Logic_视觉班课堂笔记[CC013]

- **日期**: 2019年6月10日星期四
- **授课**: CC老师
- **课程次数**: 视觉班第13次课--共计(22次课)
- **主题**: OpenGL ES 主题

课程内容

- 案例(1)--OpenGL ES GLSL 加载图片
- 探索纹理翻转的解决策略

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. **请在个人博客上更新一篇博文,将今晚所学总结!**
    - OpenGL ES 关于纹理翻转策略解析

- **要求**:
    i. 将课程内容加上自己的理解
    ii. 更新的博客地址通过QQ私发给我.互相学习

一.课程回顾

①.顶点着色器/片元着色器回顾

②.案例回顾

二.课程笔记

①.关于申请缓存区标记

```
    //2.申请一个缓存区标志
    //glGenRenderbuffers(1, &buffer);
    //glGenFramebuffers(1, &buffer);
    glGenBuffers(1, &buffer);
```

②.关于GLSL 编译

在OpenGL ES中，每个 `program` 对象有且仅有一个 `Vertex Shader` 对象和一个 `Fragment Shader` 对象连接到它。

- **Shader**：类似于C编译器
- **Program**：类似于C链接器
- **glLinkProgram**操作产生最后的可执行程序，它包含最后可以在硬件上执行的硬件指令。

Shader和Program编程概述

1️⃣.创建Shader

- 编写 `Vertex Shader` 和 `Fragment Shader` 源码。
- 创建两个 `shader` 实例： `glCreateShader;`
- 给 `Shader` 实例指定源码。 `glShaderSource`
- 编译 `shader` 源码 `glCompileShader`

2️⃣.创建Program

- 创建**program** `glCreateProgram`
- 绑定**shader**到**program** 。 `glAttachShader(GLuint program, GLuint shader)` 。每个**program**必须绑定一个**Vertex Shader** 和一个**Fragment Shader**。
- 链接**program** `glLinkProgram(GLuint program)`
- 使用**porgram** 。 `glUseProgram(GLuint program)`

③.关于GLSL 精度限定符解析

Qualifier	Meaning
highp	Satisfies the minimum requirements for the vertex language described above. Optional in the fragment language.
mediump	Satisfies the minimum requirements above for the fragment language. Its range and precision has to be greater than or the same as provided by **lowp** and less than or the same as provided by **highp.**
lowp	Range and precision that can be less than **mediump,** but still intended to represent all color values for any color channel.

> Satisfies the minimum requirements for the vertex language described above.Optional in the fragment language
>
> 满足上面描述的顶点语言的最低要求。在片段语言中是可选的
>
> satisfies the minimum requirements above for the fragment language . Its range and precision has to be greater than or the same as provided by lowp and less than of the same as provided by highp.
>
> 满足上述片段语言的最低要求。其范围和精度必须大于或等于lowp提供的范围和精度，小于highp提供的范围和精度。
>
> Range and precision that can be less than mediump , but still intended to represent all color values for any color channel.
>
> 范围和精度可以小于mediump，但仍用于表示任何颜色通道的所有颜色值。

例如:

```
lowp float color;

varying mediump vec2 Coord;

highp mat4 m;
```

精度范围

Floating Point Range 浮点数范围

- **highp** (-2^{62}, 2^{62});
- **mediump** (-2^{14}, 2^{14});
- **lowp** (-2,2);

Integer Range 整数范围

- **highp** (-2^{16}, 2^{16});
- **mediump** (-2^{10}, 2^{10});
- **lowp** (-2^{8}, 2^{8});

对于高精度和中级精度，整型范围必须可以准确地转化成相应的相同精度修饰符所表示的float型。

例如, `highp int` 可以被转换成 `highp float`， `mediump int` 可以被转换成 `mediump float`，但是 `lowp int` 不能转换成相应的 `lowp float` 。

字符常量和布尔型没有精度修饰符.当浮点数和整数构造器不含带有精度修饰符的参数时也不需要精度修饰符。

1️⃣.指定变量精度(放在数据类型之前)

```
highp vec4 position;
varying lowp vec4 color;
mediump float specularExp;
```

2️⃣.指定默认精度(放在Vertex 和 Fragment shader 源码开始处)

```
precision precision-qualifier type;

*  precision可以用来确定默认精度修饰符
*  precision-qualifier可以是lowp，mediump，或者highp.任何其他类型和修饰符都会引起错误
```

如果 `type` 是 `float` 类型，那么该精度（ `precision-qualifier`）将适用于所有无精度修饰符的浮点数声明（标量，向量，矩阵）；

如果 `type` 是 `int` 类型，那么该精度（ `precision-qualifier` ）将适用于所有无精度修饰符的整型数声明（标量，向量）；

包括全局变量声明，函数返回值声明，函数参数声明，和本地变量声明等。没有声明精度修饰符的变量将使用和它最近的precision语句中的精度。

```
precision highp float;
precision mediump int;
```

在 `Vertex Shader` 中，如果没有默认的精度，则 `float` 和 `int` 精度都为 `highp`；

在 `Fragment Shader` 中， `float` 没有默认的精度，所以必须在 `Fragment Shader` 中为 `float` 指定一个默认精度或为每个 `float` 变量指定精度。

3️⃣.预定义精度

在顶点语言中有如下预定义的全局默认精度语句

```
precision highp float;

precision highp int;

precision lowp sampler2D;

precision lowp samplerCube;
```

在片元语言中有如下预定义的全局默认精度语句：

```
precision mediump int;

precision lowp sampler2D;

precision lowp samplerCube;
```

片元语言没有默认的浮点数精度修饰符。因此，对于浮点数，浮点数向量和矩阵变量声明，要么声明必须包含一个精度修饰符，要不默认的精度修饰符在之前已经被声明过了。

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

纹理翻转策略2. 用以下也可以 .特别棒!

```
//提拉米苏同学：以下3行也可以！特别棒!!!!
    CGContextTranslateCTM(spriteContext, 0, rect.size.height);
    CGContextScaleCTM(spriteContext, 1.0, -1.0);
    CGContextDrawImage(spriteContext, rect, spriteImage);
```