CG – Shaders - Godot - Parte 2

COMPUTAÇÃO GRÁFICA

Godot

Os shaders Spatial têm mais funcionalidades integradas do que os shaders CanvasItem. A expectativa com os shaders spatial é que Godot já tenha fornecido a funcionalidade para casos de uso comuns e tudo o que o usuário precisa fazer no shader é definir os parâmetros adequados. Isso é especialmente verdadeiro para PBR (physically based rendering).

Godot

Em 3D, os objetos são desenhados usando Meshs. Meshs são um tipo de Resource que armazena geometria (a forma do seu objeto) e materiais (a cor e como o objeto reage à luz) em unidades chamadas faces. Uma malha pode ter múltiplas faces ou apenas uma.

O Godot também possui algumas PrimitiveMeshes que permitem adicionar geometria básica a uma cena sem importar meshs de outros lugares.

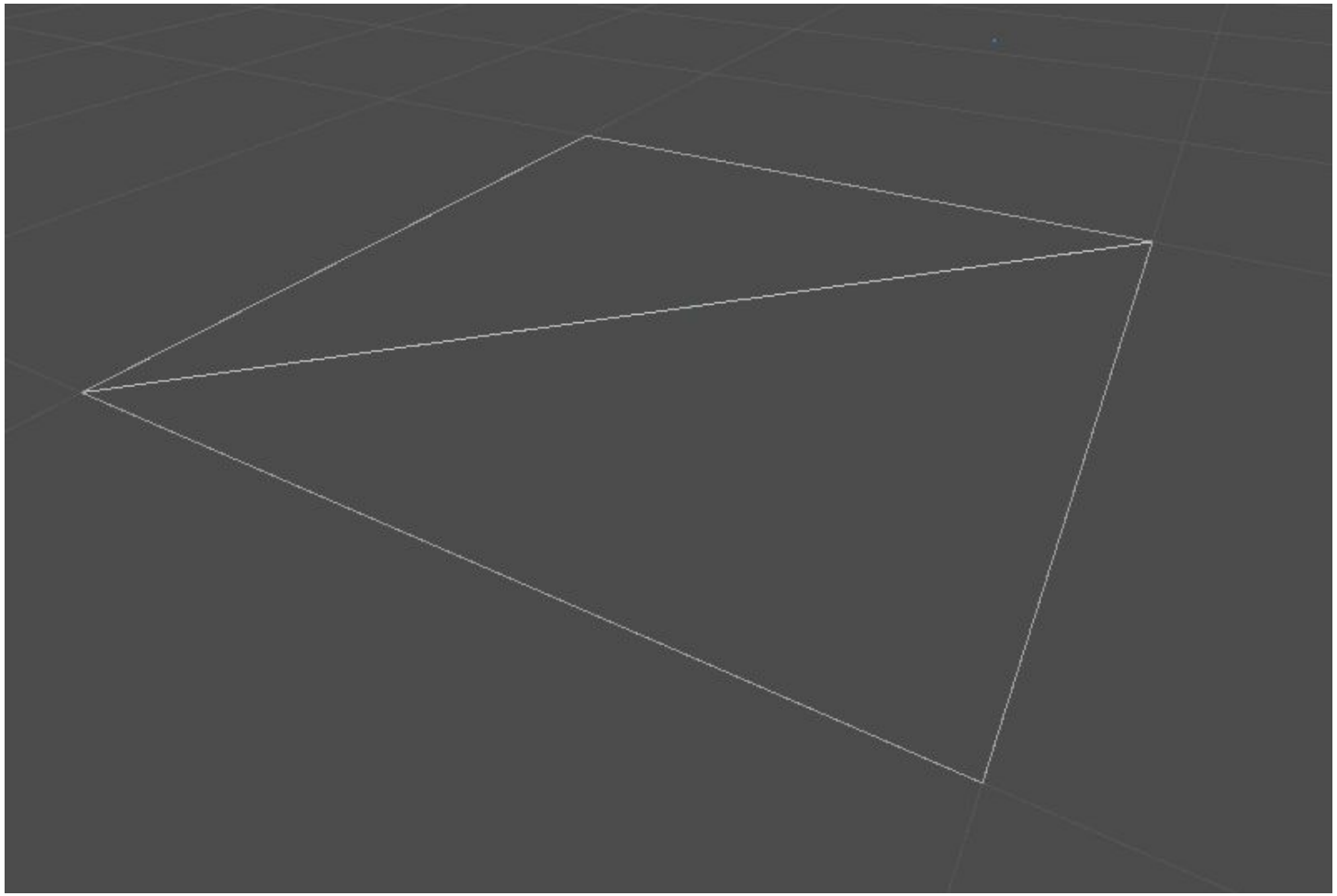
Godot - Começando

Crie uma cena 3D e adicione um novo Node MeshInstance3D à sua cena.

Na aba do inspetor ao lado de "Mesh", clique em "[vazio]" e selecione "Novo PlaneMesh". Em seguida, clique na imagem de um plano que aparece. Isso adiciona um PlaneMesh à nossa cena.

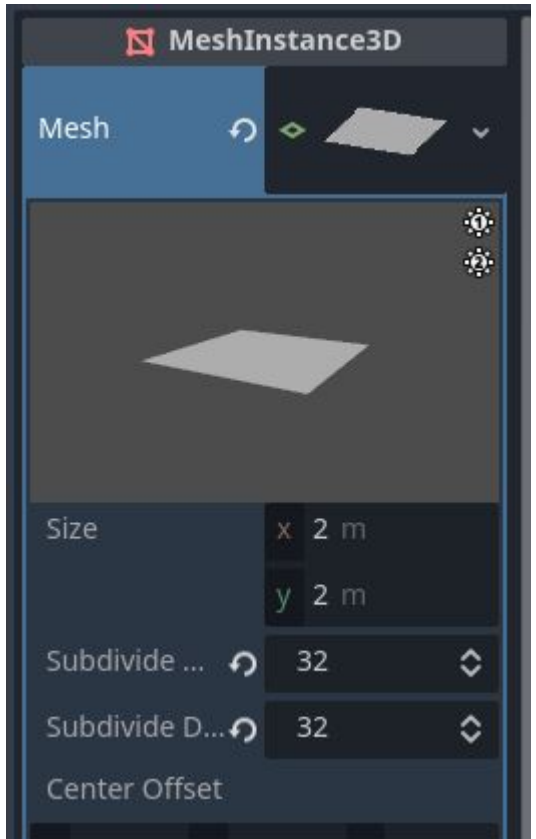
Em seguida, na janela de visualização principal, clique no canto superior esquerdo no botão que diz “Perspectiva”. Um menu aparecerá. No meio do menu estão opções de como exibir a cena. Selecione 'Exibir Wireframe'. Isso permitirá que você veja os triângulos que compõem o plano.

Godot - Começando

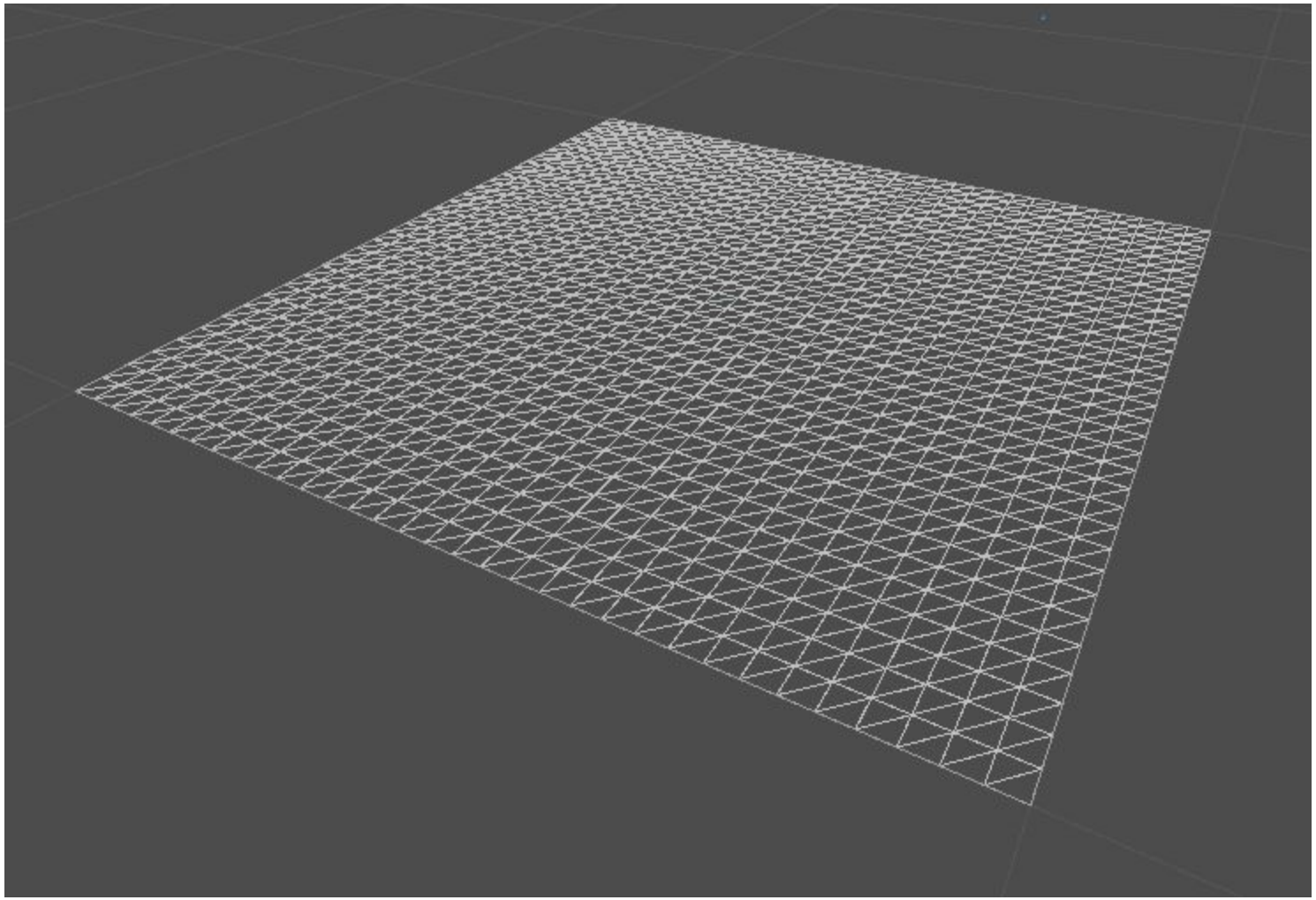


Godot - Começando

Agora modificamo o subdivide para 32x32 para aumentar o número de faces.



Godot - Começando



Godot - Começando

Clique ao lado de "Material" onde diz "[vazio]" e selecione "Novo ShaderMaterial". Em seguida, clique na esfera que aparece.

Agora clique ao lado de "Shader" onde diz "[vazio]" e selecione "Novo Shader".

Godot - Começando

O novo shader já foi gerado com uma variável shader_type e a função fragment() e vertex(). A primeira coisa que os shaders Godot precisam é de uma declaração de que tipo de shader eles são. Neste caso, shader_type é definido como spatial porque este é um shader spatial.shader_type espacial;

Por enquanto ignoramos a função fragment() e trabalharemos com a função vertex(). A função vertex() determina onde os vértices do MeshInstance3D aparecem na cena final. Iremos usá-lo para compensar a altura de cada vértice e fazer com que nossa superfície plana pareça um pequeno terreno.

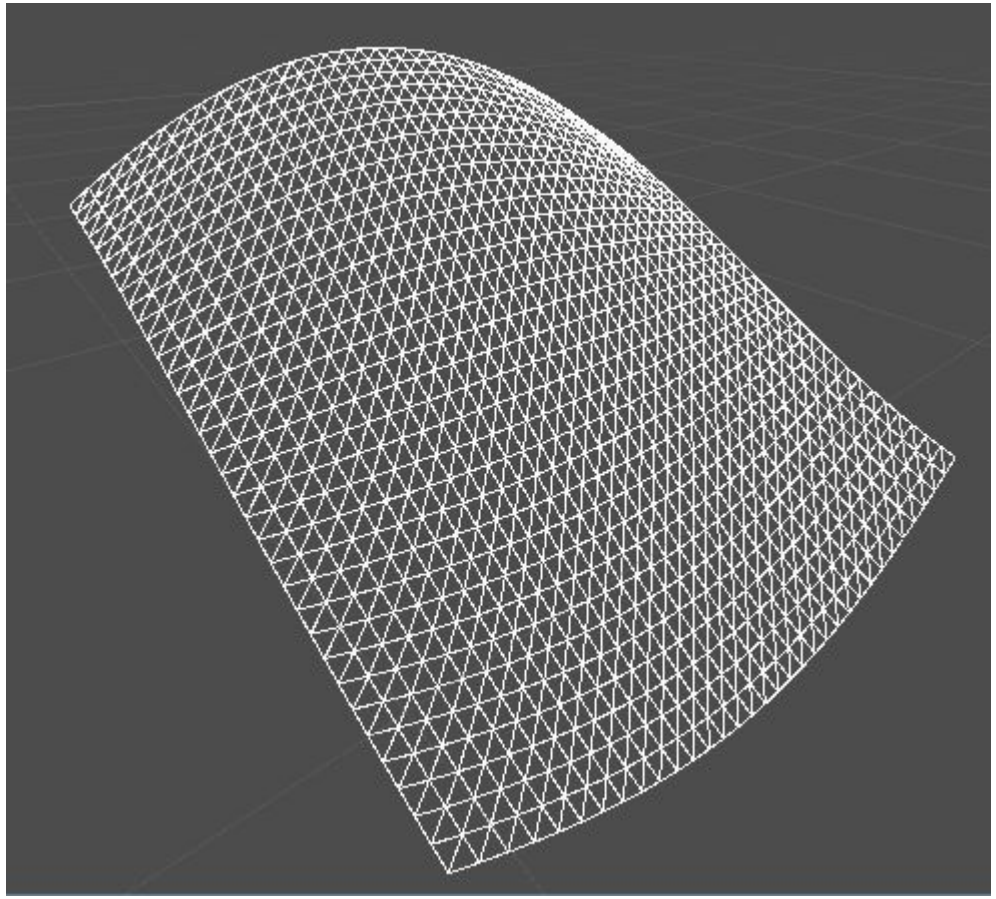
```
shader_type spatial;
```

Godot - Começando

Sem nada na função vertex(), Godot usará seu shader de vértice padrão. Podemos facilmente começar a fazer alterações adicionando uma única linha:

```
void vertex() {
  VERTEX.y += cos(VERTEX.x) * sin(VERTEX.z);
}
```

Godot - Começando



Godot - Começando

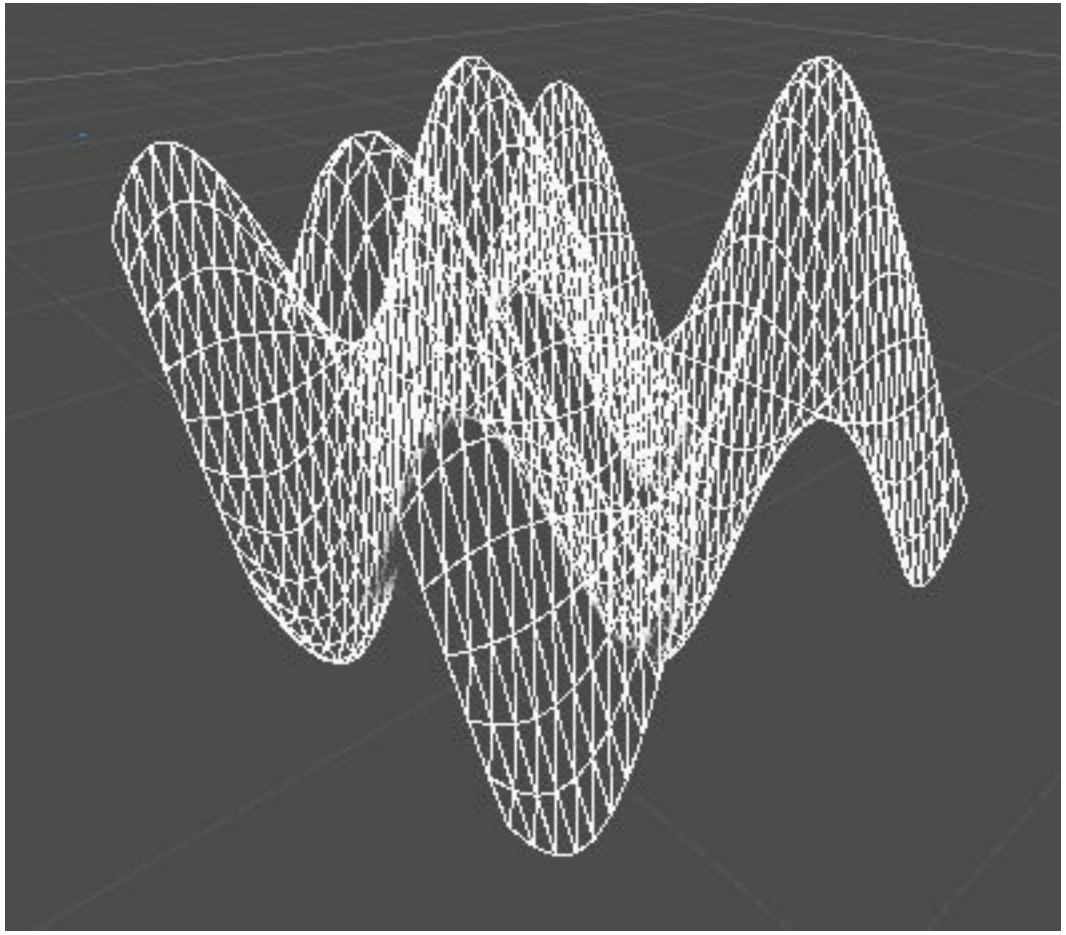
O valor y do VERTEX está sendo aumentado. E estamos passando os componentes x e z do VERTEX como argumentos para cos e sin; isso nos dá uma aparência de onda nos eixos x e z.

O que queremos alcançar é a aparência de pequenas colinas; afinal. coseno e o seno já parecem colinas. Fazemos isso escalonando as entradas para as funções cos e sin.

Godot - Começando

```
void vertex() {
  VERTEX.y += cos(VERTEX.x * 4.0) * sin(VERTEX.z * 4.0);
}
```

Godot - Começando



Godot - Ruído

O ruído é uma ferramenta muito popular para simular a aparência do terreno. Pense nisso como semelhante à função cosseno, onde você tem colinas repetidas, exceto que, com ruído, cada colina tem uma altura diferente. O Godot fornece o recurso NoiseTexture2D para gerar uma textura de ruído que pode ser acessada a partir de um shader.

Para acessar uma textura em um shader, adicione o seguinte código próximo ao topo do seu shader, fora da função vertex(). Isso permitirá que você envie uma textura de ruído para o shader.

```
uniform sampler2D noise;
```

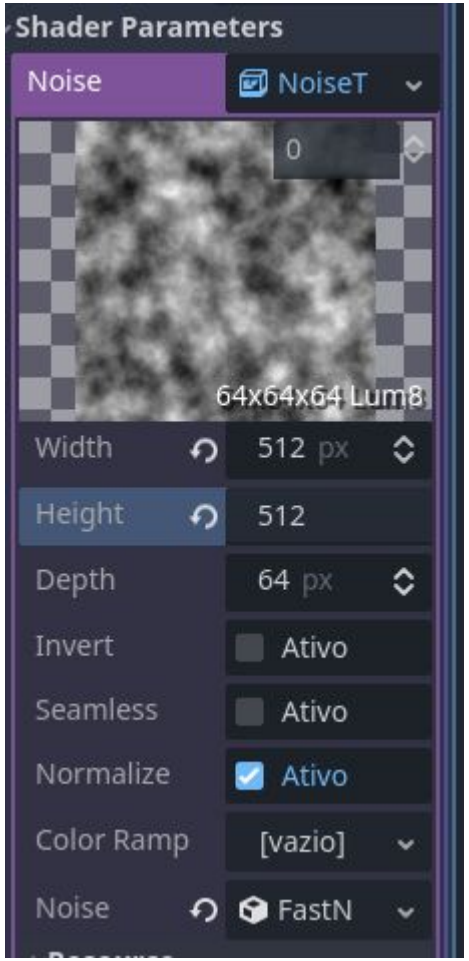
Godot - Ruído

Agora procure no inspetor seu material. Você deverá ver uma seção chamada "Shader Params". Se você abri-lo, verá uma seção chamada “Noise”.Clique ao lado onde diz "[vazio]" e selecione "Novo NoiseTexture2D".

Em seguida, em seu NoiseTexture2D, clique ao lado de onde diz "Noise" e selecione "Novo FastNoiseLite".

O FastNoiseLite é usado pelo NoiseTexture2D para gerar um mapa de altura. Depois de configurá-lo e deve ficar assim

Godot - Ruído


Shader Parameters
Noise
NoiseT
0
64x64x64 Lum8
Width
512 px
Height
512
Depth
64 px
Invert
Ativo
Seamless
Ativo
Normalize
Ativo
Color Ramp
[vazio]
Noise
FastN

Godot - Ruído

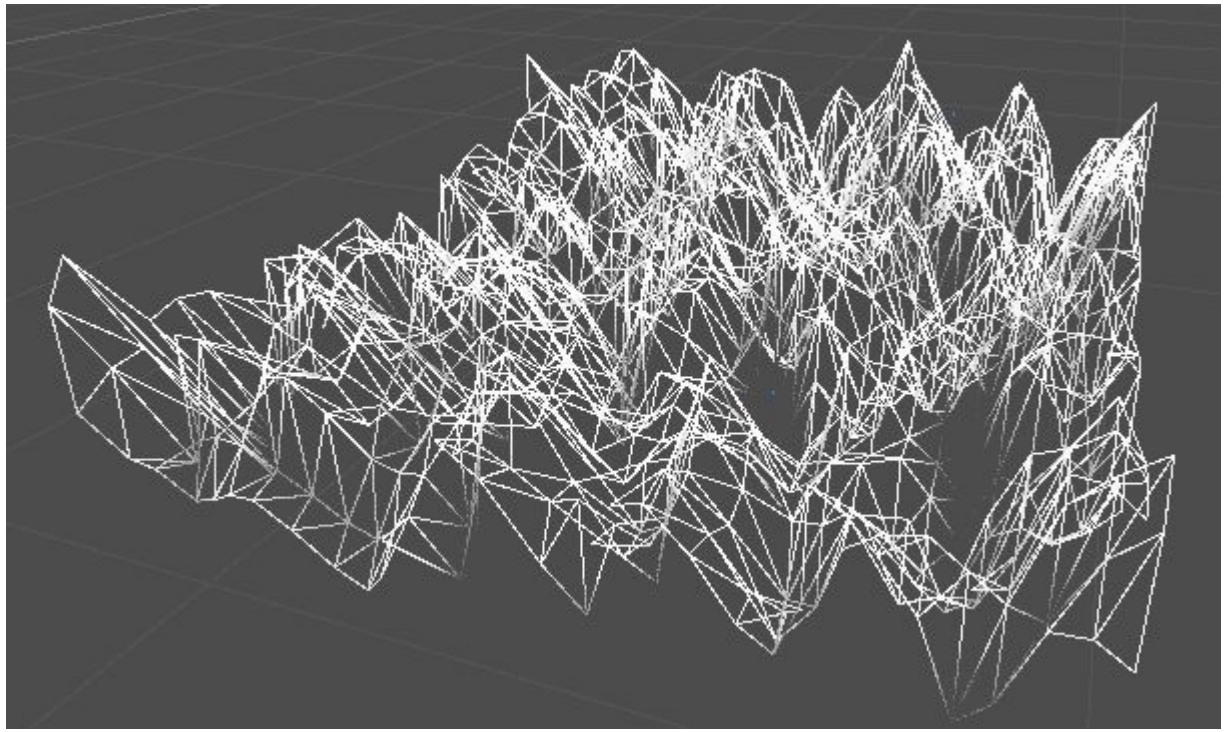
Agora, acesse a textura do ruído usando a função Texture(). Texture() recebe uma textura como primeiro argumento e um vec2 para a posição na textura como segundo argumento.

Usamos os canais x e z do VERTEX para determinar onde procurar na textura. Observe que as coordenadas do PlaneMesh estão dentro do intervalo [-1,1] (para um tamanho de 2), enquanto as coordenadas da textura estão dentro de [0,1], então para normalizar dividimos pelo tamanho do PlaneMesh por 2,0 e adicionamos 0,5. Texture() retorna um vec4 dos canais r, g, b, a na posição. Como a textura do ruído é em tons de cinza, todos os valores são iguais, então podemos usar qualquer um dos canais como altura. Neste caso usaremos o canal r ou x.

Godot - Ruído

```
void vertex() {
  float height = texture(noise, VERTEX.xz / 2.0 + 0.5).x;
  VERTEX.y += height;
}
```

Godot - Ruído



Godot - Ruído

Agora procure no inspetor seu material. Você deverá ver uma seção chamada "Shader Params". Se você abri-lo, verá uma seção chamada “Noise”.Clique ao lado onde diz "[vazio]" e selecione "Novo NoiseTexture2D".

Em seguida, em seu NoiseTexture2D, clique ao lado de onde diz "Noise" e selecione "Novo FastNoiseLite".

O FastNoiseLite é usado pelo NoiseTexture2D para gerar um mapa de altura. Depois de configurá-lo e deve ficar assim

Godot - Uniform

Variáveis uniforms permitem que você passe dados do jogo para o shader. Eles são muito úteis para controlar efeitos de shader. Os uniforms podem ser praticamente qualquer tipo de dados que possa ser usado no shader. Para usar um uniform, você o declara em seu shader usando a palavra-chave uniform. Vamos fazer um uniform que mude a altura do terreno.

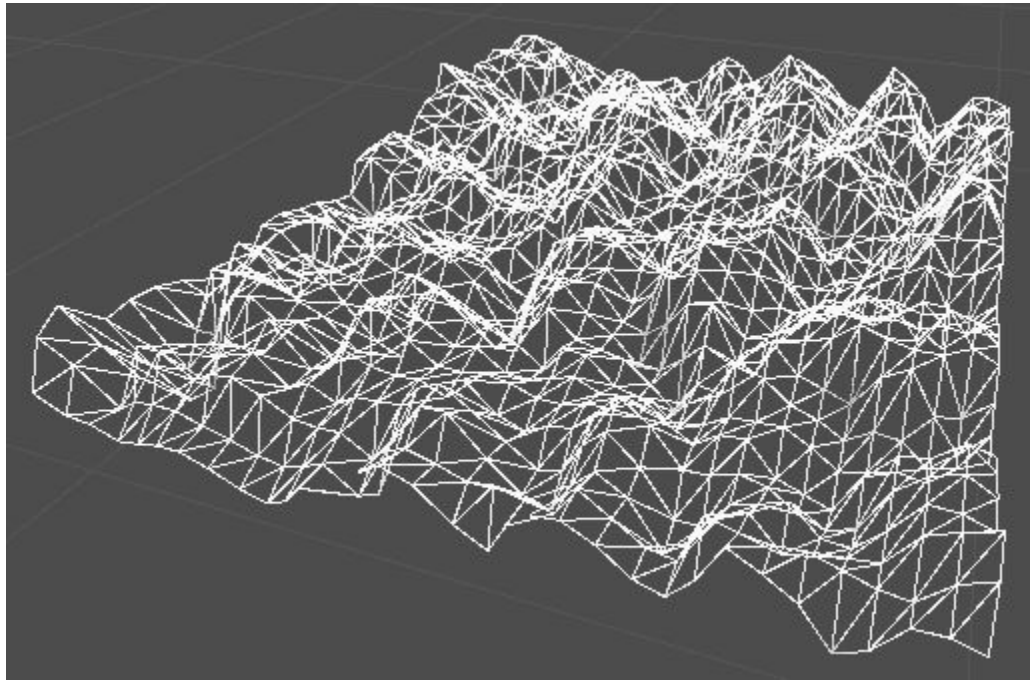
```
uniform float height_scale = 0.5;
```

Godot - Uniform

Godot permite inicializar um uniform com um valor; aqui, height_scale está definido como 0,5. Você pode definir uniforms do GDScript chamando a função set_shader_parameter() no material correspondente ao shader. O valor passado do GDScript tem precedência sobre o valor usado para inicializá-lo no shader. Você pode usar a variável uniforme em qualquer lugar dentro do seu Shader. Aqui, vamos usá-lo para definir o valor da altura em vez de multiplicar arbitrariamente por 0,5.

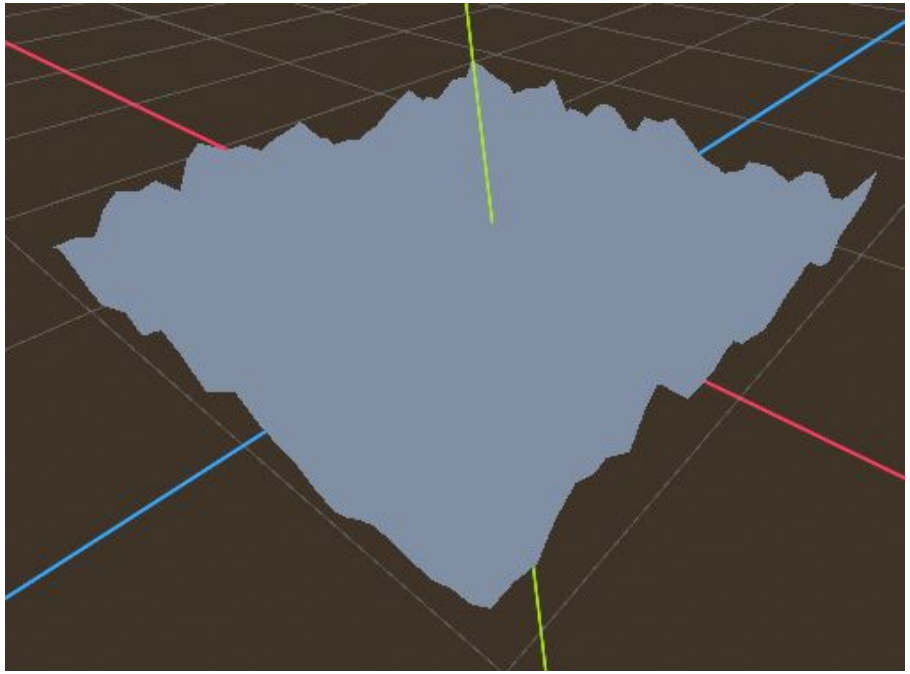
```
VERTEX.y += height * height_scale;
```

Godot - Uniform



Godot - Luz

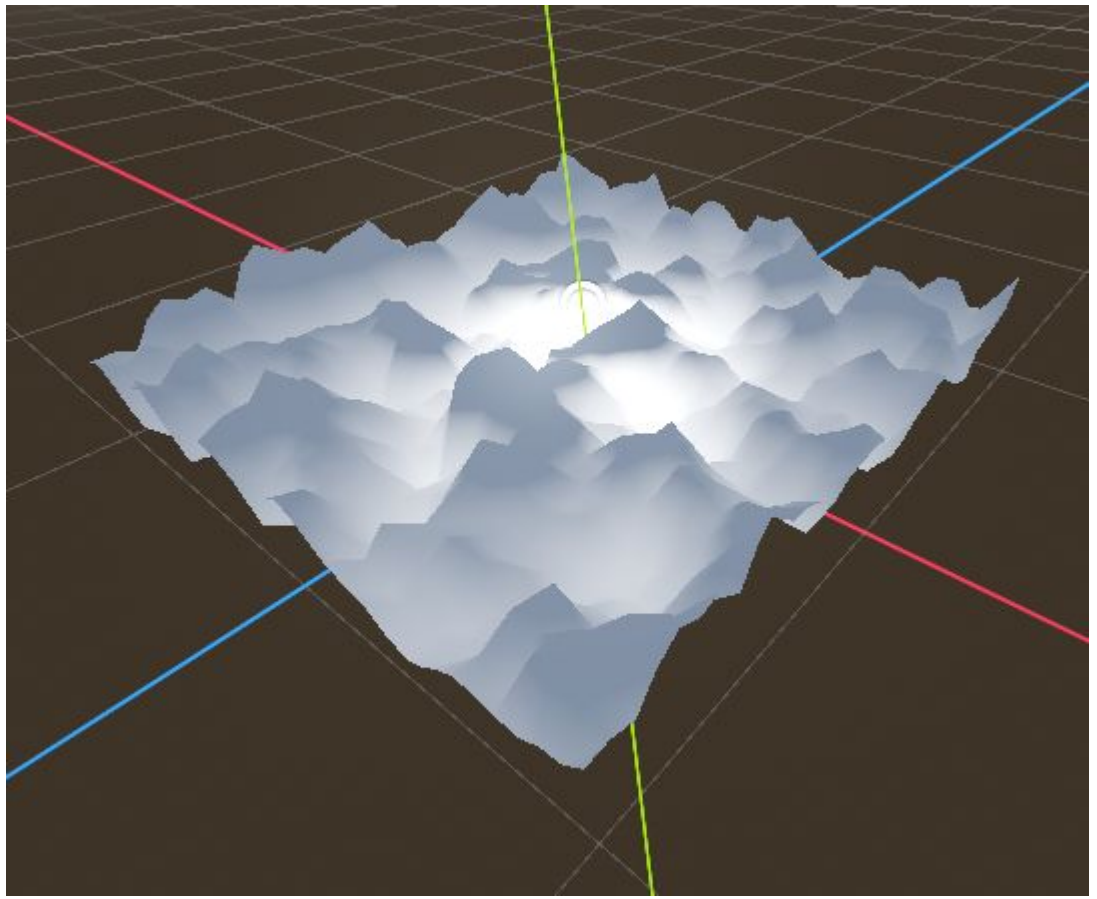
Primeiro, desligue o wireframe. Para fazer isso, clique novamente no canto superior esquerdo da janela principal, onde diz "Perspectiva", e selecione "Exibição Normal"



Godot - Luz

Observe como a cor da malha fica plana. Isso ocorre porque a iluminação é plana. Vamos adicionar uma luz! Primeiro, adicionaremos um OmniLight3D à cena. Você pode ver a luz afetando o terreno, mas parece estranho. O problema é que a luz afeta o terreno como se fosse uma superfície plana. Isso ocorre porque o light shader usa as normais da malha para calcular a luz. As normais são armazenadas na malha, mas estamos alterando o formato da malha no shader, então as normais não estão mais corretas.

Godot - Luz



Godot - Luz

Para corrigir isso, podemos recalcular as normais no shader ou usar uma textura normal que corresponda ao nosso ruído. Godot torna ambos fáceis para nós. Você pode calcular o novo normal manualmente na função de vértice e depois definir NORMAL. Com a configuração NORMAL, Godot fará todos os cálculos difíceis de iluminação para nós. Abordaremos esse método mais para frente, por enquanto leremos normais de uma textura. Contaremos novamente com NoiseTexture para calcular as normais para nós. Fazemos isso passando uma segunda textura de ruído.

```
uniform sampler2D normalmap;
```

Godot - Luz

Defina esta segunda textura uniform para outro NoiseTexture2D com outro FastNoiseLite. Mas desta vez, ative a opção As Normal map.

Godot - Luz

Agora, como este é um normal map e não um normal por vértice, vamos atribuí-lo na função fragment().

Quando temos normais que correspondem a um vértice específico definimos NORMAL, mas se você tiver um mapa normal que vem de uma textura, definimos o normal usando NORMAL_MAP.

Por último, para garantir que estamos lendo nos mesmos locais na textura de ruído e na textura do mapa normal, passaremos a posição VERTEX.xz da função vertex() para a função fragment(). Fazemos isso com varying.

Acima do vértice() defina um vec2 chamado tex_position. E dentro da função vertex() atribua VERTEX.xz a tex_position.

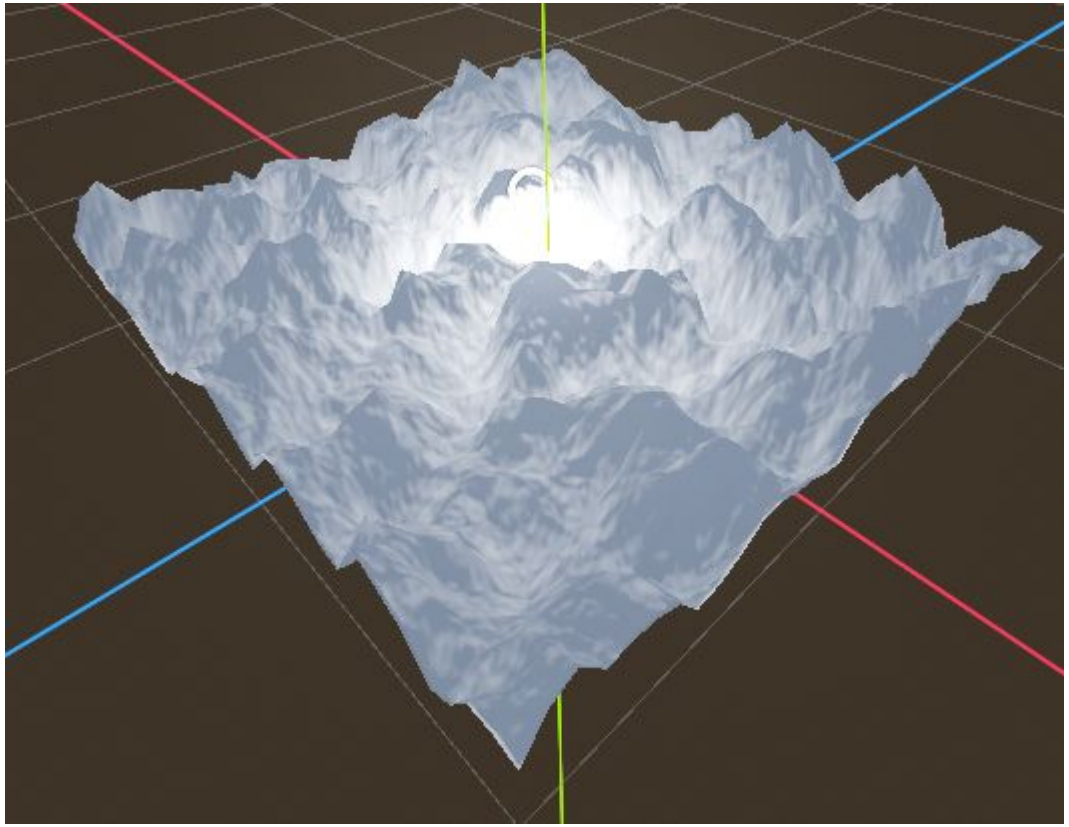
Godot - Luz

```
varying vec2 tex_position;

void vertex() {
  ...
  tex_position = VERTEX.xz / 2.0 + 0.5;
  float height = texture(noise, tex_position).x;
  ...
}
```

```
void fragment() {
  NORMAL_MAP = texture(normalmap, tex_position).xyz;
}
```

Godot - Luz



Godot - Luz