Logic_视觉班课堂笔记[CC015]

- **日期**: 2019年6月14日星期五
- **授课**: CC老师
- **课程次数**: 视觉班第15次课--共计(22次课)
- **主题**: OpenGL ES 主题

课程内容

- GLKit 加载金字塔
- GLSL 内建函数
- GLKit 光照处理

课程安排

- 08:00 - 09:00 第一节课
- 09:00 - 09:10 课间休息
- 09:10 - 10:00 第二节课
- 10:00 - 10:10 课程总结
- 10:10 - 10:30 课后答疑

课后作业:

1. 请各位同学课后完成课后复习
2. **要求**:
    i. 能独立完成
    ii. 能将案例总结成思维导图

一.课程回顾

1. 案例回顾
2. 作业参考答案讲解

二.课程笔记

顶点着色器

逻辑教育
Logic education

顶点着色器

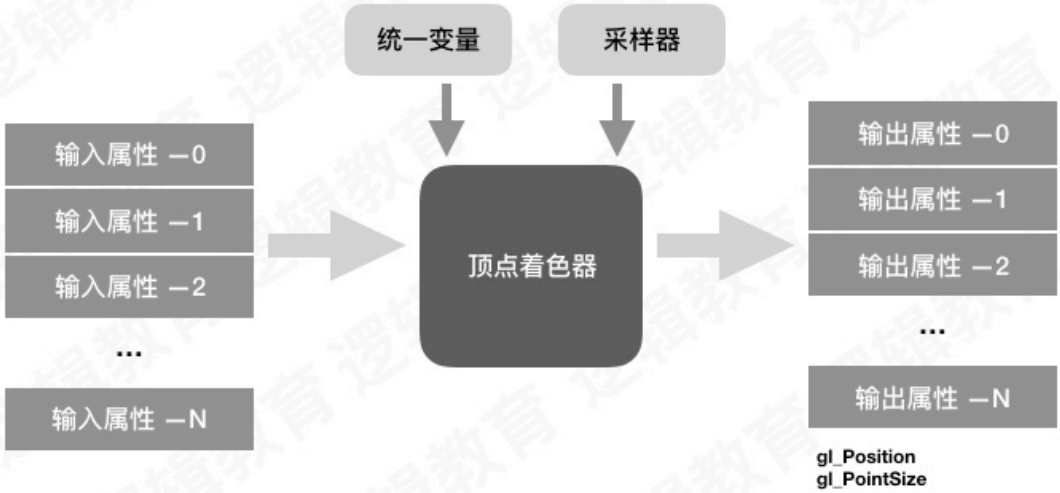
顶点着色器 输入:

1. 着色器程序—描述顶点上执行操作的顶点着色器程序源代码/可执行文件
2. 顶点着色器输入(属性) — 用顶点数组提供每个顶点的数据
3. 统一变量(uniform)—顶点/片元着色器使用的不变数据
4. 采样器—代表顶点着色器使用纹理的特殊统一变量类型.

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

OpenGL ES 3.0 顶点着色器



课程研发:CC老师
课程授课:CC老师

顶点着色器的内建变量可以分为特殊变量(顶点着色器输入/输出),统一状态(深度范围)以及规定的最大值(属性数量,顶点着色器变量数量和统一变量数量)的常量.

内建特殊变量

- `gl_VertexID`,是一个输入变量,用于保存顶点的整数索引. 这个整数型变量用highp精度限定符声明
- `gl_InstanceID`,是一个输入变量.用于保存实例化绘图调用中图元的实例编号.对于常规调用的绘图调用,该值为0; `gl_InstanceID` 是一个整数型变量,用 `highp` 精度限定修饰符声明.
- `gl_Position` 用于输出顶点位置的裁剪坐标.该值在裁剪和视口变换用于执行相应的图元裁剪以及从裁剪坐标到屏幕坐标的顶点位置转换. 如果顶点着色器未写入 `gl_Position`.则 `gl_Position` 的值未定义. `gl_Position` 是浮点变量,用 `highp` 精度限定修饰符声明.
- `gl_PointSize` ,可以写入像素表示点精灵的尺寸. 在渲染点精灵时使用.顶点着色器输出的gl_PointSize 值被限定在OpenGL ES 3.0 实现支持的非平滑点大小范围之内. `gl_PointSize` 是一个浮点变量.用于 `highp` 精度限定符声明
- `gl_FrontFacing` : 是一个特殊变量,但不是由顶点着色器直接写入的,而是根据顶点着色器生成的位置值和渲染图元的类型生成的.它是一个布尔值.

内建统一状态

在顶点着色器内可用的唯一内建 Uniform 状态是窗口坐标中的深度范围.这些由内建统一变量名 `gl_DepthRange` 给出.

```
struct gl_DepthRangeParameters
{
    highp float near; //near z
    highp float far;  //near far
    highp float diff; //far - near
}

uniform gl_DepthRangeParameters gl_DepthRange;
```

内建常量

```
const mediump int gl_MaxVertexAttribs = 16;
```

```
const mediump int gl_MaxVertexUniformVectors = 256;
const mediump int gl_MaxVertexOutputVectors = 16;
const mediump int  gl_MaxVertexTextureImageUnits = 16;
const mediump int gl_MaxCombinedTextureImageUnits = 32;
```

`gl_MaxVertexAttribs`：可以指定的顶点数据最大的数量.
`gl_MaxVertexUniformVectors`：顶点着色器可以使用的vect4 Uniform 变量最大数量.
`gl_MaxVertexOutputVectors`：是输出向量的最大数量
`gl_MaxVertexTextureImageUnits`：顶点着色器可用纹理单元的最大数量.
`gl_MaxCombinedTextureImageUnits`：顶点/片段着色器中的可用纹理单元的最大数量的总和.

顶点着色器中的矩阵变换

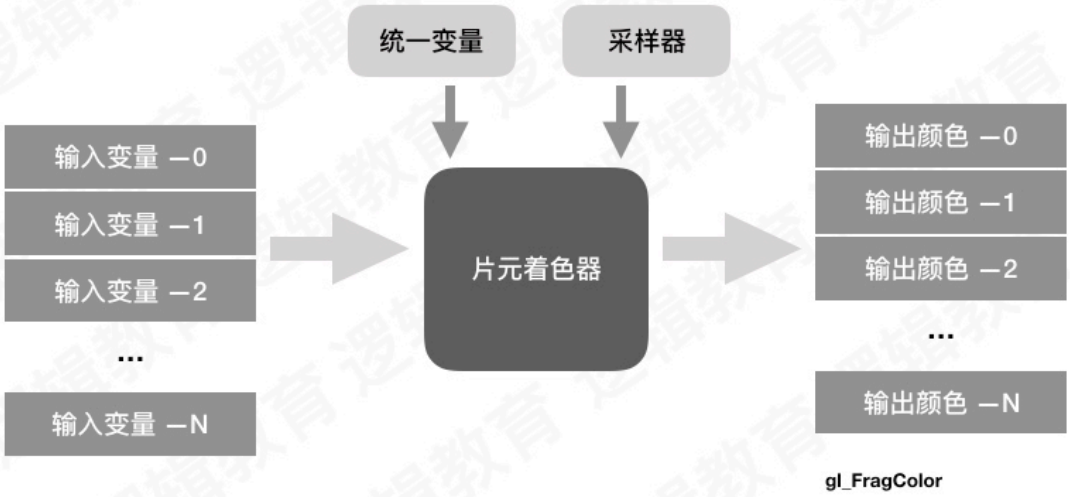
MVP矩阵(模型-视图-投影矩阵.) 顶点着色器的位置输入保存的是物体坐标,而输出的坐标保存为裁剪坐标. MVP矩阵是3D图形中进行这种变换的3个非常重要的变换矩阵的乘积: 模型矩形,视图矩阵和投影矩阵.

组成MVP矩阵的每个单独矩阵执行的变换如下:

- 模型矩阵 -- 将物体坐标变换为世界坐标
- 视图矩阵 -- 将世界坐标变换为眼睛坐标(观察坐标)
- 投影矩阵 -- 将眼睛坐标(观察者坐标)变换为裁剪坐标.

片元着色器

OpenGL ES 3.0 片段着色器/片元着色器



片元着色器业务:

片元着色器 业务:

1. 计算颜色
2. 获取纹理值
3. 往像素点中填充颜色值(纹理值/颜色值);

总结: 它可以用于图片/视频/图形中每个像素的颜色填充(比如给视频添加滤镜,实际上就是将视频中每个图片的像素点颜色填充进行修改.)

片元着色代码案例:

```
varying lowp vec2 varyTextCoord;
uniform sampler2D colorMap;
void main()
{
    gl_FragColor = texture2D(colorMap, varyTextCoord);
}
```

内建特殊变量

- `gl_FragCoord`：片段着色器中一个只读变量,这个变量保存片段的窗口相对坐标.

- `gl_FrontFacing`：片段着色器中的一个只读变量,这个布尔变量时正面图元是为true,否则为false;
- `gl_PointCoord`：只读变量,可以在渲染点精灵使用.保存了点精灵的纹理坐标,这个坐标在点精灵光栅化期间自动生成,处于(0,1)区间.
- `gl_FragDepth`：一个只写输出变量,在片段着色器写入时,覆盖片段的固定功能深度值.尽量减少手动实现深度值写入.这个功能需要谨慎使用,因为它可能禁用许多GPU的深度优化.例如,许多GPU都有"`Early`-Z"的功能,在执行片段着色器之前进行深度测试,使用"Early-z"的好处就是不能通过深度测试的片段就不会被着色(从而减低了着色器调用次数,提高了性能),但是使用gl_FragDepth,就必须禁用该功能.因为GPU在执行着色器之前不知道深度值.

内建常量

```
const mediump int gl_MaxFragmentInputVectors = 15;
const mediump int gl_MaxTextureImageUnits = 16;
const mediump int gl_MaxFragmentUniformVectors = 224;
const mediump int gl_MaxDrawBuffers = 4;
const mediump int gl_MinProgramTexelOffset = -8;
const mediump int gl_MaxProgramTexelOffset = 7
```

- `gl_MaxFragmentInputVectors`：片段着色器输入的最大数量.
- `gl_MaxFragmentUniformVectors`：可用纹理图像单元的最大数量
- `gl_MaxFragmentUniformVectors`：片段着色器可用Vec4 Uniform变量最大数量
- `gl_MaxDrawBuffers`：多重渲染目标最大支持数量

多重纹理渲染

最终颜色,由多张纹理合成计算.

```
//片元着色器代码
attribute vec2 v_texCoord;
uniform sampler2D s_baseMap;
uniform sampler2D s_SecondMap;
void main()
{
    vec4 baseColor;
    vec4 secondColor;

    baseColor = texture(s_baseMap,v_texCoord);
```

```
    secondColor = texture(s_SecondMap,v_texCoord);

    gl_FragColor = baseColor * secondColor;

}
```

```
//客户端代码：将各个纹理对象绑定到纹理单元0和1,为采样器设置数值,将采集器绑定到对应的纹理单元
glActiveTexutre(GL_TEXTURE0);
glBindTeture(GL_TEXTURE_2D,baseMapTexId);
glUniformli(baseMapTexId,0);

glActiveTexutre(GL_TEXTURE1);
glBindTeture(GL_TEXTURE_2D,secondMapTexId);
glUniformli(secondMapTexId,1);
```

三.课程总结

> 详细请参考课后总结视频,这里只是做为关键字记录.

四.课程答疑

> 详细请参考课后答疑视频,这里只是做为关键字记录