# 3. OpenGL 다루기

### 3.1 Setting Up a 2D Drawing Canvas : Coordinate System

2차원 평면에 그림을 그리기 위해서는 좌표계(Coordinate System)가 필요로 하다. 실제로 사용자의 마우스 입력에 의해 그려지는 곳은 모니터의 윈도우이기 때문에 윈도우 좌표계를 알아야 하며, 가상공간을 다루는 것은 OpenGL이기 때문에 OpenGL에서 사용되는 좌표계를 알아야 한다.

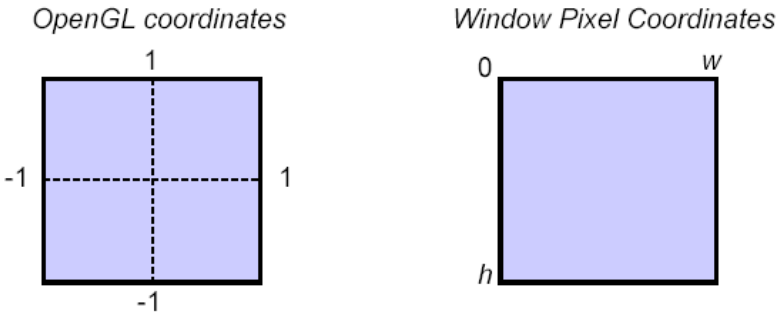


**Fig 3.1 OpenGL 좌표계와 Window 좌표계**

OpenGL 좌표계는 우리가 알고 있는 좌표계와 동일하다. 즉, 우축으로 갈수록 x 값이 증가하고, 위로 갈수록 y 값이 증가한다. 반대로 window 좌표계에서는 실제로 우축으로 갈수록 x 값이 증가하지만 아래로 향할수록 y 값이 증가하는 시스템을 따른다. OpenGL 좌표계 영역 [-1,1]x[-1,1]과 window 좌표계 영역 [0,w]x[0,h]이 대응된다고 가정하자. 윈도우에서 특정한 위치에 점을 찍고자 하여 마우스의 커서를 그 위치에 놓고 클릭을 하여 위치를 (x, y)로 받아들인다면, OpenGL 좌표계에서는 아래의 (s,t)로 바뀌게 된다.

$$2:w = s+1:x, \qquad 2:h = t+1:h-y$$
$$s = \frac{2}{w}x - 1, \qquad t = \frac{2}{h}(h-y) - 1$$

### 3.2 Geometric Primitives

OpenGL에서 사용되는 기본적인 기하학적 프리미티브들은 다음의 세 개로 구성된다.

- **Points**
- **Line Segments**
- **Filled Polygon**

점은 그 자체 하나의 점으로 그 프리미티브를 나타낼 수 있으며, 선분은 두 개의 점에 의해서 정의되며, 다각형은 여러 개의 점들의 집합으로 구성된다. 그러므로, OpenGL에서는 점들로 표현하는 방법으로 여러 종류의 프리미티브를 나타낼 수 있다. glBegin()함수와 glEnd()함수 사이에 원하는 수의 정점을 정의하면 원하는 형태의  프리미티브를 정의하게 되며 윈도우에 그려지게 된다. 먼저 점들을 그리는 코드를 살펴보자.

*Drawing 2-D Points:*

```
extern Vec2 v[k];

glBegin(GL_POINTS);
  for(int i=0; i<k; i++)
    glVertex2dv(v[i]);
glEnd();
```

*Drawing 3-D Points:*

```
extern Vec3 v[k];

glBegin(GL_POINTS);
  for(int i=0; i<k; i++)
    glVertex3dv(v[i]);
glEnd();
```

**Fig 3.2 2차원 점과 3차원 점 그리기 OpenGL 코드**

그리려는 것이 점집합이기 때문에 glBegin() 함수의 인자가 **GL_POINTS** 가 되어야 한다. 위 예제의 코드에서는 점들이 벡터 형태로 표현되어 있기 때문에 점을 정의하는 함수가 벡터타입을 사용하고 있음을 확인할 수 있다. 선분을 그리려고 한다면 다음의 코드로 변경하면 된다. 점을 그리는 코드와 달라진 부분이 무엇인지 살펴보자. 우선 인자가 **GL_LINES** 로 바뀌었고 정점들이 짝수 개 정의되어 있어야 한다.

```
glBegin(GL_LINES);
   glVertex2f(0.0, 0.0);
   glVertex2f(1.0, 0.0);
   glVertex2f(0.0, 0.0);
   glVertex2f(0.0, 1.0);
glEnd();
```

**Fig 3.3 선분 그리기 OpenGL 코드**

다음은 다양한 종류의 선분 프리미티브를 정의하는 방법에 대해서 살펴보도록 한다.

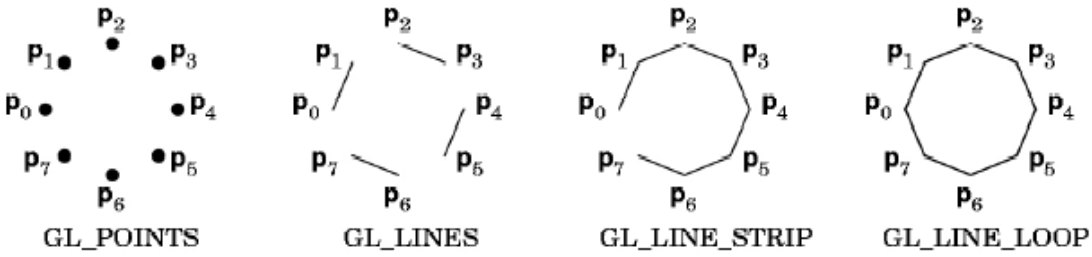


**Fig 3.4 다양한 선분 Type**

다각형은 일반적으로 Loop과 Interior을 합한 것을 말하게 되는데 내부가 존재한다고 보면 된다. 즉, 어떠한 색상으로 채워질 수 있는 프리미티브이다. 컴퓨터그래픽스에서는 3차원에서의 렌더링 Quality를 위하여 몇 가지 가정을 적용한다. 사용하는 다각형은 simple하고 flat하며 convex하다고 가정한다. 다각형이 simple 하다는 것은 다각형의 선분들끼리의 교차는 정점에서만 일어난다는 것인데, 별 모양을 한붓그리기로 그리게 되면 선분의 내부 점에서 교차가 일어나는데, 이와 같은 다각형은 피하겠다는 의미이다. 다각형이 Flat하다는 것은 2차원 공간에서는 사실이지만 3차원 공간에서는 사실이 아닐 수도 있다. 예를 들면, 3차원 공간에서 4개의 정점에 의해서 만들어지는 다각형은 일반적으로 하나의 평면에 놓여 있다

고 보기는 힘들다. 왜냐하면, 3개의 점정이 하나의 평면을 구성하기 때문에 특별히 나머지 1개의 정점이 그 평면에 놓일 수도 있지만 그렇지 않을 확률이 더 많다고 볼 수 있다. 그러므로 하나의 다각형이 2개의 평면으로 이루어지는 경우가 발생하게 되는 셈이다. 따라서 이런 문제를 미리 방지하기 위해서 다각형이 하나의 평면에 놓인다고 가정하게 된다. 또한 다각형이 오목(concave)하다면 모양이 여러 다각형의 합집합으로 인식될 수도 있기 때문에 볼록 형태의 모양을 사용한다. 따라서 convex, simple, flat 하다는 조건을 다 만족하는 것은 삼각형이므로 컴퓨터그래픽스에서는 대부분의 물체들을 삼각 메수 형태로 물체를 표현하는 데이터 구조를 사용한다.

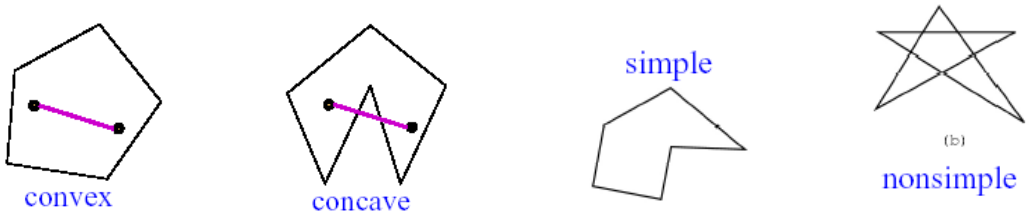


**Fig 3.5 Convex, Concave, Simple, Nonsimple**

다각형을 그리는 일반적인 프리미티브 코드는 기본적으로 glBegin()함수와 glEnd()함수로 사용되며 glBegin()함수의 인자로 들어가는 Type에 따라 사용된 정점들로부터 다양한 형태의 다각형을 만들 수 있다.

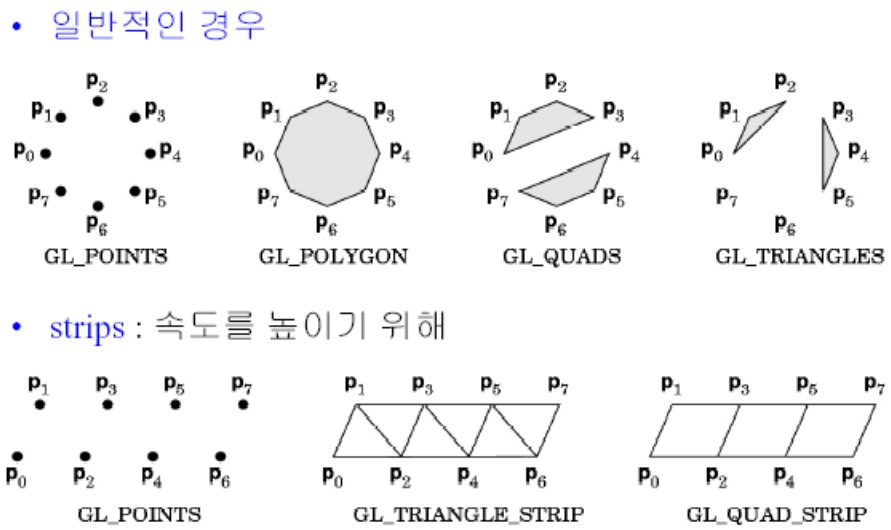


**Fig 3.6 다양한 다각형 타입들**