Primitive Assembly（图元组装）

示例代码见：OpenGL_ES_Samples/02_PrimitiveAssembly

Primitive（图元）

什么是图元?

图元就是能直接用glDrawArray或glDrawElement绘制出来的几何体。图元是由一系列的Vertex（顶点）来描述，每个顶点包含多个Attribute，比如位置、纹理坐标、法线方向等。

OpenGL ES 有哪些图元?

Points

- 一个Vertex表示一个Point，在VertexShader中可以用内置的输出变量gl_PointSize来控制点的大小。
- 一个大小不为1的点，表示以这个点的位置为中心，边长为gl_PointSize的正方形。
- 点的大小是有限制范围的，可以通过以下方式获取：

```
GLfloat   pointSizeRange[2];
glGetFloatv(GL_ALIASED_POINT_SIZE_RANGE, pointSizeRange);
```

- 在FragmentShader中有一个内置变量 gl_PointCoord用来表示点在屏幕中的位置(待续)

Lines

- 两个点表示一个Line，对一串Vertex数据，有三种Line解释方法：GL_LINES、GL_LINE_STRIP和GL_LINE_LOOP。
- 线段可以通过函数glLineWidth来设置线的宽度。
- 线的宽度是有限制的，可以通过以下方式获取：

```
GLfloat   lineWidthRange[2];
glGetFloatv(GL_ALIASED_LINE_WIDTH_RANGE, lineWidthRange);
```

Trangles

- 三个点表示一个Trangle，对一串Vertex数据，有三种Trangle的解释方法：GL_TRIANGLES,、GL_TRIANGLE_STRIP和GL_TRIANGLE_FAN。

Performence Tips

- glDrawArray 速度快，因为内存连续；费内存，因为Vertex数据有重复。
- glDrawElement省内存，因为Vertex数据只有一份；稍微慢点，因为数据不连续。
- 尽量一次绘制尽可能多的Vertex，如果是GL_TRANGLE_STRIP需要注意拼接方法：重复前面的Strip的最后一个Vertex和后面的Strip的第一个Vertex。

Primitive Assembly

通过vertex Shader后标准化的Vertex处理流程为:

- Clipping: 把投影矩阵之外的Primitive抛弃，不同的Primitive有不同的处理方法。
- Perspective Division: 把3D中的Primitive投影到Screen。
- Viewport Transformation: 把3D坐标转换到屏幕坐标。

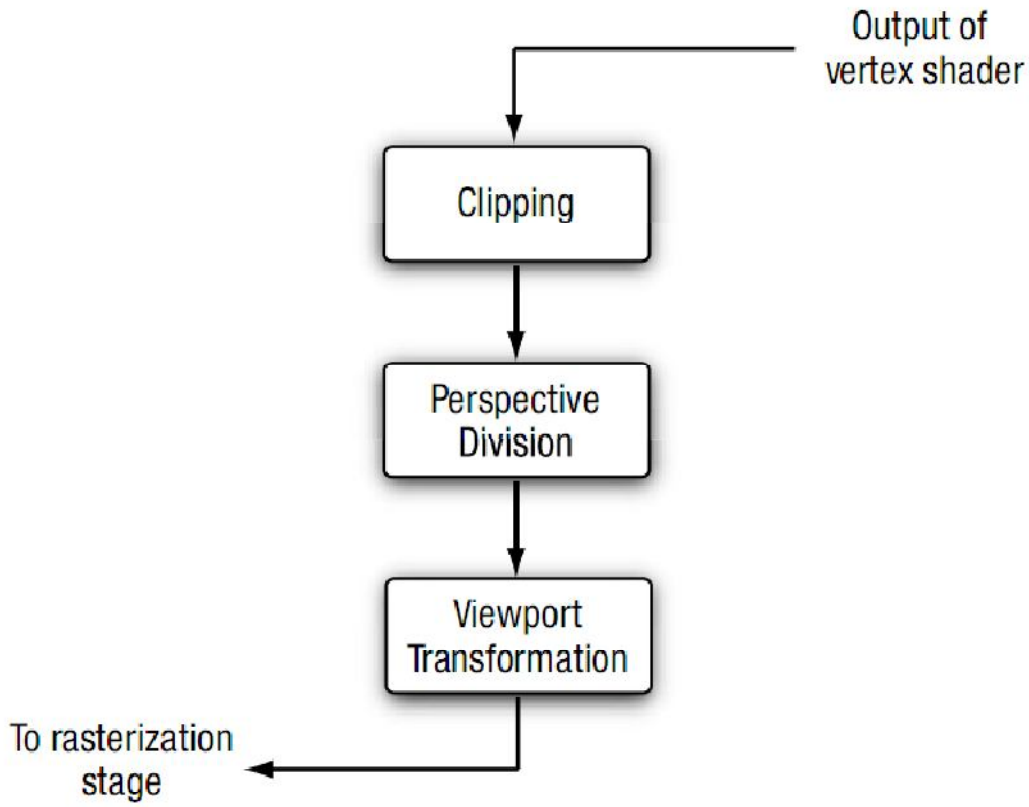

Figure 7-6 OpenGL ES Primitive Assembly Stage

Rasterization

此流程的功能为:把经过组装的每个Primitive，转换为适当的Fragment。(即把抽像的几个概念转换为像素点的集合)

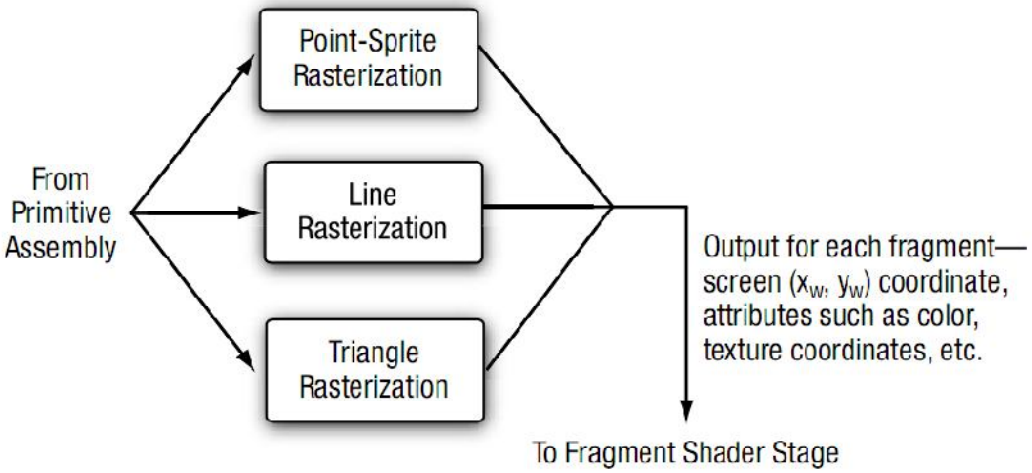

Figure 7-9 OpenGL ES Rasterization Stage