计算机图形学小白入门

————从0开始实现OpenGL

Draw函数流程与Shader

授课：赵新政
资深三维工程师

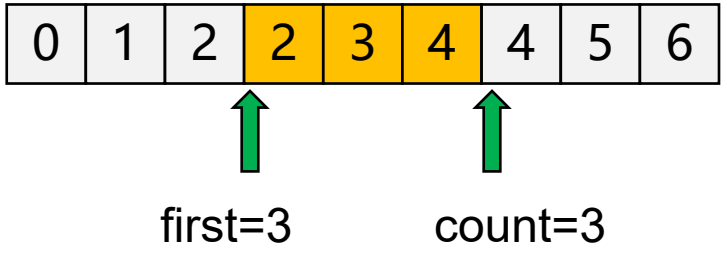
专注3D图形学技术
教育品牌

渲染指令发出函数

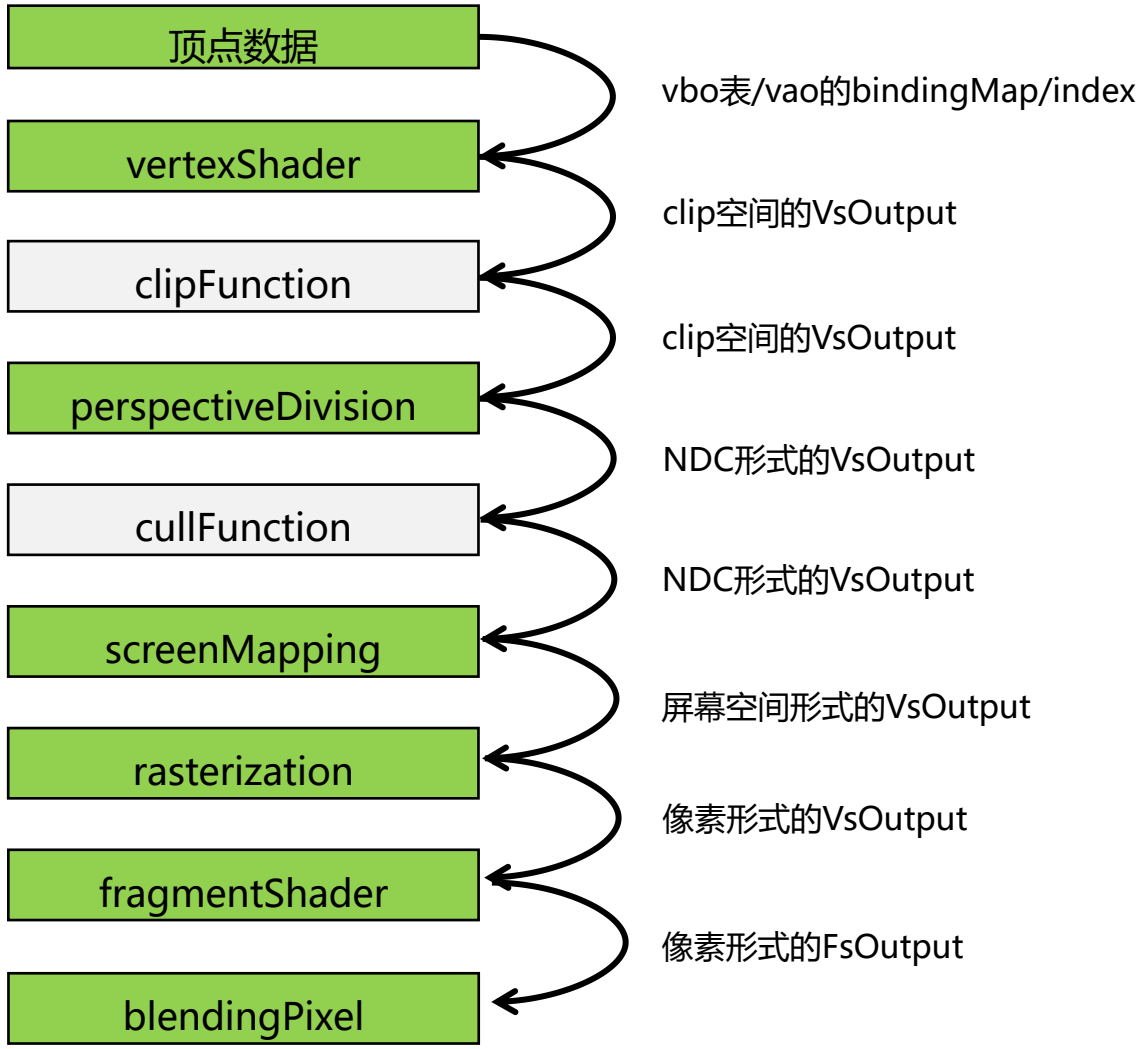
- 我们设计绘制指令函数如下形式：

```
void drawElement(const uint32_t& drawMode, const uint32_t& first,  const uint32_t& count);
```

- 调用Draw函数之前，必须绑定好需要绘制的VAO/EBO
- **drawMode**：绘制直线/三角形
- **first**：从ebo的第几个index开始绘制
- **count**：绘制ebo当中多少个index的顶点



渲染函数流程规划


顶点数据
vertexShader
clipFunction
perspectiveDivision
cullFunction
screenMapping
rasterization
fragmentShader
blendingPixel
vbo表/vao的bindingMap/index
clip空间的VsOutput
clip空间的VsOutput
NDC形式的VsOutput
NDC形式的VsOutput
屏幕空间形式的VsOutput
像素形式的VsOutput
像素形式的FsOutput

Shader类规划

- 我们设计如下Shader类的虚基类

Class Shader

virtual VsOutput vertexShader(
//VAO当中的bindingMap
const std::map<uint32_t, BindingDescription>& bindingMap,
//VBO当中的bufferMap
const std::map<uint32_t,BufferObject*>& bufferMap,
//当前要处理的顶点的index
const uint32_t& index
) = 0;

virtual void fragmentShader(const VsOutput& input, FsOutput& output) = 0;

Uniform变量

- 分析对于一个Mesh的一次Draw调用
- 对于Shader而言，每次VertexShader调用都会传入**不同**顶点的属性；每次FragmentShader都会传入**不同**片元数据；
- 但是每个顶点都会使用**相同**的model/view/projection矩阵，每个Fragment都会使用相同的光照数据

- **Uniform变量**：把每次vs公用数据以及每次fs公用数据进行抽取，从而构成统一且公用的数据
- 不同种类的Shader子类，其Uniform变量也不尽相同；
- 举例：Default Shader

Class DefaultShader
//uniforms math::mat4f mModelMatrix; math::mat4f mViewMatrix; math::mat4f mProjectionMatrix;

Draw函数流程与Shader

DefaultShader->Shader

Class DefaultShader
VsOutput vertexShader(.....) ;
void fragmentShader(.....) = 0;
//tool functions
//uniforms math::mat4f mModelMatrix; math::mat4f mViewMatrix; math::mat4f mProjectionMatrix;