Logic_视觉班课堂笔记[CC021]

- **日期**: 2019年6月28日星期五
- **授课**: CC老师
- **课程次数**: 视觉班第21次课--共计(22次课)
- **主题**: OpenGL ES 主题

课程内容

- OpenGL ES 自定义滤镜(3)

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

课程回顾

- 六边形马赛克
- 三角形马赛克滤镜

课程笔记

缩放滤镜

- 顶点着色器

```
//顶点坐标/纹理坐标映射关系.放大?
//放大过程. 顶点着色器完成.

//顶点坐标
attribute vec4 Position;
```

```
//纹理坐标
attribute vec2 TextureCoords;
//纹理坐标
varying vec2 TextureCoordsVarying;
//时间撮(及时更新)
uniform float Time;
//PI
const float PI = 3.1415926;

void main (void) {

    //一次缩放效果时长 0.6
    float duration = 0.6;
    //最大缩放幅度
    float maxAmplitude = 0.3;

    //表示时间周期.范围[0.0~0.6];
    float time = mod(Time, duration);

    //amplitude [1.0,1.3]
    float amplitude = 1.0 + maxAmplitude * abs(sin(time * (PI / dura
tion)));

    // 顶点坐标x/y 分别乘以放大系数[1.0,1.3]
    gl_Position = vec4(Position.x * amplitude, Position.y * amplitud
e, Position.zw);

    // 纹理坐标
    TextureCoordsVarying = TextureCoords;
}
```

灵魂出窍片元着色器

```
//灵魂出窍滤镜：是两个层的叠加，并且上面的那层随着时间的推移，会逐渐放大且不
透明度逐渐降低。这里也用到了放大的效果，我们这次用片段着色器来实现

precision highp float;
//纹理采样器
uniform sampler2D Texture;
//纹理坐标
varying vec2 TextureCoordsVarying;
//时间撮
uniform float Time;
```

```
void main (void) {

    //一次灵魂出窍效果的时长 0.7
    float duration = 0.7;
    //透明度上限
    float maxAlpha = 0.4;
    //放大图片上限
    float maxScale = 1.8;

    //进度值[0,1]
    float progress = mod(Time, duration) / duration; // 0~1
    //透明度[0,0.4]
    float alpha = maxAlpha * (1.0 - progress);
    //缩放比例[1.0,1.8]
    float scale = 1.0 + (maxScale - 1.0) * progress;

    //1.放大纹理坐标
    //根据放大笔记.得到放大纹理坐标 [0,0],[0,1],[1,1],[1,0]
    float weakX = 0.5 + (TextureCoordsVarying.x - 0.5) / scale;
    float weakY = 0.5 + (TextureCoordsVarying.y - 0.5) / scale;
    //放大纹理坐标
    vec2 weakTextureCoords = vec2(weakX, weakY);

    //获取对应放大纹理坐标下的纹素(颜色值rgba)
    vec4 weakMask = texture2D(Texture, weakTextureCoords);

    //原始的纹理坐标下的纹素(颜色值rgba)
    vec4 mask = texture2D(Texture, TextureCoordsVarying);

    //颜色混合 默认颜色混合方程式 = mask * (1.0-alpha) + weakMask * alp
ha;
    gl_FragColor = mask * (1.0 - alpha) + weakMask * alpha;
```

抖动滤镜片元着色器代码

```
//抖动滤镜: 颜色偏移 + 微弱的放大效果
precision highp float;
//纹理
uniform sampler2D Texture;
//纹理坐标
varying vec2 TextureCoordsVarying;
//时间撮
uniform float Time;
```

```
void main (void) {

    //一次抖动滤镜的时长 0.7
    float duration = 0.7;
    //放大图片上限
    float maxScale = 1.1;
    //颜色偏移步长
    float offset = 0.02;

    //进度[0,1]
    float progress = mod(Time, duration) / duration; // 0~1
    //颜色偏移值范围[0,0.02]
    vec2 offsetCoords = vec2(offset, offset) * progress;
    //缩放范围[1.0-1.1];
    float scale = 1.0 + (maxScale - 1.0) * progress;

    //放大纹理坐标.
    vec2 ScaleTextureCoords = vec2(0.5, 0.5) + (TextureCoordsVarying
 - vec2(0.5, 0.5)) / scale;

    //获取3组颜色rgb
    //原始颜色+offsetCoords
    vec4 maskR = texture2D(Texture, ScaleTextureCoords + offsetCoord
s);
    //原始颜色-offsetCoords
    vec4 maskB = texture2D(Texture, ScaleTextureCoords - offsetCoord
s);
    //原始颜色
    vec4 mask = texture2D(Texture, ScaleTextureCoords);

    //从3组来获取颜色:
    //maskR.r,mask.g,maskB.b 注意这3种颜色取值可以打乱或者随意发挥.不一定
写死.只是效果会有不一样.大家可以试试.
    //mask.a 获取原图的透明度
    gl_FragColor = vec4(maskR.r, mask.g, maskB.b, mask.a);


}
```

闪白滤镜片元着色器代码

```
//闪白滤镜: 添加白色图层 ,白色图层的透明度随着时间变化
precision highp float;
```

```
//纹理采样器
uniform sampler2D Texture;
//纹理坐标
varying vec2 TextureCoordsVarying;
//时间撮
uniform float Time;
//PI 常量
const float PI = 3.1415926;

void main (void) {

    //一次闪白滤镜的时长 0.6
    float duration = 0.6;
    //表示时间周期[0.0,0.6]
    float time = mod(Time, duration);
    //白色颜色遮罩层
    vec4 whiteMask = vec4(1.0, 1.0, 1.0, 1.0);
    //振幅: (0.0,1.0)
    float amplitude = abs(sin(time * (PI / duration)));
    //纹理坐标对应的纹素(RGBA)
    vec4 mask = texture2D(Texture, TextureCoordsVarying);

    //利用混合方程式；白色图层 + 原始纹理图片颜色 来进行混合
    gl_FragColor = mask * (1.0 - amplitude) + whiteMask * amplitude;
}
```

毛刺滤镜片元着色器代码

```
//毛刺滤镜：撕裂 + 微弱的颜色偏移
//具体的思路是，我们让每一行像素随机偏移 -1 ~ 1 的距离（这里的 -1 ~ 1 是对于纹理坐标来说的），但是如果整个画面都偏移比较大的值，那我们可能都看不出原来图像的样子。所以我们的逻辑是，设定一个阈值，小于这个阈值才进行偏移，超过这个阈值则乘上一个缩小系数。则最终呈现的效果是：绝大部分的行都会进行微小的偏移，只有少量的行会进行较大偏移
precision highp float;
//纹理
uniform sampler2D Texture;
//纹理坐标
varying vec2 TextureCoordsVarying;
//时间撮
uniform float Time;
//PI
const float PI = 3.1415926;
```

```
//随机数
float rand(float n) {
    //fract(x),返回x的小数部分数据
    return fract(sin(n) * 43758.5453123);
}

void main (void) {

    //最大抖动
    float maxJitter = 0.06;
    //一次毛刺滤镜的时长
    float duration = 0.3;
    //红色颜色偏移量
    float colorROffset = 0.01;
    //绿色颜色偏移量
    float colorBOffset = -0.025;

    //时间周期[0.0,0.6];
    float time = mod(Time, duration * 2.0);
    //振幅:[0,1];
    float amplitude = max(sin(time * (PI / duration)), 0.0);

    //像素随机偏移[-1,1]
    float jitter = rand(TextureCoordsVarying.y) * 2.0 - 1.0; // -1~1

    //是否要做偏移.
    bool needOffset = abs(jitter) < maxJitter * amplitude;

    //获取纹理X值.根据needOffset,来计算它X撕裂.
    //needOffset = YES ,撕裂较大;
    //needOffset = NO,撕裂较小.
    float textureX = TextureCoordsVarying.x + (needOffset ? jitter :
 (jitter * amplitude * 0.006));

    //撕裂后的纹理坐标x,y
    vec2 textureCoords = vec2(textureX, TextureCoordsVarying.y);

    //颜色偏移3组颜色
    //根据撕裂后获取的纹理颜色值
    vec4 mask = texture2D(Texture, textureCoords);
    //撕裂后的纹理颜色偏移
    vec4 maskR = texture2D(Texture, textureCoords + vec2(colorROffse
t * amplitude, 0.0));
    //撕裂后的纹理颜色偏移
    vec4 maskB = texture2D(Texture, textureCoords + vec2(colorBOffse
t * amplitude, 0.0));
```

```
    //红色/蓝色部分发生撕裂.
    gl_FragColor = vec4(maskR.r, mask.g, maskB.b, mask.a);
}
```

课程总结

课程答疑