逻辑教育
Logic education

Hello CC

OpenGL 主题【2】

视觉班—OpenGL 渲染基础

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

课堂目标:

1. 掌握OpenGL渲染架构图
2. 理解OpenGL 数据传递的3种方式
3. 掌握OpenGL 提供的存储着色器
4. 正投影/透视投影 API的使用
5. OpenGL 上常见图元
6. 案例: 在存储着色器的情况下渲染图形并能通过键盘控制.

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

课后作业:

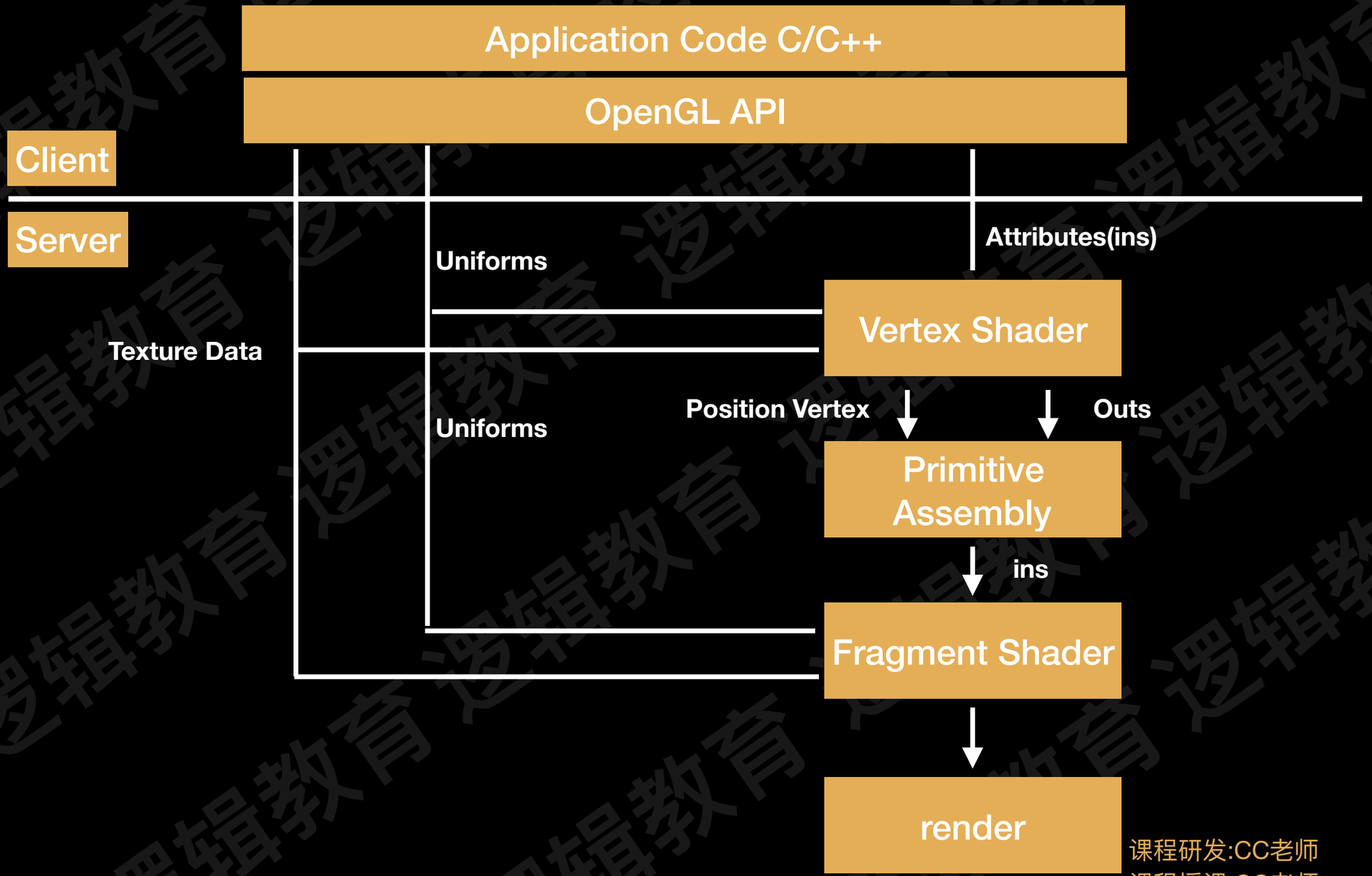
1. 请在个人博客上更新一篇博文,选题如下:
    01. OpenGL 渲染流程图解析
    02. OpenGL 固定存储器着色器理解

要求:

01.将课程内容加上自己的理解
02.更新的博客地址发送到讨论群.互相学习

课程研发:CC老师
课程授课:CC老师

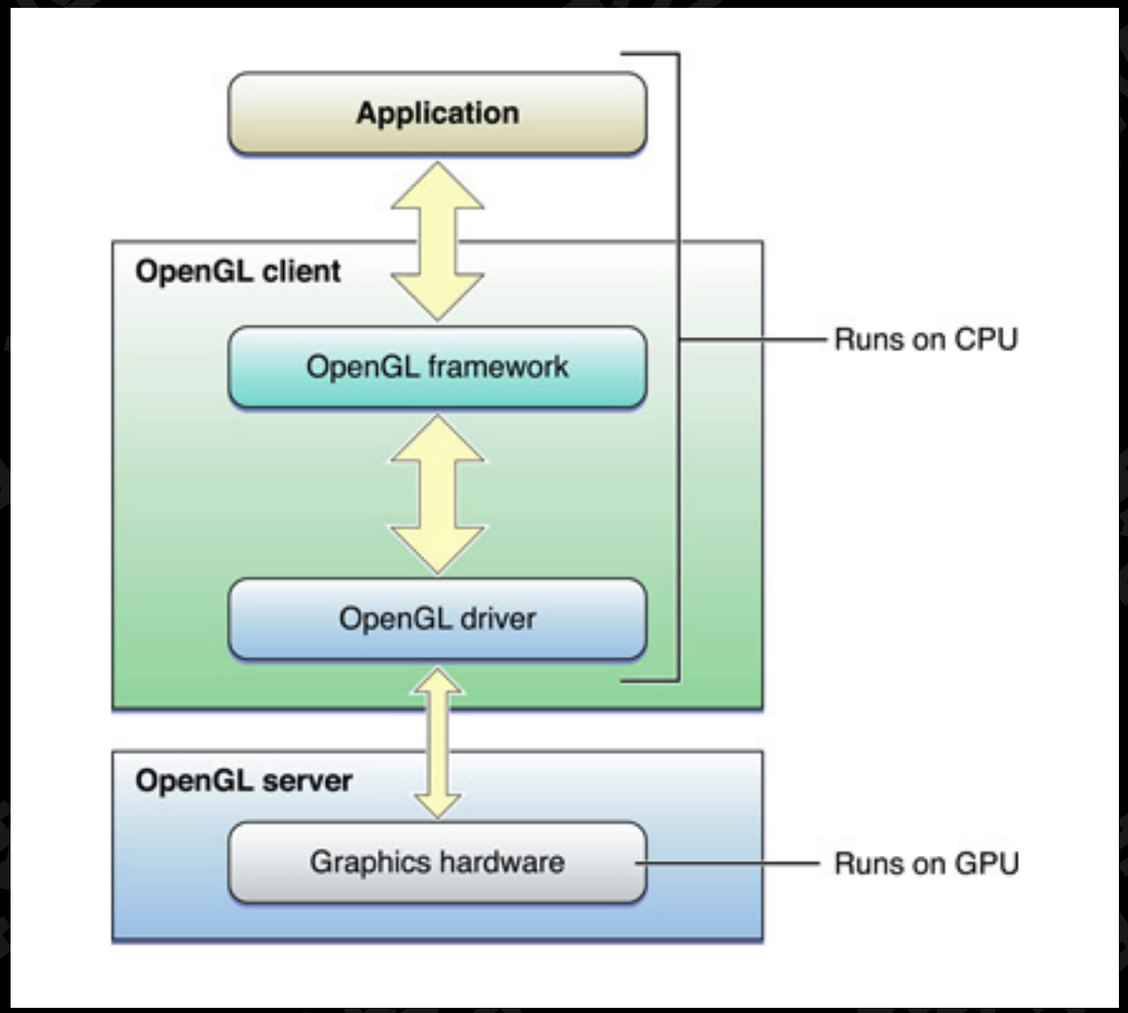
OpenGL 渲染架构



课程研发:CC老师
课程授课:CC老师

由于OpenGL是基于C的API，因此它非常便携且受到广泛支持。作为C API，它与基于Objective-C的Cocoa应用程序无缝集成。OpenGL提供应用程序用于生成2D或3D图像的函数。您的应用程序将渲染的图像呈现给屏幕或将它们复制回自己的内存。

OpenGL规范没有提供自己的窗口层。它依赖于OS X定义的功能来将OpenGL绘图与窗口系统集成。您的应用程序创建OS X OpenGL 渲染上下文并将渲染目标附加到其上（称为可绘制对象）。渲染上下文管理OpenGL状态更改和通过调用OpenGL API创建的对象。



课程研发:CC老师
课程授课:CC老师

课堂练习题:

下面的说明正确的是:[ ]

A. 顶点着色器只能接受Attribute值

B. 片元着色器可以间接获取Attribute值

C. 片元着色器可以接受Attribute值

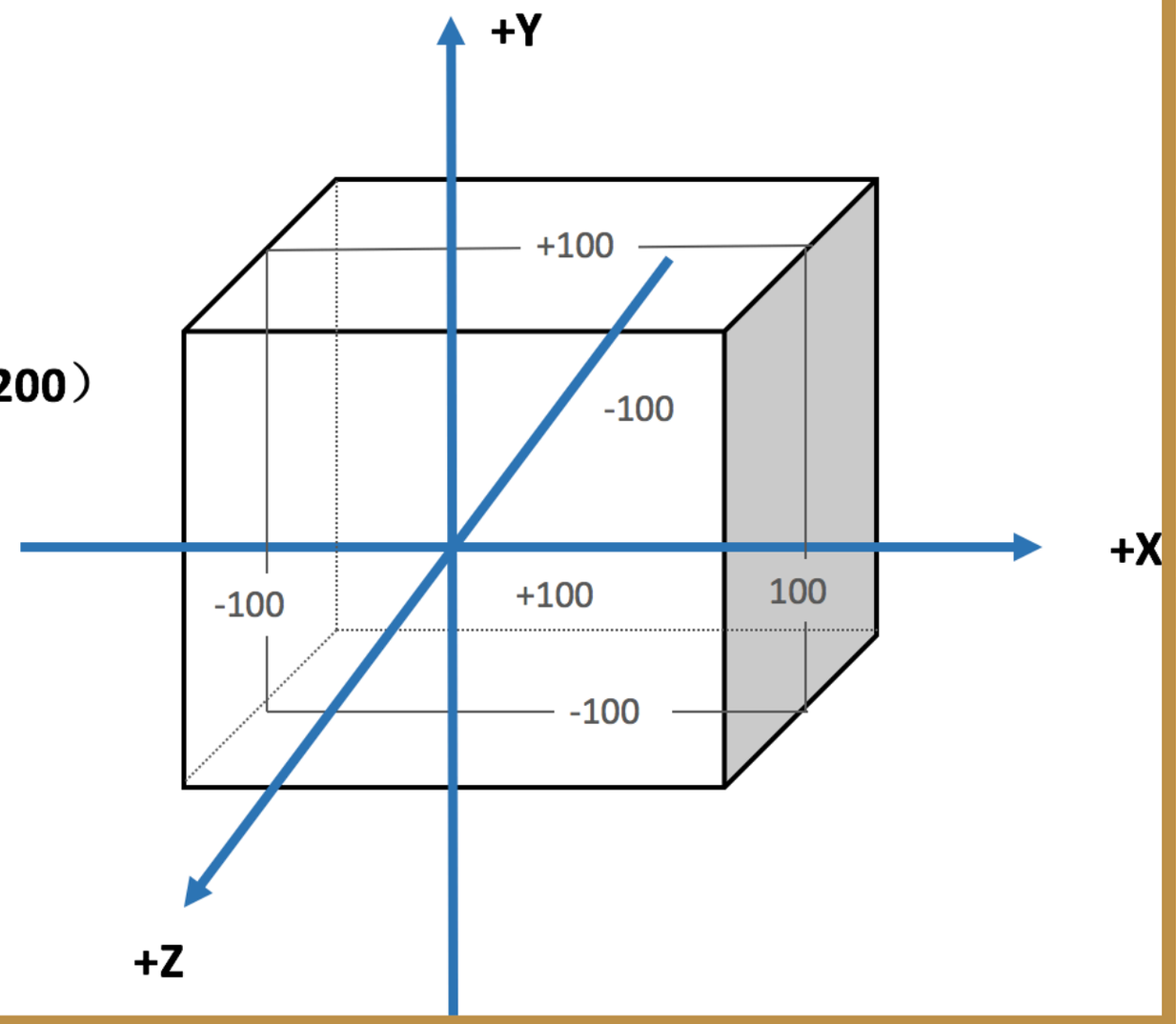
D. 纹理单元数据给顶点着色器实际上是没有必要的.

课程研发:CC老师
课程授课:CC老师

正投影

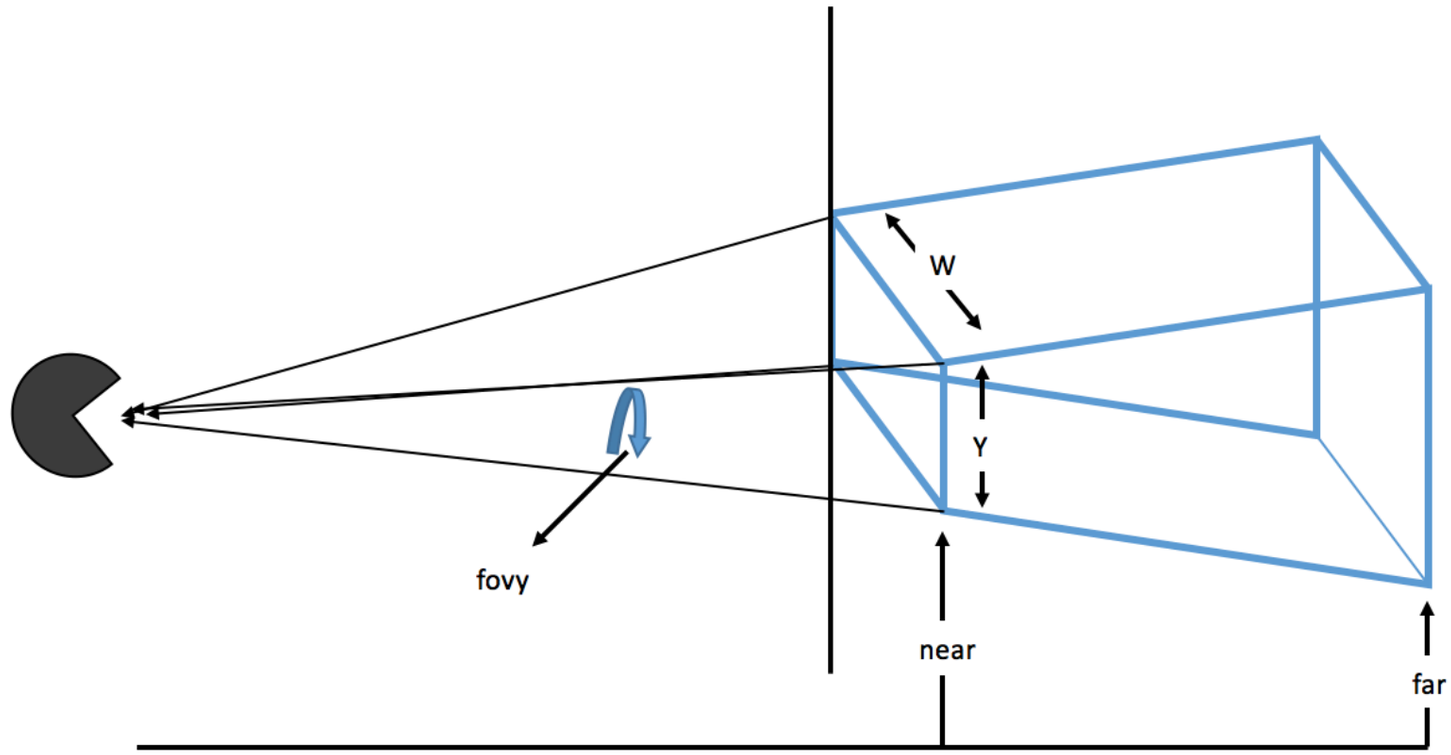
GLFrumstum::SetOrthographic(GLfloat xMin,GLfloat xMax,GLfloat yMin,
GLfloat yMax ,GLfloat zMin,GLfloat zMax);

笛卡尔可视区域，大小为（**200*200*200**）



课程研发:CC老师
课程授课:CC老师

透视投影



课程研发:CC老师
课程授课:CC老师

透视投影

GLFrustum类通过setPerspective 方法为我们构建一个平截头体。
CLFrustum::SetPerspective(float fFov , float fAspect ,float fNear ,float fFar);

参数:
fFov:垂直方向上的视场角度
fAspect:窗口的宽度与高度的纵横比
fNear:近裁剪面距离
fFar:远裁剪面距离

纵横比 = 宽(w)/高(h)

课程研发:CC老师
课程授课:CC老师

课堂练习题:

下面的说明正确的是:[ ]

A. 正投影不能使用在3D图形上

B. 透视投影不能正确渲染平面图形

C. 正投影和透视投影的目的是为了让3D图形映射成二维平面坐标

D. 正投影只能渲染平面图形

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

存储着色器初始化

```
// GLShaderManager 的初始化
GLShaderManager shaderManager;
shaderManager.InitializeStockShaders();
```


课程研发:CC老师
课程授课:CC老师

单元着色器

```
GLShaderManager::UserStockShader(GLT_SHADER_IDENTITY,GLfloat vColor[4]);
```

参数1: 存储着色器种类-单元着色器

参数2: 颜色

使用场景: 绘制默认OpenGL 坐标系(-1,1)下图形. 图形所有片段都会以一种颜色填充

课程研发:CC老师
课程授课:CC老师

平面着色器

```
GLShaderManager::UserStockShader(GLT_SHADER_FLAT,GLfloat mvp[16],GLfloat vColor[4]);
```

参数1: 存储着色器种类-平面着色器
参数2: 允许变化的4*4矩阵
参数3: 颜色

使用场景: 在绘制图形时, 可以应用变换(模型/投影变化).

课程研发:CC老师
课程授课:CC老师

上色着色器

```
GLShaderManager::UserStockShader(GLT_SHADER_SHADED,GLfloat mvp[16]);
```

参数1: 存储着色器种类-上色着色器
参数2: 允许变化的4*4矩阵

使用场景: 在绘制图形时, 可以应用变换(模型/投影变化) 颜色将会平滑地插入到顶点之间称为平滑着色.

课程研发:CC老师
课程授课:CC老师

默认光源着色器

```
GLShaderManager::UserStockShader(GLT_SHADER_DEFAULT_LIGHT,GLfloat mvMatrix[16],GLfloat pMatrix[16],GLfloat vColor[4]);
```

参数1: 存储着色器种类-默认光源着色器
参数2: 模型4*4矩阵
参数3: 投影4*4矩阵
参数4: 颜色值

使用场景: 在绘制图形时, 可以应用变换(模型/投影变化) 这种着色器会使绘制的图形产生阴影和光照的效果.

课程研发:CC老师
课程授课:CC老师

点光源着色器

```
GLShaderManager::UserStockShader(GLT_SHADER_POINT_LIGHT_DIEF,GLfloat mvMatrix[16],GLfloat pMatrix[16],GLfloat vLightPos[3],GLfloat vColor[4]);
```

参数1: 存储着色器种类-点光源着色器
参数2: 模型4*4矩阵
参数3: 投影4*4矩阵
参数4: 点光源的位置
参数5: 漫反射颜色值

使用场景: 在绘制图形时, 可以应用变换(模型/投影变化) 这种着色器会使绘制的图形产生阴影和光照的效果.它与默认光源着色器非常类似,区别只是光源位置可能是特定的.

课程研发:CC老师
课程授课:CC老师

纹理替换矩阵着色器

```
GLShaderManager::UserStockShader(GLT_SHADER_TEXTURE_REPLACE,GLfloat mvMatrix[16],GLint nTextureUnit);
```

参数1: 存储着色器种类-纹理替换矩阵着色器
参数2: 模型4*4矩阵
参数3: 纹理单元

使用场景: 在绘制图形时, 可以应用变换(模型/投影变化)这种着色器通过给定的模型视图投影矩阵.使用纹理单元来进行颜色填充.其中每个像素点的颜色是从纹理中获取.

课程研发:CC老师
课程授课:CC老师

纹理调整着色器

```
GLShaderManager::UserStockShader(GLT_SHADER_TEXTURE_MODULATE,GLfloat mvMatrix[16],GLfloat vColor[4],GLint nTextureUnit);
```

参数1: 存储着色器种类-纹理调整着色器
参数2: 模型4*4矩阵
参数3: 颜色值
参数4: 纹理单元

使用场景: 在绘制图形时, 可以应用变换(模型/投影变化)这种着色器通过给定的模型视图投影矩阵. 着色器将一个基本色乘以一个取自纹理单元nTextureUnit 的纹理.将颜色与纹理进行颜色混合后才填充到片段中.

课程研发:CC老师
课程授课:CC老师

纹理光源着色器

```
GLShaderManager::UserStockShader(GLT_SHADER_TEXTURE_POINT_LIGHT_DIEF,GLfloat mvMatrix[16],GLfloat pMatrix[16],GLfloat vLightPos[3],GLfloat vBaseColor[4],GLint nTextureUnit);
```

参数1: 存储着色器种类-纹理光源着色器
参数2: 模型4*4矩阵
参数3: 投影4*4矩阵
参数4: 点光源位置
参数5: 颜色值(几何图形的基本色)
参数6: 纹理单元

使用场景: 在绘制图形时, 可以应用变换(模型/投影变化)这种着色器通过给定的模型视图投影矩阵. 着色器将一个纹理通过漫反射照明计算进行调整(相乘).

课程研发:CC老师
课程授课:CC老师

课堂练习题:

下面的归类合理的是:[CD]

A. 顶点着色器,片元着色器,细分着色器,纹理替换矩阵着色器

B. 平面着色器,单元着色器,顶点着色器,点光源着色器

C. 顶点着色器,片元着色器,细分着色器 属于可编程管线下的着色器

D. 单元着色器,默认光源着色器,平面着色器,纹理光源着色器 属于固定管线下的着色器

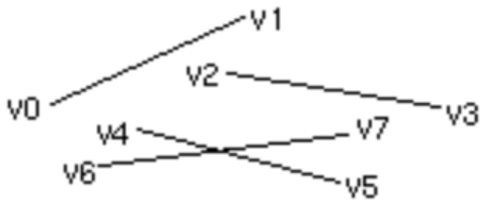
课程研发:CC老师
课程授课:CC老师

OpenGL 基本7种基本图元

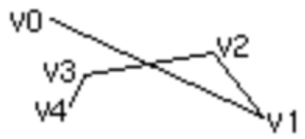
图元	描述
GL_POINTS	每个顶点在屏幕上都是单独点
GL_LINES	每一对顶点定义一个线段
GL_LINE_STRIP	一个从第一个顶点依次经过每一个后续顶点而绘制的线条
GL_LINE_LOOP	和GL_LINE_STRIP相同,但是最后一个顶点和第一个顶点连接起来了.
GL_TRIANGLES	每3个顶点定义一个新的三角形
GL_TRIANGLE_STRIP	共用一个条带(strip)上的顶点的一组三角形
GL_TRIANGLE_FAN	以一个圆点为中心呈扇形排列,共用相邻顶点的一组三角形

课程研发:CC老师
课程授课:CC老师

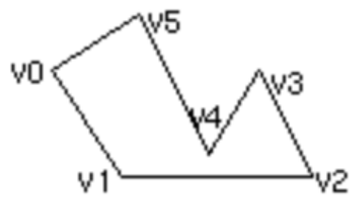
OpenGL 基本基本图元



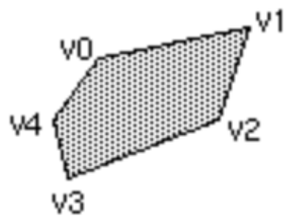
GL_LINES



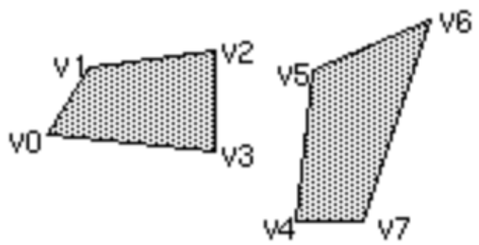
GL_LINE_STRIP



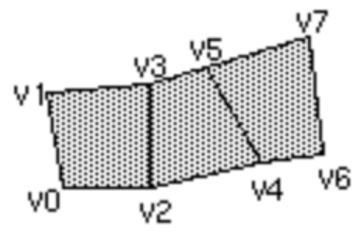
GL_LINE_LOOP



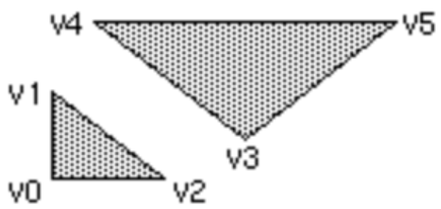
GL_POLYGON



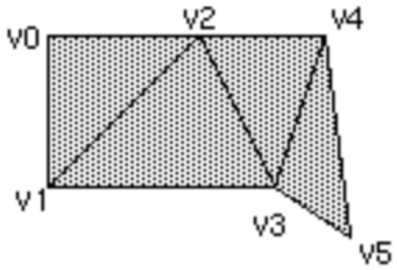
GL_QUADS



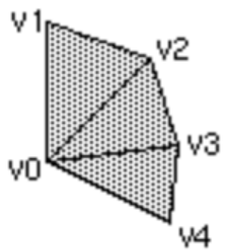
GL_QUAD_STRIP



GL_TRIANGLES



GL_TRIANGLE_STRIP



GL_TRIANGLE_FAN

课程研发:CC老师
课程授课:CC老师

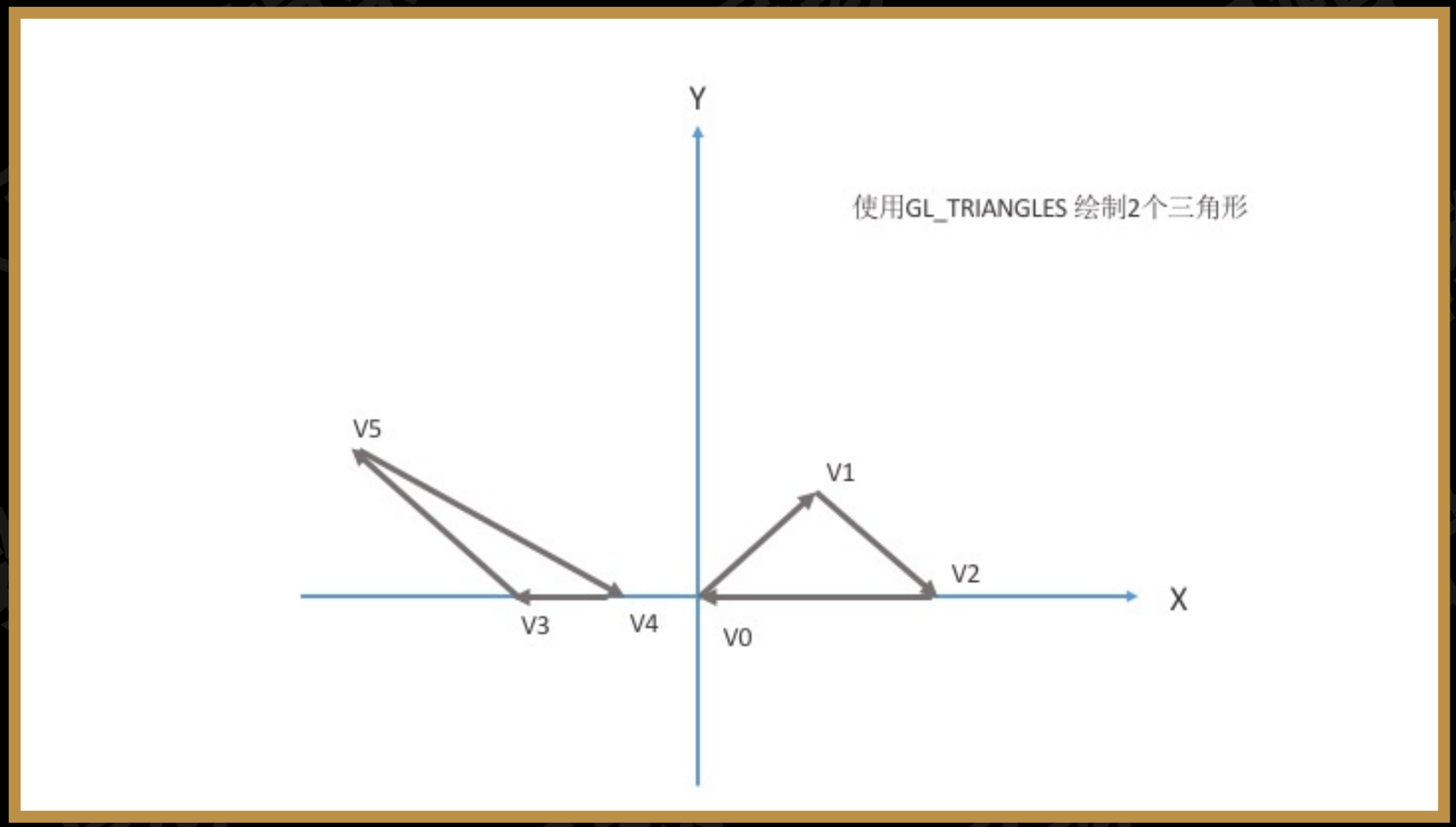
OpenGL 点/线

```
//1.最简单也是最常用的 4.0f,表示点的大小
  glPointSize(4.0f);

//2.设置点的大小范围和点与点之间的间隔
GLfloat sizes[2] = {2.0f,4.0f};
GLfloat step = 1.0f;

//3.获取点大小范围和最小步长
glGetFloatv(GL_POINT_SIZE_RANGE ,sizes);
glGetFloatv(GL_POINT_GRAULARITY ,&step);

//4.通过使用程序点大小模式来设置点大小
glEnable(GL_PROGRAM_POINT_SIZE);

//5.这种模式下允许我们通过编程在顶点着色器或几何着色器中设置点大小。着色器内建变量:
gl_PointSize, 并且可以在着色器源码直接写
 gl_PointSize = 5.0

//6. 设置线段宽度
  glLineWidth(2.5f);
```

课程研发:CC老师
课程授课:CC老师

OpenGL 三角形

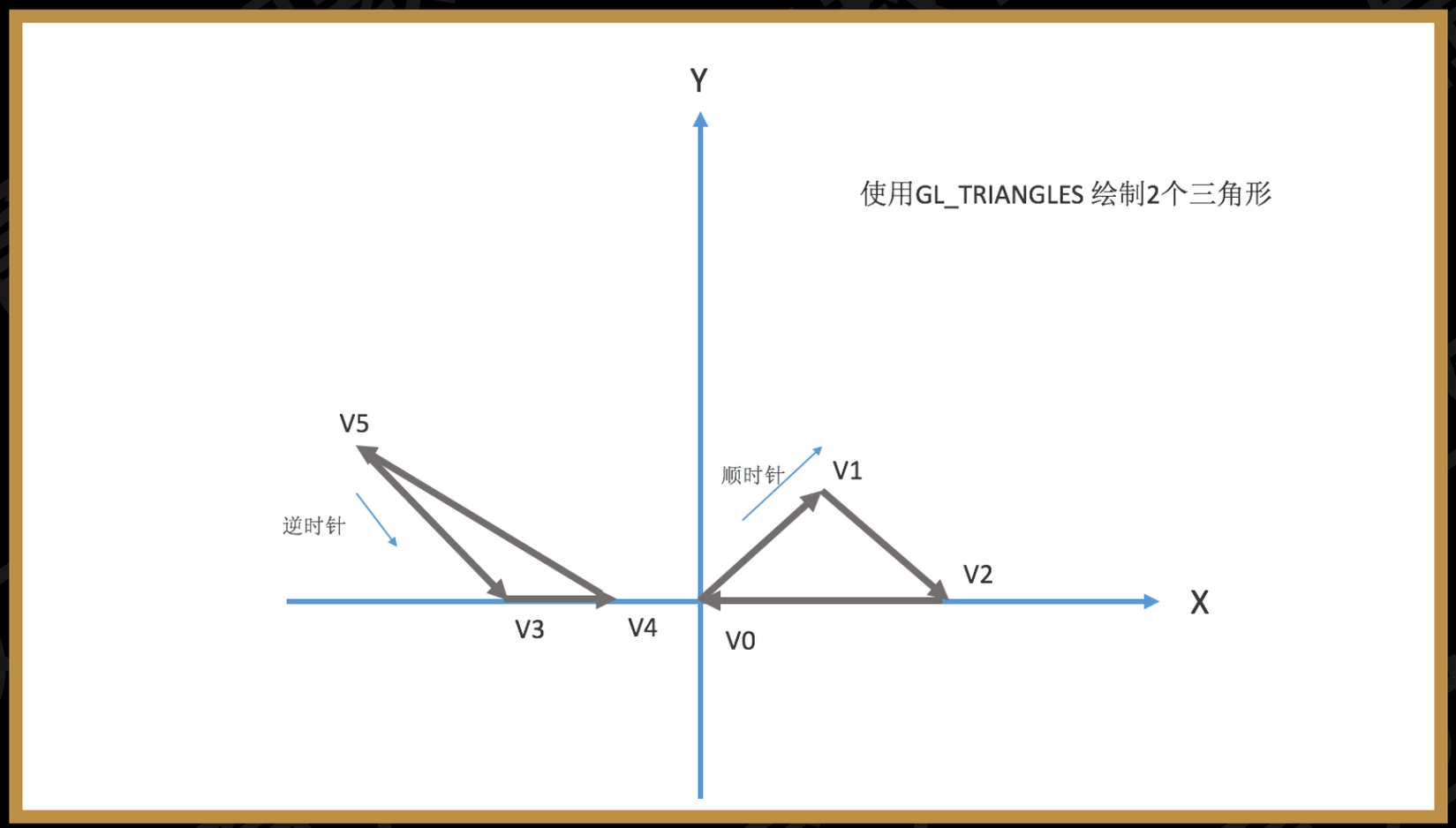
对于OpenGL 光栅化最欢迎的是三角形.3个顶点就能构成一个三角形. 三角形类型来自于顶点.并不是所有的三角形都是正三角形等.



课程研发:CC老师
课程授课:CC老师

OpenGL 三角形环绕方式

在绘制第一个三角形时，线条是按照从V0-V1，再到V2。最后再回到V0的一个闭合三角形。 这个是沿着顶点顺时针方向。这种顺序与方向结合来指定顶点的方式称为环绕



课程研发:CC老师
课程授课:CC老师

OpenGL 三角形环绕方式

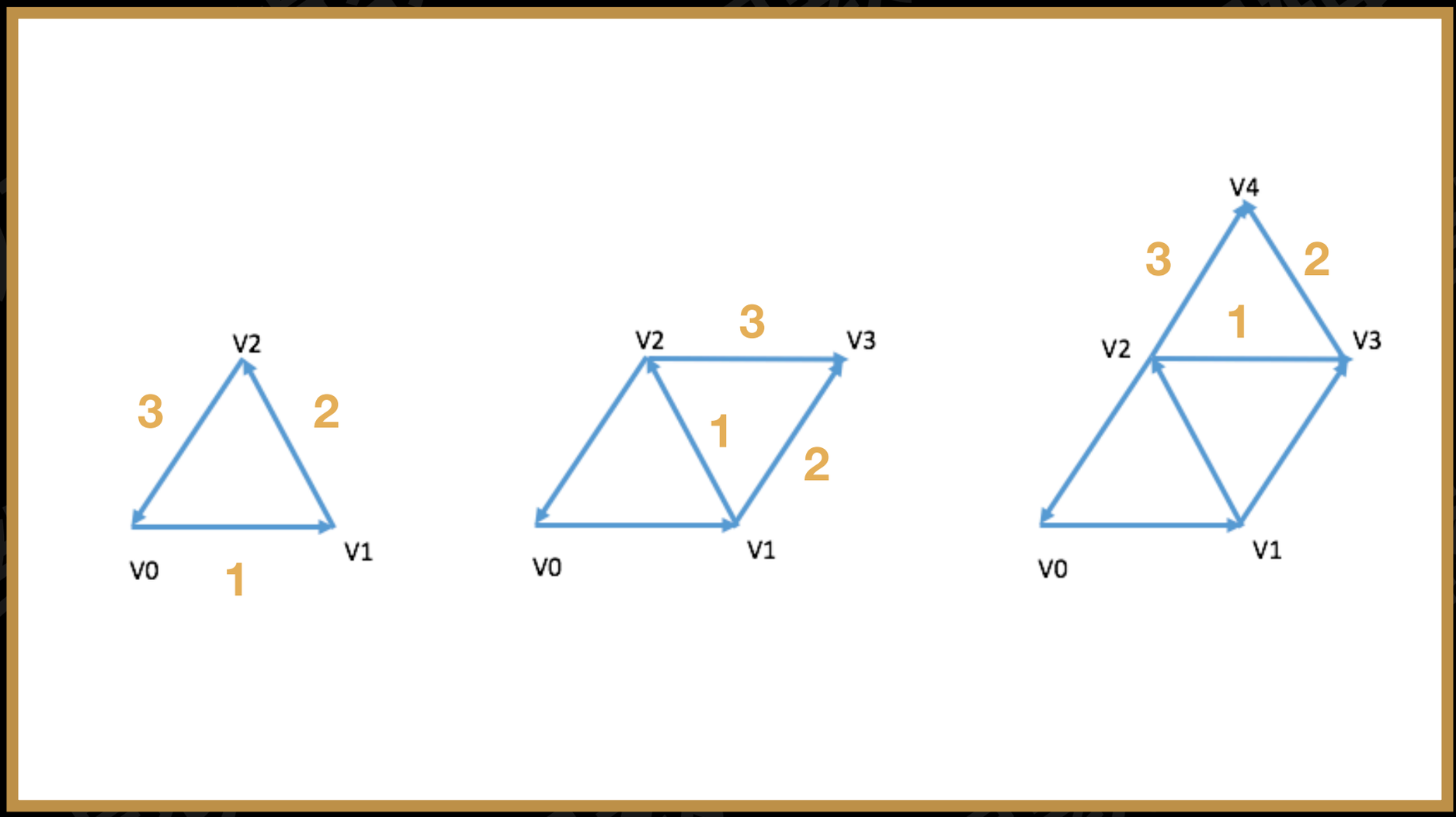
在默认情况下,OpenGL 认为具有逆时针方向环绕的多边形为正面. 这就意味着上图左边是正面,右边是反面.

```
glFrontFace(GL_CW);
GL_CW:告诉OpenGL 顺时针环绕的多边形为正面;
GL_CCW:告诉OpenGL 逆时针环绕的多边形为正面;
```

课程研发:CC老师
课程授课:CC老师

OpenGL 三角形带

对于很多表面或者形状而言,我们会需要绘制几个相连的三角形. 这是我们可以使用GL_TRIANGLE_STRIP 图元绘制一串相连三角形,从而节省大量的时间.



课程研发:CC老师
课程授课:CC老师

OpenGL 三角形带

优点:

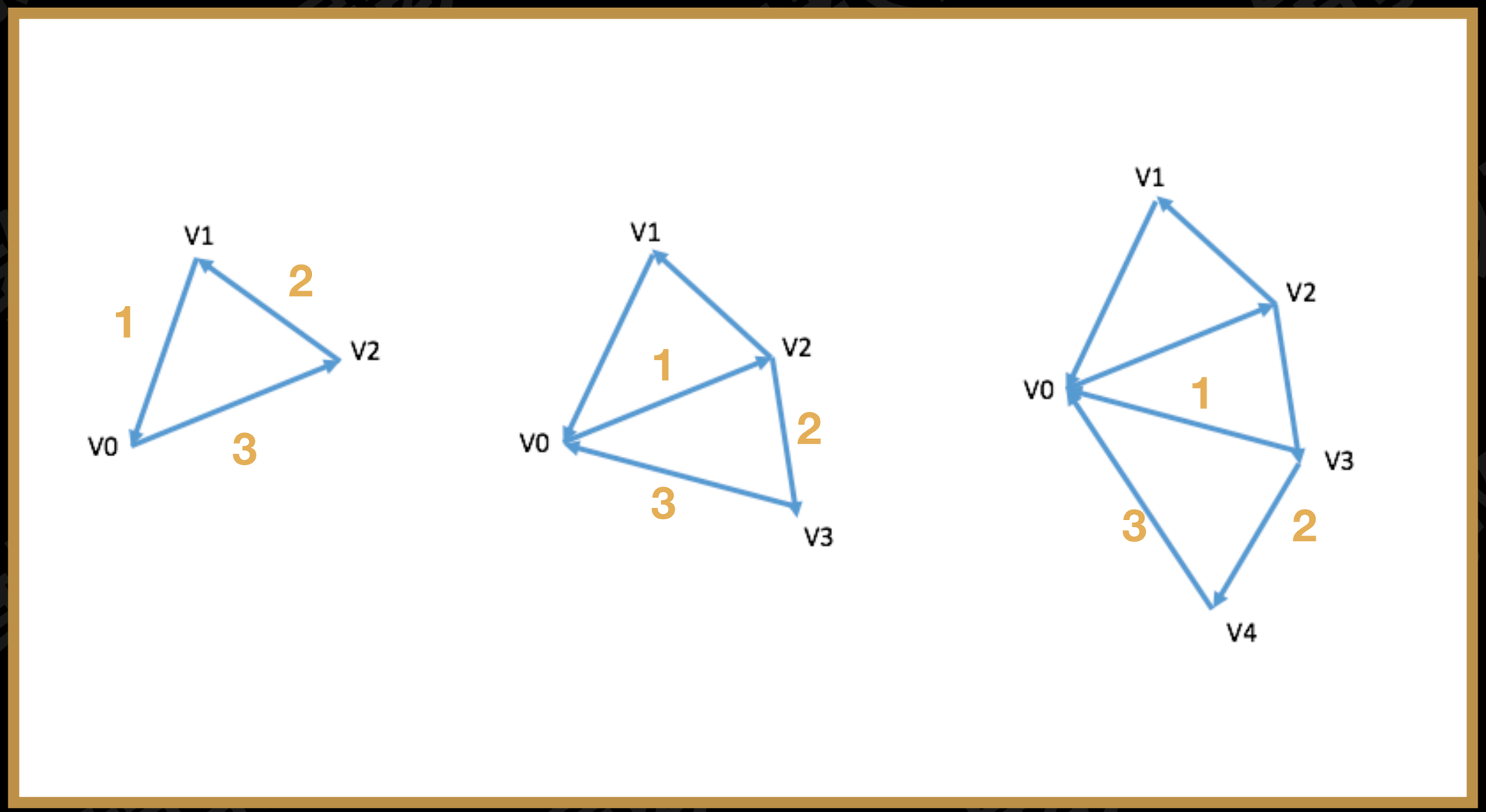
1. 用前3个顶点指定第1个三角形之后，对于接下来的每一个三角形，只需要再指定1个顶点。需要绘制大量的三角形时，采用这种方法可以节省大量的程序代码和数据存储空间

2.提供运算性能和节省带宽。更少的顶点意味着数据从内存传输到图形卡的速度更快，并且顶点着色器需要处理的次数也更少了。

课程研发:CC老师
课程授课:CC老师

OpenGL 三角形扇

对于很多表面或者形状而言,我们会需要绘制几个相连的三角形. 这是我们可以使用GL_TRIANGLE_FAN 图元绘制一组围绕一个中心点相连的三角形



课程研发:CC老师
课程授课:CC老师

OpenGL 工具类 GLBatch

GLBatch ,是在GLTools中包含的一个简单容器类.

```
void GLBatch::Begain(GLeunm primitive,GLuint nVerts,GLuint nTexttureUnints = 0);

参数1：图元
参数2：顶点数
参数3：一组或者2组纹理坐标（可选）

//复制顶点数据(一个由3分量x,y,z顶点组成的数组)
void GLBatch::CopyVerterxData3f(GLfloat *vVerts);

//复制表面法线数据
void GLBatch::CopyNormalDataf(GLfloat *vNorms);

//复制颜色数据
void GLBatch::CopyColorData4f(GLfloat *vColors);

//复制纹理坐标数据
void GLBatch::CopyTexCoordData2f(GLFloat *vTextCoords, GLuint uiTextureLayer);

//结束数据复制
void GLBatch::End(void);

//绘制图形
void GLBatch::Draw(void);
```

课程研发:CC老师
课程授课:CC老师

使用固定存储着色器熟悉OpenGL下多种图元组合实现

案例1: 001—OpenGL 图元绘制

课程研发:CC老师
课程授课:CC老师