CG – Shaders - Godot

COMPUTAÇÃO GRÁFICA

Godot

O Godot é um motor de jogo de código aberto publicado no âmbito da licença MIT desenvolvido por sua própria comunidade. Foi usado internamente em várias empresas da América Latina antes de ter se tornado código aberto e lançado para o público. Tem a capacidade de exportar os projetos para diversas plataformas.

Godot

Utilizaremos a Godot, pois além de ser código aberto, não necessitamos de nenhum instalação, simplesmente baixamos o executável no site e podemos rodar.

https://godotengine.org

Shaders

Os shaders CanvasItem são usados para desenhar todos os objetos 2D em Godot, enquanto os shaders Spatial são usados para desenhar todos os objetos 3D.

Para usar um shader ele deve estar anexado dentro de um Material que deve estar anexado a um objeto.

Todos os objetos derivados de um CanvasItem possuem uma propriedade material. Isso inclui todos os elementos GUI, Sprite2Ds, TileMaps, MeshInstance2Ds etc. Eles também têm a opção de herdar o material de seus pais. Isso pode ser útil se você tiver um grande número de nós nos quais deseja usar o mesmo material.

Shaders

Para começar, crie um nó Sprite2D. Você pode usar qualquer CanvasItem, desde que esteja desenhando na tela, então para este tutorial usaremos um Sprite2D, pois é o CanvasItem mais fácil para começar a desenhar.

No Inspetor, clique ao lado de "Texture" onde diz "[vazio]" e selecione "Carregar" e selecione "Icon.svg". Para novos projetos, este é o ícone Godot. Agora você deve ver o ícone na janela de visualização.

Shaders

Em seguida, olhe para baixo no Inspetor, na seção CanvasItem, clique ao lado de "Material" e selecione "Novo ShaderMaterial". Isso cria um novo recurso material. Clique na esfera que aparece. Godot atualmente não sabe se você está escrevendo um CanvasItem Shader ou um Spatial Shader e visualiza a saída dos Spatial Shaders. Então o que você está vendo é a saída do Spatial Shader padrão.

Clique ao lado de "Shader" e selecione "New Shader". Por fim, clique no shader que você acabou de criar e o editor de shaders será aberto. Agora você está pronto para começar a escrever seu primeiro shader.

Shaders

No Godot, todos os shaders começam com uma linha especificando que tipo de shader eles são. Ele usa o seguinte formato:

```
shader_type canvas_item;
```

Shaders

Como estamos escrevendo um shader CanvasItem, especificamos canvas_item na primeira linha. Todo o nosso código ficará abaixo desta declaração.

Esta linha informa a engine quais variáveis e funcionalidades integradas devem fornecer a você.

No Godot você pode substituir três funções para controlar como o shader funciona; vertex, fragment e light.

Shaders

Hoje trabalharemos na criação de um shader com funções de vertex e fragment. As funções em light são significativamente mais complexas que as funções de vertex e fragment e não vamos ver ainda.

Shaders

A função de fragment é executada para cada pixel em um Sprite2D e determina a cor desse pixel.

Eles estão restritos aos pixels cobertos pelo Sprite2D, o que significa que você não pode usar um para, por exemplo, criar um contorno em torno de um Sprite2D.

Shaders

A função de fragment mais básica não faz nada exceto atribuir uma única cor a cada pixel.Fazemos isso escrevendo um vec4 na variável integrada COLOR.

COLOR é uma variável de entrada para a função de fragmento e a saída final dela.

```
void fragment(){
  COLOR = vec4(0.4, 0.6, 0.9, 1.0);
}
```

Shaders

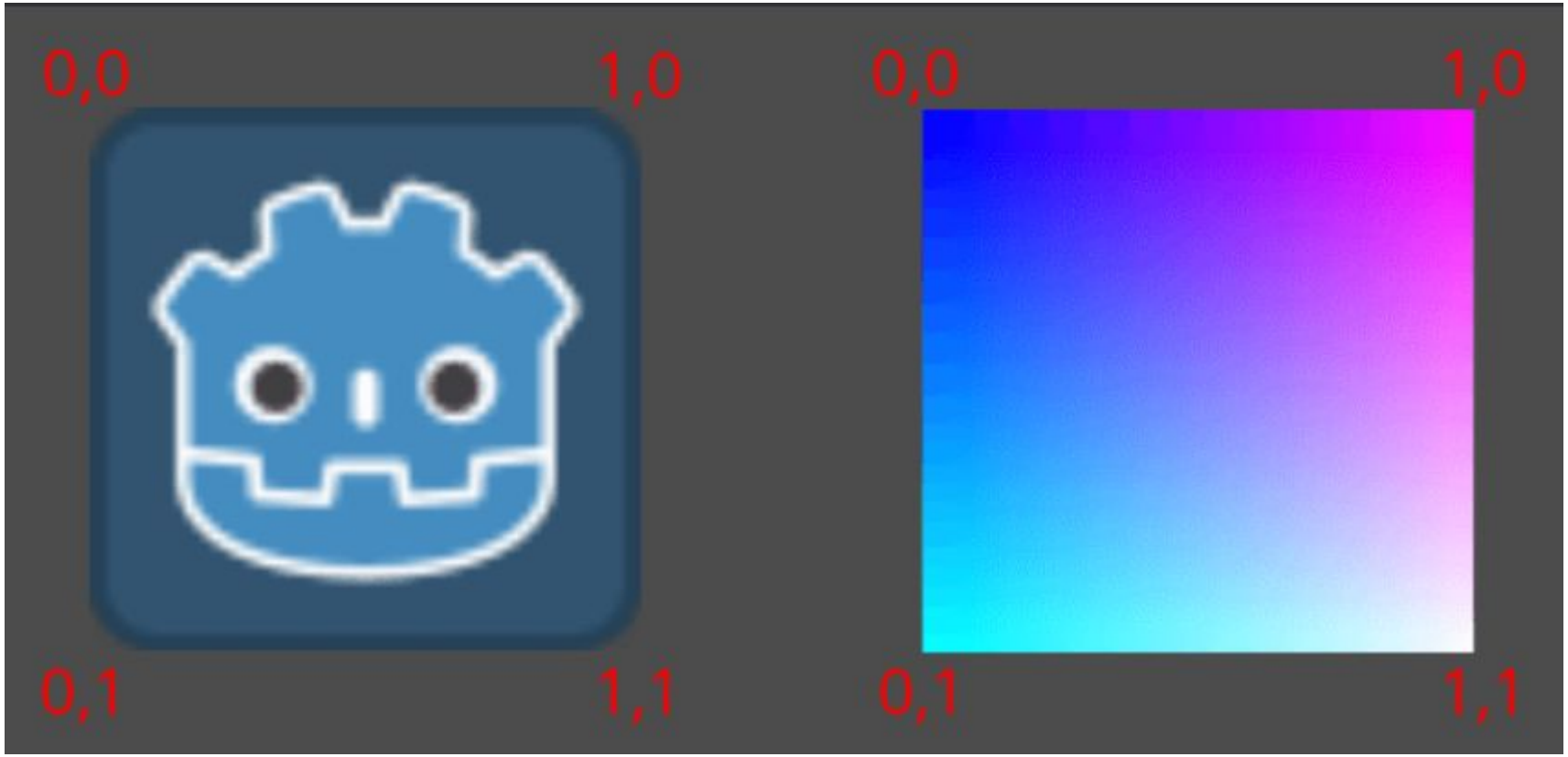
Existem muitas entradas para a função de fragment que você pode usar para calcular COLOR. UV é uma delas.

As coordenadas UV são especificadas em seu Sprite2D e informam ao shader onde ler as texturas para cada parte da malha.

Na função de fragment você só pode ler UV, mas pode usá-la em outras funções ou para atribuir valores diretamente a COLOR.

UV varia entre 0-1 da esquerda para a direita e de cima para baixo.

Shaders



Shaders

```
void fragment() {
  COLOR = vec4(UV, 0.5, 1.0);
}
```

Shaders

A função de fragment padrão lê uma textura e a exibe. Ao sobrescrever a função de fragment padrão, você perde essa funcionalidade e precisa implementá-la sozinho.

Você lê texturas usando a função de texture. Certos nós, como Sprite2Ds, possuem uma variável de textura dedicada que pode ser acessada no shader usando TEXTURE. Use-o junto com UV e textura para desenhar o Sprite2D.

Shaders

```
void fragment(){
  COLOR = texture(TEXTURE, UV); //read from texture
  COLOR.b = 1.0; //set blue channel to 1.0
}
```

Shaders

A entrada uniform é usada para passar dados para um shader que será o mesmo em todo o shader.

Você pode usar uniformes definindo-os na parte superior do seu shader assim:

```
uniform float size;
```

```
uniform float blue = 1.0; // you can assign a default value to uniforms

void fragment(){
  COLOR = texture(TEXTURE, UV); //read from texture
  COLOR.b = blue;
}
```

Shaders

Agora você pode alterar a quantidade de azul no Sprite2D no editor. Olhe novamente para o Inspetor onde você criou seu shader. Você deverá ver uma seção chamada "Shader Param".

Desdobre essa seção e você verá o uniform que acabou de declarar. Se você alterar o valor no editor, o valor padrão fornecido no shader será substituído.

Shaders

Você pode alterar os uniforms por código usando a função set_shader_parameter() que é chamada no recurso material do nó. Com um nó Sprite2D, o código a seguir pode ser usado para definir o uniform azul.

```
var blue_value = 1.0
material.set_shader_parameter("blue", blue_value)
```

Shaders

Usamos a função vertex para calcular onde na tela cada vértice deve terminar.

A variável mais importante na função de vertex é VERTEX. Inicialmente, ela especifica as coordenadas dos vértices em seu modelo, mas você também utiliza ela para determinar onde realmente desenhar esses vértices.

Shaders

VERTEX é um vec2 que é inicialmente apresentado no espaço local (ou seja, não relativo à câmera, viewport ou nós pais).

Você pode mover a posição dos vértices adicionando diretamente ao VERTEX.

```
void vertex() {
  VERTEX += vec2(10.0, 0.0);
}
```

Shaders

Combinado com a variável integrada TIME, isso pode ser usado para animação básica.

```
void vertex() {
  // Animate Sprite2D moving in big circle around its location
  VERTEX += vec2(cos(TIME)*100.0, sin(TIME)*100.0);
}
```

Shaders

Use a função de vértice para calcular onde na tela cada vértice deve terminar.A variável mais importante na função de vértice é VERTEX. Inicialmente, ele especifica as coordenadas dos vértices em seu modelo, mas você também escreve nele para determinar onde realmente desenhar esses vértices. VERTEX é um vec2 que é inicialmente apresentado no espaço local (ou seja, não relativo à câmera, viewport ou nós pais).Você pode compensar os vértices adicionando diretamente ao VERTEX.