Introdução aos Shaders

COMPUTAÇÃO GRÁFICA

Shaders

Shaders são escritos na linguagem C-like GLSL. O GLSL contém recursos úteis especificamente direcionados à manipulação de vetores e matrizes. Shaders sempre começam com uma declaração de versão, seguida por uma lista de variáveis de entrada e saída e sua função principal. O ponto de entrada de cada shader está em sua função principal, onde processamos quaisquer variáveis de entrada e produzimos os resultados em suas variáveis de saída. Um shader normalmente tem a seguinte estrutura:

Shaders

```
#version version_number
in type in_variable_name;
in type in_variable_name;
out type out_variable_name;
void main() {
  out_variable_name = weird_stuff_we_processed;
}
```

Shaders

Quando estamos falando especificamente sobre o shader de vértice, cada variável de entrada também é conhecida como atributo de vértice. Há um número máximo de atributos de vértice que podemos declarar limitados pelo hardware. O OpenGL garante que sempre há pelo menos 16 atributos de vértice de 4 componentes disponíveis, mas alguns hardwares podem permitir mais que você pode recuperar consultando GL_MAX_VERTEX_ATTRIBS.

Shaders

GLSL tem, como qualquer outra linguagem de programação, tipos de dados para especificar com que tipo de variável queremos trabalhar. GLSL tem a maioria dos tipos básicos padrão que conhecemos de linguagens como C: int, float, double, uint e bool. O GLSL também apresenta dois tipos de contêiner que usamos muito, vetores e matrizes.

Shaders

Um vetor em GLSL é um contêiner de 2,3 ou 4 componentes para qualquer um dos tipos básicos.

Eles podem assumir a seguinte forma (n representa o número de componentes):

- vecn: o vetor padrão de n floats.
- bvecn: um vetor de n booleanos.
- ivecn: um vetor de n inteiros.
- uvecn: um vetor de n inteiros sem sinal.
- dvecn: um vetor de n componentes double.

Na maioria das vezes, usaremos o vecn básico, pois os floats são suficientes para a maioria dos nossos propósitos. Componentes de um vetor podem ser acessados via vec.x onde x é o primeiro componente do vetor. Você pode usar .x, .y, .z e .w para acessar seu primeiro, segundo, terceiro e quarto componentes, respectivamente. GLSL também permite usar rgba para cores ou stpq para coordenadas de textura, acessando os mesmos componentes. O tipo de dados vetorial permite uma seleção de componentes interessante e flexível chamada swizzling. Swizzling nos permite usar sintaxe como esta:

Shaders

```
vec2 someVec = vec2(1.0, 2.0);
vec4 differentVec = vec4(1.0, 2.0, 1.0, 5.0);
vec4 differentVec = vec4(someVec.rgr, 5.0);
vec3 anotherVec = differentVec.zyw;
vec4 otherVec = someVec.xxxx + anotherVec.yxzy;
```

Shaders

Você pode usar qualquer combinação de até 4 letras para criar um novo vetor (do mesmo tipo) desde que o vetor original tenha esses componentes; não é permitido acessar o componente .z de um vec2 por exemplo. Também podemos passar vetores como argumentos para diferentes chamadas de construtores de vetores, reduzindo o número de argumentos necessários:

Shaders

```
vec2 vect = vec2(0.5, 0.7);
vec4 result = vec4(vect, 0.0, 0.0);
vec4 otherResult = vec4(result.xyz, 1.0);
```

Shaders

Você pode usar qualquer combinação de até 4 letras para criar um novo vetor (do mesmo tipo) desde que o vetor original tenha esses componentes; não é permitido acessar o componente .z de um vec2 por exemplo. Também podemos passar vetores como argumentos para diferentes chamadas de construtores de vetores, reduzindo o número de argumentos necessários: