UFABC

Computação Gráfica

André Brandão

Aula 06

Aula prática

Vertex Buffer Objects e Shaders

Slides de autoria do

Professor André Balan

UFABC

Vertex Buffer Objects

Aula prática

- Trabalhar com Vertex Buffer Objects e Shaders
- Ao invés de se trabalhar com glVertex, como foi feito na Aula 02, primeiro, colocaremos os vértices como VBO.
- Remover o código de glVertex do loop principal.

Vertex Buffer Object

- Inserir o código abaixo, após if(glewInit()!=GLEW_OK)

```
float v[] = {
          +0.0, +0.5,
          -0.5, -0.5,
          +0.5, -0.5
    };

GLuint VBO1;
glGenBuffers(1, &VBO1);
glBindBuffer(GL_ARRAY_BUFFER, VBO1);
glBufferData(GL_ARRAY_BUFFER, sizeof(v), v, GL_STATIC_DRAW);
```

- É importante que o código do VBO esteja ANTES do loop while.

Vertex Buffer Object

- Precisamos definir e habilitar o ponteiro de vértices.
- O código a seguir deve, também, ser inserido antes do loop while, logo após as definições do VBO.

```
glVertexPointer(2, GL_FLOAT, 0, nullptr);
glEnableClientState(GL_VERTEX_ARRAY);
```

- Isso também deve ser feito uma única vez no programa, antes do loop while, abaixo das definições do VBO.

Vertex Buffer Object

- Finalmente, invocamos a função para desenhar a primitiva (neste caso, um triângulo)

```
while (...) {
 glClearColor(0.0, 0.0, 0.0, 1.0);
 glClear(GL_COLOR_BUFFER_BIT);

 /* Código de renderização OpenGL vai aqui */
 glDrawArrays(GL_TRIANGLES, 0, 3);

 /* Troca o buffer de fundo com o buffer de exibição */
 glfwSwapBuffers(window);

 glfwPollEvents();
}
```

Shaders

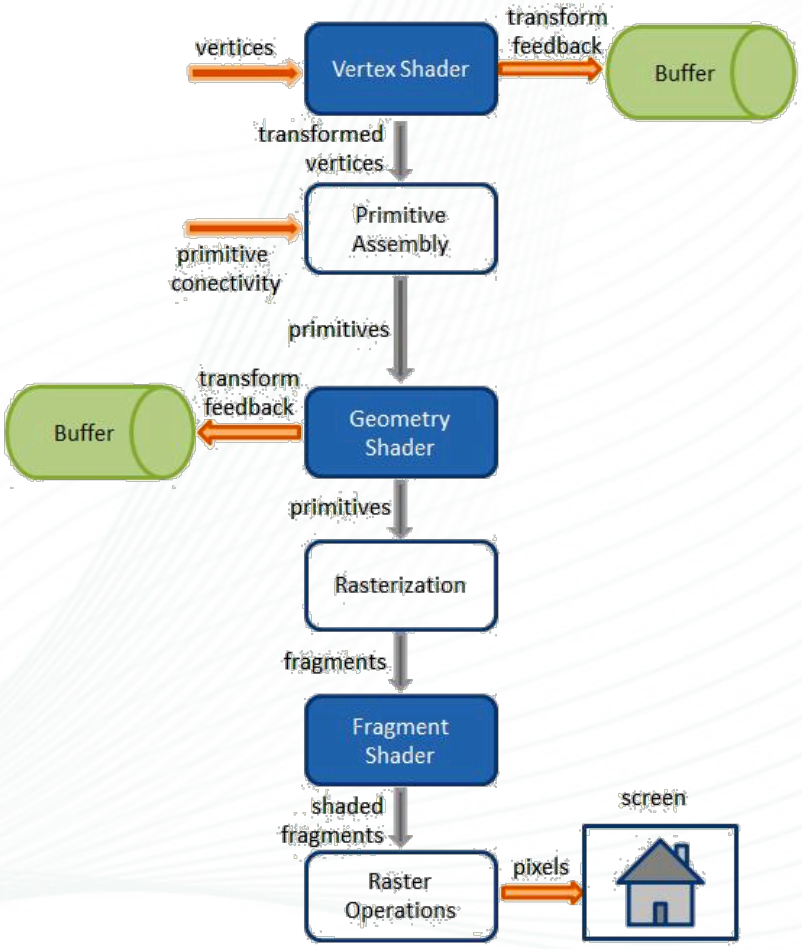
Slides de autoria do
Professor André Balan

UFABC

Shaders

- Podemos programar duas etapas principais do Pipeline:
  - Vertex shader
  - Fragment shader

OpenGL Pipeline



Shaders

- Inserir os seguintes includes:

```
#include <fstream>
#include <sstream>
```

- Trabalharemos com arquivos

Shaders

- Código para ler de um arquivo o programa shader em GLSL
- Criar a função readFile (inserir o código antes da função main)

```
string readFile(string filename) {
    ifstream t(filename);
    stringstream buffer;
    buffer << t.rdbuf();
    return buffer.str();
}
```

- Poderia fazer o programa dos shaders hard-coded no próprio código do programa...

Shaders

- Criar os arquivos:

  vertex.sdr

  fragment.sdr

- Os arquivos podem ser criados por meio de um editor de textos, por exemplo, o Bloco de notas.

Shaders

- Os Shaders em GLSL
  - Vertex Shader

```
#version 330 core
in vec4 vPosition;
void main() {
   gl_Position = vPosition;
}
```

  - Fragment Shader

```
#version 330 core
out vec4 fColor;
void main() {
   fColor = vec4(1.0, 0.0, 1.0, 1.0);
}
```

Shaders

- Lendo os arquivos de programas GLSL e colocando-os em strings.
- Inserir o código a seguir, após as definições do VBO.

```
string vss = readFile("vertex.sdr");
string fss = readFile("fragment.sdr");

const char *vertexShaderText = vss.c_str();
const char *fragmentShaderText = fss.c_str();
```

Shaders

- Compilando e habilitando os shaders.
- Inserir o código a seguir, após a linha

  const char *fragmentShaderText = fss.c_str();

```
GLuint vertexShader = glCreateShader(GL_VERTEX_SHADER);
glShaderSource(vertexShader, 1, &vertexShaderText, nullptr);
glCompileShader(vertexShader);

GLuint fragmentShader = glCreateShader(GL_FRAGMENT_SHADER);
glShaderSource(fragmentShader, 1, &fragmentShaderText, nullptr);
glCompileShader(fragmentShader);

GLuint shaderProgram = glCreateProgram();
glAttachShader(shaderProgram, vertexShader);
glAttachShader(shaderProgram, fragmentShader);

glLinkProgram(shaderProgram);
glUseProgram(shaderProgram);
```

Compile e execute o projeto

Vai dar certo! ☺

Fim da Aula 06

André Luiz Brandão