OpenGL ES 3.0 Programming Guide

Chapter 1. Introduction to OpenGL ES 3.0

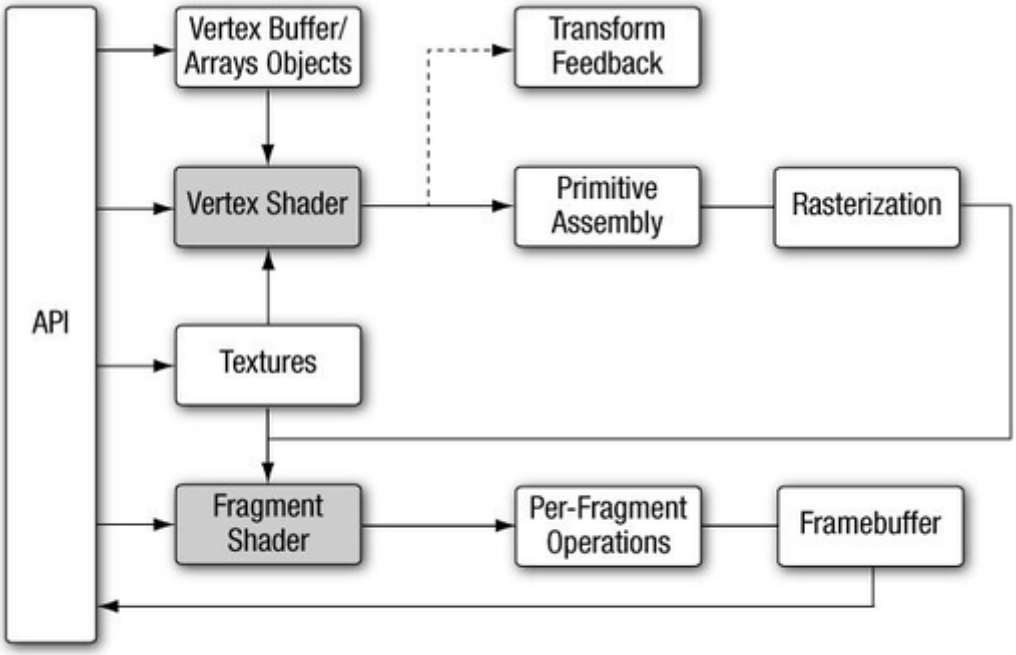

Figure 1-1 OpenGL ES 3.0 Graphics Pipeline

Vertex Shader

- Input
  - Shader program
  - Vertex shader input (or attributes)
  - Uniforms
  - Samplers
- Output
  - Vertex shader output variable
    - OpenGL ES 2.0에서는 varying
- Transform feedback
  - Vertex shader output을 output buffer로 보냄
  - output buffer로 보낸 vertex shader output은 fragment shader로 보낼 수도 있고 보내지 않을 수도 있음
  - output buffer는 vertex shader에서 input으로 사용 가능

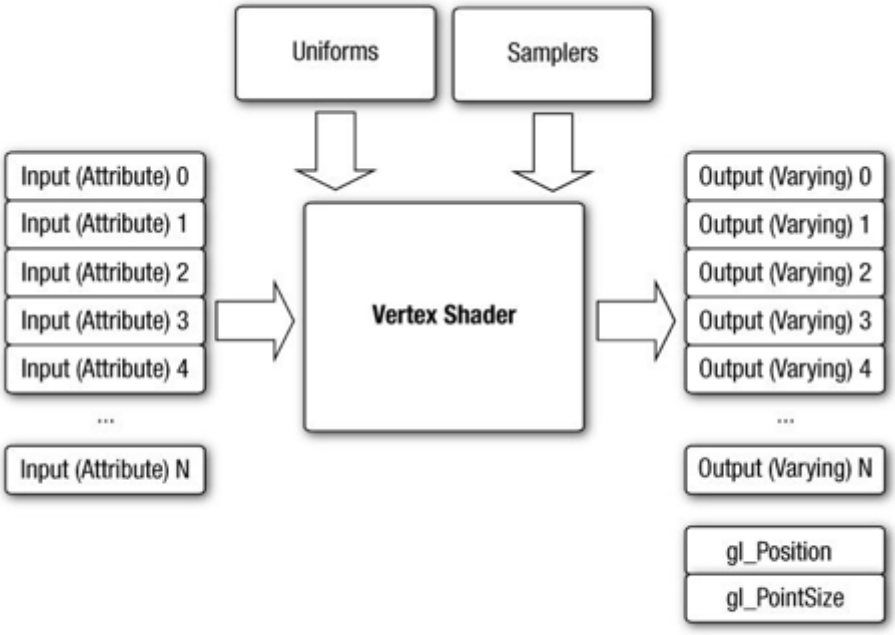

Figure 1-2 OpenGL ES 3.0 Vertex Shader

- 역할
  - 좌표 변환
  - Per-vertex light 계산
  - Per-vertex color 계산
  - Texture 좌표 생성 또는 변환
- Sample code

```
1.   #version 300 es
2.   uniform mat4 u_mvpMatrix; // matrix to convert a_position
3.                            // from model space to normalized
4.                            // device space
5.
6.   // attributes input to the vertex shader
7.   in vec4 a_position;       // position value
8.   in vec4 a_color;          // input vertex color
9.
10.  // output of the vertex shader - input to fragment
11.  // shader
12.  out vec4 v_color;         // output vertex color
13.  void main()
14.  {
15.     v_color = a_color;
16.     gl_Position = u_mvpMatrix * a_position;
17.  }
```

Primitive Assembly

- Vertex shader의 다음 stage
- Primitive
  - Triangle, line 또는 point sprite와 같은 geometric object
- Clipping

    - 각 primitive가 view frustum 내에 있는지에 따라 clipping 수행 여부 결정
    - Primitive가 view frustum 에 걸쳐 있으면 clipping 수행
    - Primitive가 view frustum 밖에 있으면 discard
- Clipping 후에 vertex 좌표는 스크린 좌표로 변환
- Culling
    - front face인지, back face인지에 따라 primitive를 discard할지 결정
- 해당 stage가 끝나면 rasterization stage로 넘어간다.

Rasterization

- Primitive를 two-dimensional fragment로 변환하는 과정
- Two-dimensional fragment
    - 화면에 그려지는 pixel을 나타냄

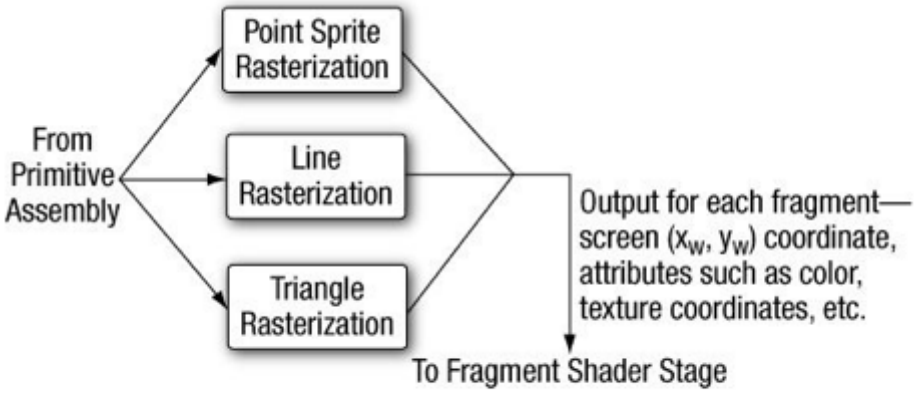

Figure 1-3 OpenGL ES 3.0 Rasterization Stage

Fragment Shader

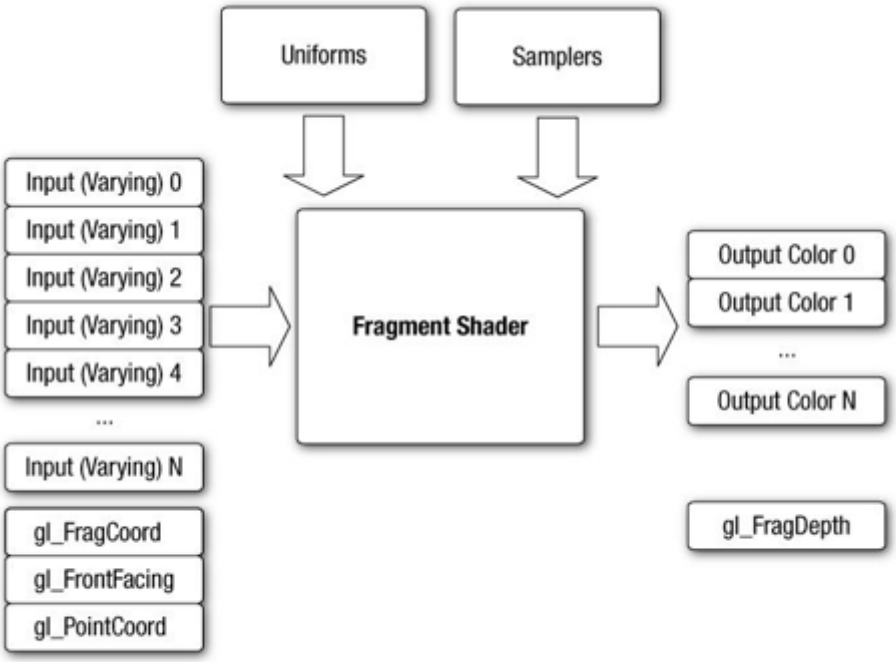

Figure 1-4 OpenGL ES 3.0 Fragment Shader

- Rasterization stage에서 생성된 fragment를 처리
- Input
    - Shader program
    - Input variables
    - Uniforms
    - Samplers
- 역할
    - Fragment를 discard
    - Color 결정
- Sample code

```
1.  #version 300 es
2.  precision mediump float;
3.
4.  in vec4 v_color;     // input vertex color from vertex shader
5.
6.  out vec4 fragColor; // output fragment color
7.  void main()
8.  {
9.     fragColor = v_color;
10. }
```

Per-Fragment Operations

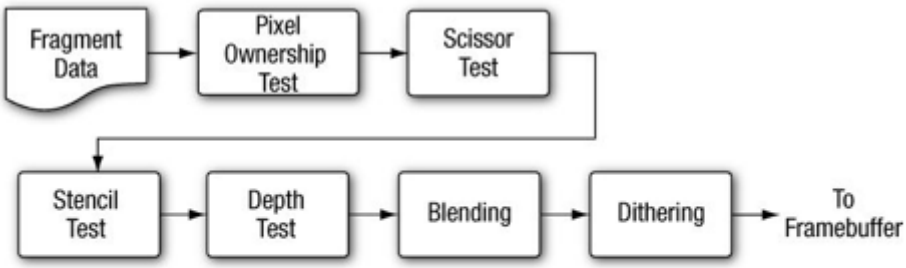

Figure 1-5 OpenGL ES 3.0 Per-Fragment Operations

- 각 fragment에 대해서 다음 function이 수행된다.
- Pixel ownership test
    - Framebuffer의 특정 위치에 있는 pxiel이 OpenGL ES에 의해서 소유되고 있는지를 확인한다.
    - Window system에서 해당 pixel을 화면에 보여줄지 말지를 결정할 수 있도록 해준다.
- Scissor test
    - Fragment가 scissor rectangle 내에 있는지 확인한다.
    - Fragment가 scissor rectangle 내에 없으면 discard
- Stencil and depth tests
    - 해당 위치의 Stencil과 depth value를 사용해서 fragment를 그릴지 말지를 결정한다.

- Blending
    - Framebuffer에 있는 color와 새로 그려질 fragment color를 blending
- Dithering
    - Framebuffer에 color value를 저장하기 위해서 사용되는 precision이 제한되기 때문에 발생하는 문제를 최소화한다.