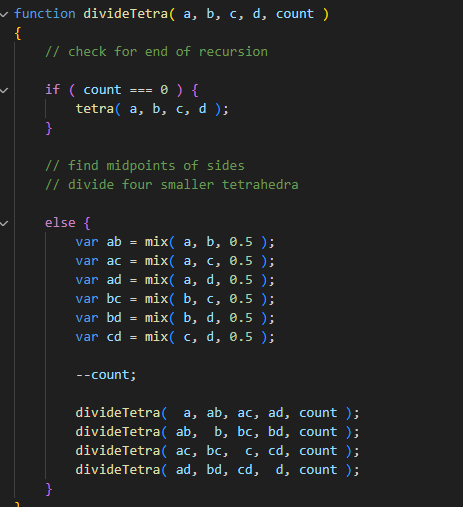


Gasket3 com 5000 pontos. Observa-se que os pontos tendem-se a se formar em triângulos

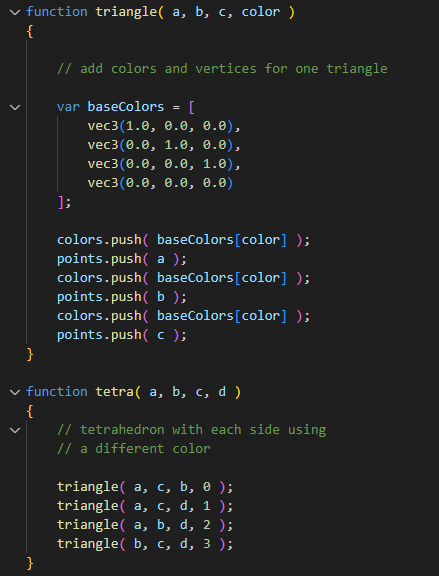
Gasket4:



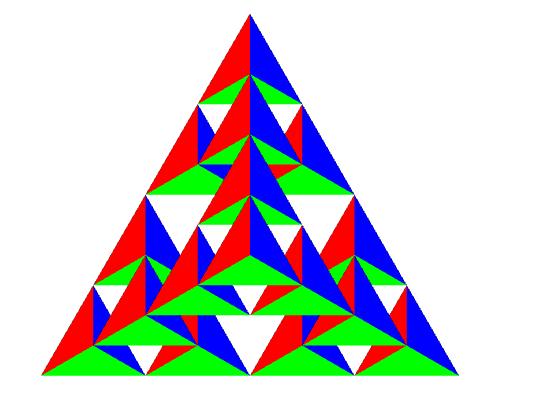
Gasket4 funciona de forma similar ao Gasket2, com a diferença de ser 3D e a função divideTetra formar tetraedros ao invés de triângulos.



A função funciona de forma similar ao divideTriangle do Gasket2, chamando recursivamente a função para criar novos tetraedros a partir dos pontos intermediários entre os pontos de outro triângulo.



No Gasket4 têm-se uma função triangle, que insere as cores nos vértices de cada tetraedro, a fim de deixar cada face dele com uma cor básica RGB.

Diminuindo-se a quantidade de subdivisões, observa-se o processo mais claramente, onde cada tetraedro tem seus vértices no vértice de outro.

Gasket5:

Gasket5 possui o mesmo processo do Gasket2, mas ele oferece a possibilidade de se alterar a quantidade de subdivisões por meio de um slider HTML.