计算机图形学原理

--------教程4

李瑞辉

大纲

作业 1 中的问题

- 如何表示一个对象并将数据发送到 GPU？
- 如果使用索引绘制对象，如何编程？
- 如何表示多个对象？
- 如何使用具有对象属性的顶点着色器和片段着色器？

问题1

问题：如何表示一个对象并将数据发送到 GPU？

解决方案：使用 VAO 和 VBO！参见教程 02。

问题1

问题：如何表示一个对象并将数据发送到 GPU？

进一步研究（存储对象数据的不同方式）

$$(one\ object\ with\ one\ array)$$

$$\begin{bmatrix} Position\ Data \\ Color\ Data \\ Normal\ Data \\ \ldots\ldots \end{bmatrix} \Rightarrow one\ Vertex\ Array\ Object \Rightarrow one\ Vertex\ Buffer\ Object\ \#1$$

$$(one\ object\ with\ multiple\ arrays)$$

$$\begin{matrix} [Position\ Data] \\ [Color\ Data] \\ [Normal\ Data] \\ [\ldots\ldots] \end{matrix} \Rightarrow \text{one VAO} \Rightarrow \begin{cases} multiple\ VBOs \begin{cases} Buffer\ Data\ of\ Position \\ Buffer\ Data\ of\ Color \\ Buffer\ Data\ of\ Normal \\ \ldots\ldots \end{cases} \#2 \\ one\ VBO \begin{cases} BufferSubData\ of\ Position \\ BufferSubData\ of\ Color \\ BufferSubData\ of\ Normal \\ \ldots\ldots \end{cases} \#3 \end{cases}$$

问题1

问题：如何表示一个对象并将数据发送到 GPU？

$$(one\ object\ with\ one\ array)$$

$$\begin{bmatrix} Position\ Data \\ Color\ Data \\ Normal\ Data \\ \ldots\ldots \end{bmatrix} \Rightarrow one\ Vertex\ Array\ Object \Rightarrow one\ Vertex\ Buffer\ Object\ \#1$$

x, y, z

r, g, b

```
const GLfloat triangle[] =
{
    -0.5f, -0.5f, +0.0f,  // left
    +1.0f, +0.0f, +0.0f,  // color

    +0.5f, -0.5f, +0.0f,  // right
    +1.0f, +0.0f, +0.0f,

    +0.0f, +0.5f, +0.0f,  // top
    +1.0f, +0.0f, +0.0f,
};
```

定义对象数据

问题1

问题：如何表示一个对象并将数据发送到 GPU？

$one\ model\ array \Rightarrow one\ Vertex\ Array\ Object \Rightarrow one\ Vertex\ Buffer\ Object\ \#1$

```
GLuint vaoID;
glGenVertexArrays(1, &vaoID);
glBindVertexArray(vaoID);  //first VAO

GLuint vboID;
glGenBuffers(1, &vboID);
glBindBuffer(GL_ARRAY_BUFFER, vboID);
glBufferData(GL_ARRAY_BUFFER, sizeof(triangle), triangle, GL_STATIC_DRAW);

// 1st attribute: vertex position
glEnableVertexAttribArray(0);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), 0);

// 2nd attribute: vertex color
glEnableVertexAttribArray(1);
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float),
    (char*)(3 * sizeof(float)));
```

问题1

问题：如何表示一个对象并将数据发送到 GPU？

$$
\begin{aligned}
&(\textit{one model with multiple arrays}) \\
&[\textit{Position Data}] \\
&[\textit{Color Data}] \\
&[\textit{Normal Data}] \\
&[\ldots\ldots]
\end{aligned}
\Rightarrow \text{one VAO} \Rightarrow
\begin{cases}
\textit{multiple VBOs} \begin{cases} \textit{Buffer Data of Position} \\ \textit{Buffer Data of Color} \\ \textit{Buffer Data of Normal} \\ \ldots\ldots \end{cases} \#2 \\
\textit{one VBO} \begin{cases} \textit{BufferSubData of Position} \\ \textit{BufferSubData of Color} \\ \textit{BufferSubData of Normal} \\ \ldots\ldots \end{cases} \#3
\end{cases}
$$

```
/*We're going to create a red triangle*/
const GLfloat triangle_verts[] =
{
    +0.0f, +1.0f, +0.0f, // top point

    -1.0f, -1.0f, +0.0f, // left point

    +1.0f, -1.0f, +0.0f, // right point
};
const GLfloat triangle_color[] =
{
    +1.0f, +0.0f, +0.0f,

    +1.0f, +0.0f, +0.0f,

    +1.0f, +0.0f, +0.0f,
};
```

问题1

```
/*This is a handle to Vertex Array Object*/
GLuint vao;
/*Allocate and assign a Vertex Array Object to our handle*/
glGenVertexArrays(1, &vao);
/*Bind our Vertex Array Object as the current used object*/
glBindVertexArray(vao);

/*This is a handle to Vertex Buffer Object*/
GLuint vbo[2];
/*Allocate and assign two Vertex Buffer Objects to our handle*/
glGenBuffers(2, vbo);
/*Bind first VBO as being the active buffer and storing vertex attributes*/
glBindBuffer(GL_ARRAY_BUFFER, vbo[0]);
/*Copy the data from triangle_verts array to our buffer*/
glBufferData(GL_ARRAY_BUFFER, sizeof(triangle_verts), triangle_verts, GL_STATIC_DRAW);
/*Enable attribute index 0 as being used*/
glEnableVertexAttribArray(0);
/*Specify that our coordinate data is going into attribute index 0, and contains 3 floats per vertex*/
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);

/*Bind second VBO as being the active buffer and storing color attributes*/
glBindBuffer(GL_ARRAY_BUFFER, vbo[1]);
/*Copy the data from triangle_color array to our buffer*/
glBufferData(GL_ARRAY_BUFFER, sizeof(triangle_color), triangle_color, GL_STATIC_DRAW);
/*Enable attribute index 1 as being used*/
glEnableVertexAttribArray(1);
/*Specify that our color data is going into attribute index 1, and contains 3 floats per vertex*/
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 0, 0);
```

$\Rightarrow$ one VAO

$\Rightarrow$ *multiple VBOs*

$\Rightarrow$ *Buffer Data of Position*

$\Rightarrow$ *Buffer Data of Color*

问题1

问题：如何表示一个对象并将数据发送到 GPU？

$$
\begin{array}{c}
(one\ model\ with\ multiple\ arrays) \\
[Position\ Data] \\
[Color\ Data] \\
[Normal\ Data] \\
[\ldots\ldots]
\end{array}
\Rightarrow \text{one VAO} \Rightarrow
\begin{cases}
multiple\ VBOs \begin{cases} Buffer\ Data\ of\ Position \\ Buffer\ Data\ of\ Color \\ Buffer\ Data\ of\ Normal \\ \ldots\ldots \end{cases} \#2 \\
one\ VBO \begin{cases} BufferSubData\ of\ Position \\ BufferSubData\ of\ Color \\ BufferSubData\ of\ Normal \\ \ldots\ldots \end{cases} \#3
\end{cases}
$$

```
/*We're going to create a red triangle*/
const GLfloat triangle_verts[] =
{
    +0.0f, +1.0f, +0.0f, // top point

    -1.0f, -1.0f, +0.0f, // left point

    +1.0f, -1.0f, +0.0f, // right point
};
const GLfloat triangle_color[] =
{
    +1.0f, +0.0f, +0.0f,

    +1.0f, +0.0f, +0.0f,

    +1.0f, +0.0f, +0.0f,
};
```

问题1

```
/*This is a handle to Vertex Array Object*/
GLuint vao;
/*Allocate and assign a Vertex Array Object to our handle*/
glGenVertexArrays(1, &vao);  ⇒ one VAO
/*Bind our Vertex Array Object as the current used object*/
glBindVertexArray(vao);

/*This is a handle to Vertex Buffer Object*/
GLuint vbo;
/*Allocate and assign a Vertex Buffer Objects to our handle*/
glGenBuffers(1, &vbo);       ⇒ one VBO
/*Bind first VBO as being the active buffer and storing vertex attributes*/
glBindBuffer(GL_ARRAY_BUFFER, vbo);
/*Copy all the triangle data to our buffer*/
glBufferData(GL_ARRAY_BUFFER, sizeof(triangle_verts)+sizeof(triangle_color),
             NULL, GL_STATIC_DRAW);
glBufferSubData(GL_ARRAY_BUFFER, 0, sizeof(triangle_verts), triangle_verts);⇒ SubData of Position
glBufferSubData(GL_ARRAY_BUFFER, sizeof(triangle_verts), sizeof(triangle_color), triangle_color);
                                                ⇒ SubData of Color
/*Enable attribute index 0 as being used*/
glEnableVertexAttribArray(0);
/*Specify that our coordinate data is going into attribute index 0, and contains 3 floats per vertex*/
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
/*Enable attribute index 1 as being used*/
glEnableVertexAttribArray(1);
/*Specify that our color data is going into attribute index 1, and contains 3 floats per vertex*/
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 0, (char*)(sizeof(triangle_verts)));
```

问题2

问题：如果对象是用索引绘制的，如何编程？

解决方案：使用索引数组和*glDrawElements* 。请参见教程 03。

问题3

问题：如何表示多个对象？

解决方案：使用 VAO 数组（全局变量）。请参见教程 03。

回想一下：OpenGL 是一个状态机：一个一个地进行对象操作。记住*glBindVertexArray* 。

问题4

问题：如何使用带
对象属性的顶点着色器和片段着色器？

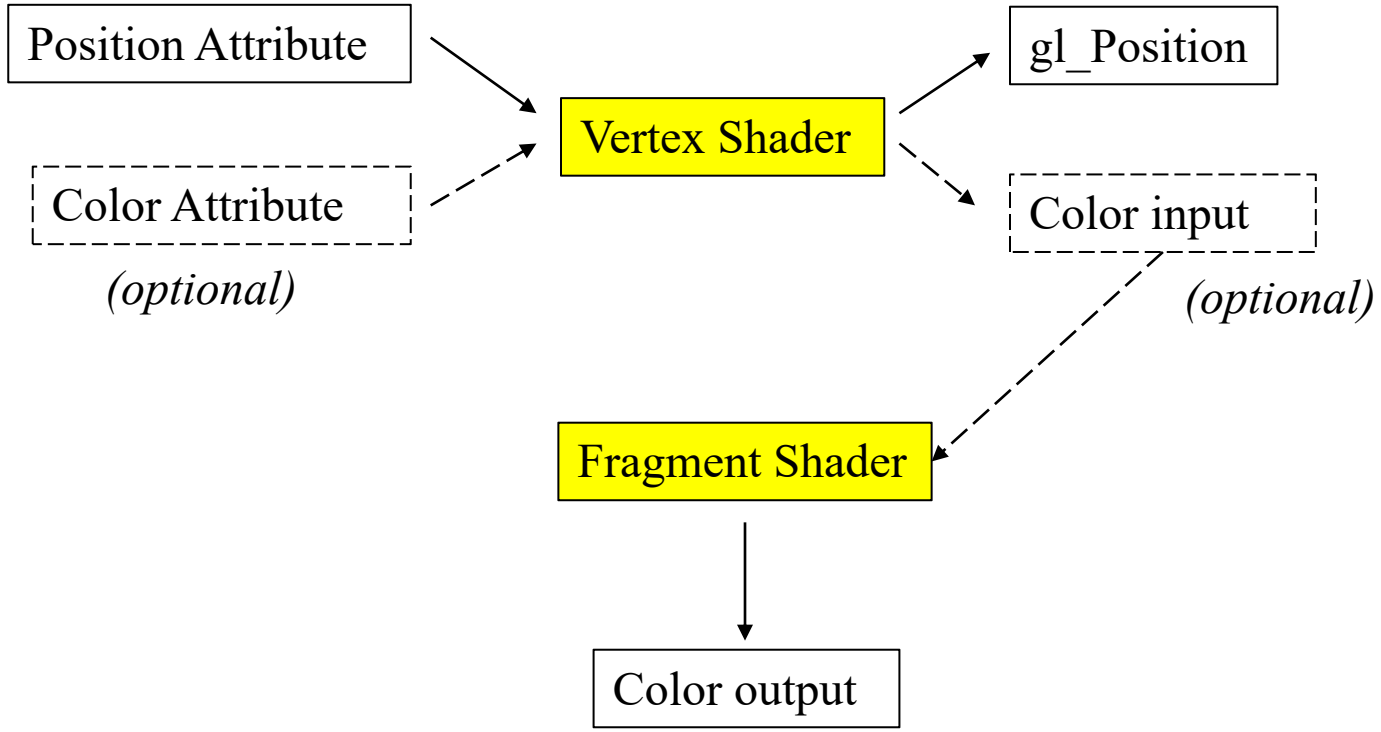
```
GLuint vaoID;
glGenVertexArrays(1, &vaoID);
glBindVertexArray(vaoID);  //first VAO

GLuint vboID;
glGenBuffers(1, &vboID);
glBindBuffer(GL_ARRAY_BUFFER, vboID);
glBufferData(GL_ARRAY_BUFFER, sizeof(triangle), triangle, GL_ST

// 1st attribute: vertex position
glEnableVertexAttribArray(0);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(floa

// 2nd attribute: vertex color
glEnableVertexAttribArray(1);
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(floa
    (char*)(3 * sizeof(float)));
```

VertexShaderCode.glsl | FragmentShaderCode.glsl | main.cpp

```
#version 430

in layout(location=0) vec3 position;
in layout(location=1) vec3 vertexColor;

uniform mat4 modelTransformMatrix;

out vec3 theColor;

void main()
{
    vec4 v = vec4(position, 1.0);
    vec4 out_position = modelTransformMatrix * v;
    gl_Position = out_position;
    theColor = vertexColor;
}
```

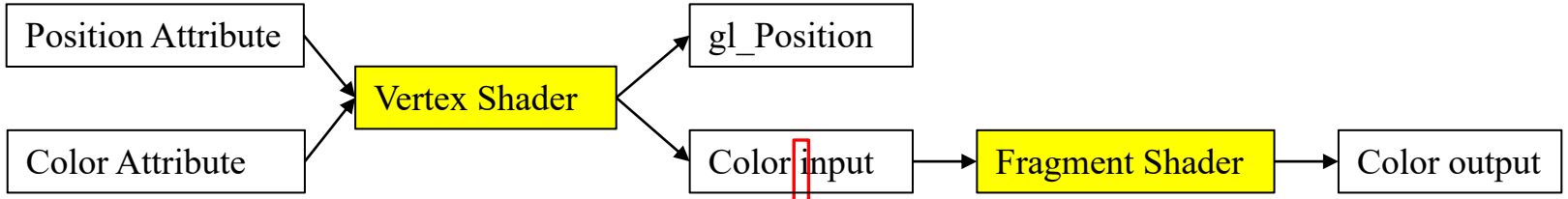
问题4

问题：如何使用带
对象属性的顶点着色器和片段着色器？



问题4

例子



VertexShaderCode.glsl | FragmentShaderCode.glsl | main.cpp

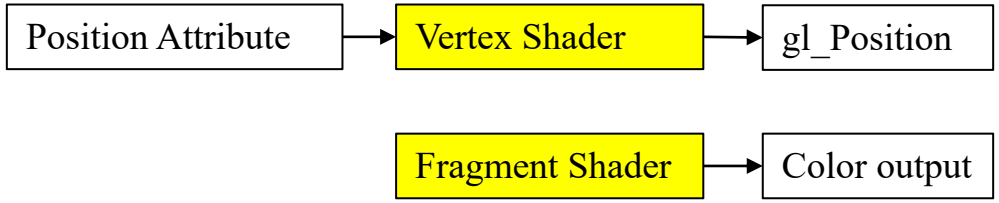
```
#version 430

in layout(location=0) vec3 position;
in layout(location=1) vec3 vertexColor;

uniform mat4 modelTransformMatrix;

out vec3 theColor;

void main()
{
    vec4 v = vec4(position, 1.0);
    vec4 out_position = modelTransformMatrix * v;
    gl_Position = out_position;
    theColor = vertexColor;
}
```

VertexShaderCode.glsl | FragmentShaderCode.glsl | main.cpp

```
#version 430

out vec4 Color;
in vec3 theColor;

void main()
{
    Color = vec4(theColor, 1.0);
}
```

问题4

例子



VertexShaderCode.glsl*

```
#version 430

in layout(location=0) vec3 position;

void main()
{
    vec4 v = vec4(position, 1.0);
    gl_Position = v;
}
```

FragmentShaderCode.glsl*

```
#version 430

out vec4 daColor;

void main()
{
    daColor = vec4(0.0,1.0,0.0,1.0);
}
```

sendDataToOpenGL () {

对象数组 01；

VAO
glGenVertexArrays ();
glBindVertexArray ();

VBO
glGenBuffers ();
glBindBuffer ();
glBufferData ();

glEnableVertexAttribArray ();
glVertexAttribPointer ();

对象数组 02；
VAO；
伊博;
glEnableVertexAttribArray ();
glVertexAttribPointer ();

对象数组 03；
……..
……..
}

油漆GL （）{
glGetUniformLocation （“模型矩阵”）；
// 同样，对于视图和投影矩阵

glBindVertexArray (01);
模型矩阵= 平移 * 旋转 * 缩放 * …;
viewMatrix = glm:lookAt ();
projectionMatrix = glm:perspective ();
glUniformMatrix4fv(&模型矩阵);
// 同样，用于视图和投影
glDrawArrays (); / glDrawElements ();

glBindVertexArray (02);
模型矩阵= 平移 * 旋转 * 缩放 * …;
// 可选的视图和投影
glUniformMatrix4fv(& 模型矩阵）；
// 可选的视图和投影
glDrawArrays (); / glDrawElements ();

glBindVertexArray (03);
……..
……..
}

```
[VertexShader]

#version ***
…….
…….

Uniform modelMatrix;
Uniform viewMatrix;
Uniform projectionMatrix;

void main()
{
    gl_Position = …..;
}
```

```
[FragmentShader]

#version ***
…….
…….

void main()
{
    Color = …..;
}
```