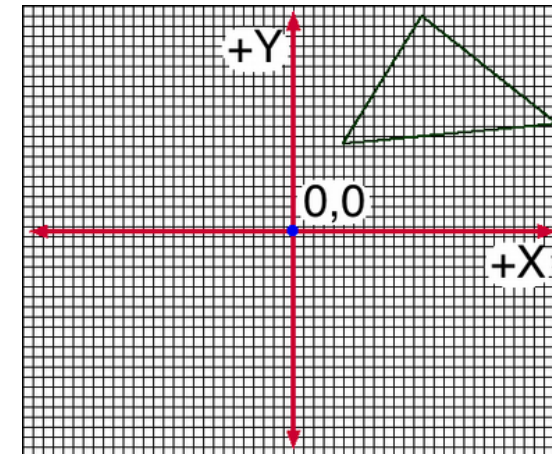


Graphics Programming

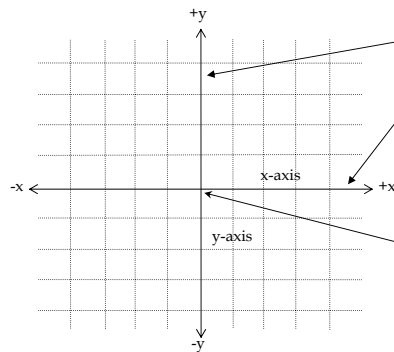
321190
2009년 봄학기
3/10/2009
박경신

Coordinate Systems



2D Cartesian Coordinate Systems

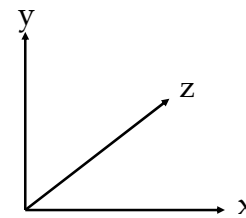
□ Cartesian Coordination Systems



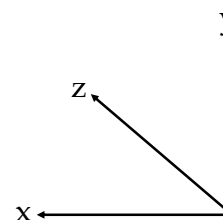
Two axes: **x-axis** and **y-axis**, two straight lines perpendicular to each other, both pass through origin and extends infinitely in two opposite directions

원점 (Origin)은 좌표계의 중심에 위치하고 있고 값은 (0, 0)이다.

3D Cartesian Coordinate Systems

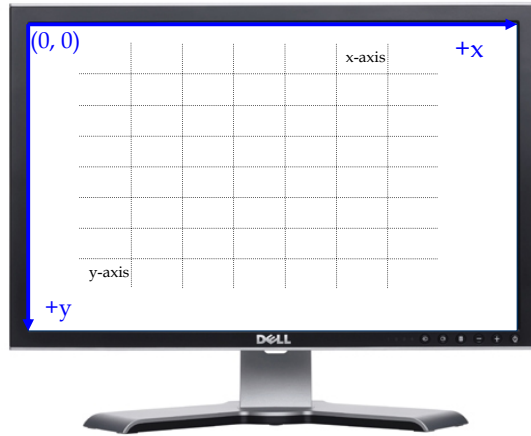


□ 왼손 좌표계 (Left-handed coordinate system)는 x+는 오른쪽, y+는 위쪽, z+는 화면안쪽.



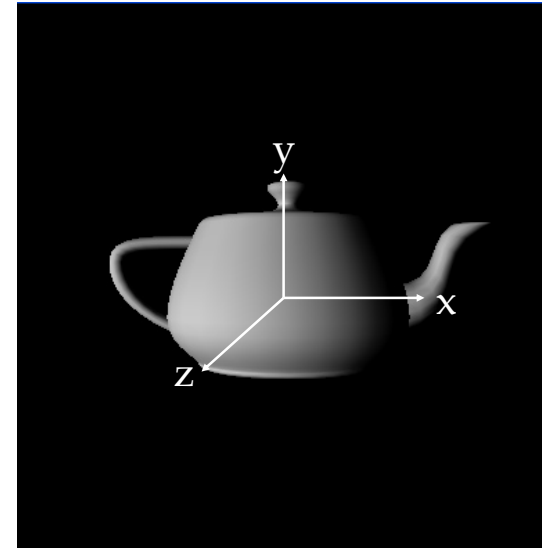
□ 오른손 좌표계 (Right-handed coordinate system)는 x+는 왼쪽, y+는 위쪽, z+는 화면안쪽.

Screen Coordinate System



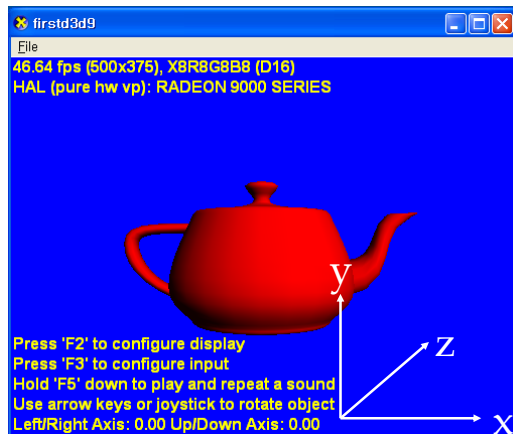
- Screen coordinate system은 원점 (Origin)이 화면의 좌측상단에 위치하고 값은 (0, 0)이다. x+ 오른쪽. y+ 아래쪽.
- 1 unit = 1 pixel

3D Coordinate Systems



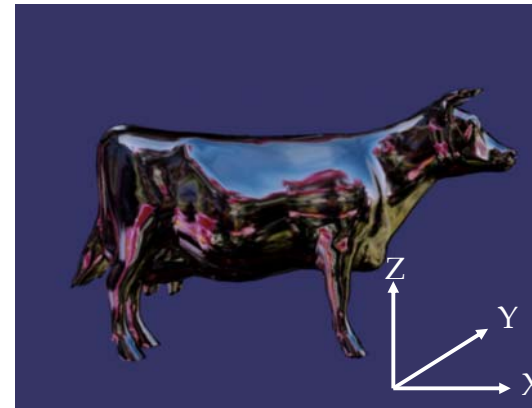
- OpenGL은 오른손 좌표계 (Right-handed coordinate system)
- x+ 오른쪽. y+ 위쪽. z+ 화면 밖으로 나오는 방향.

3D Coordinate Systems



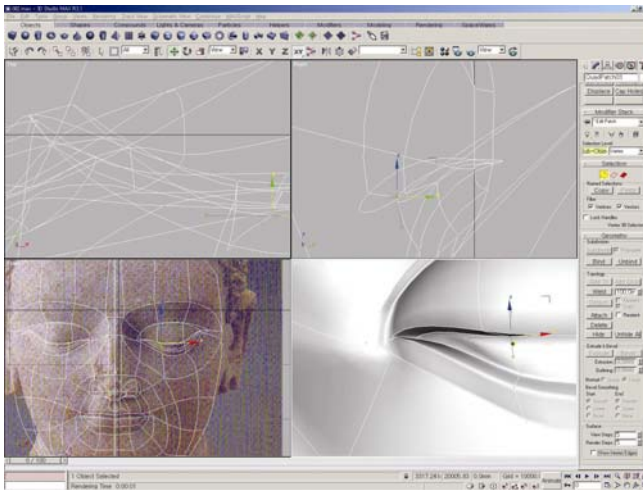
- Direct3D는 왼손 좌표계 (Left-handed coordinate system).
- x+ 오른쪽. y+ 위쪽. z+ 화면 안쪽으로 들어가는 방향.

3D Coordinate Systems



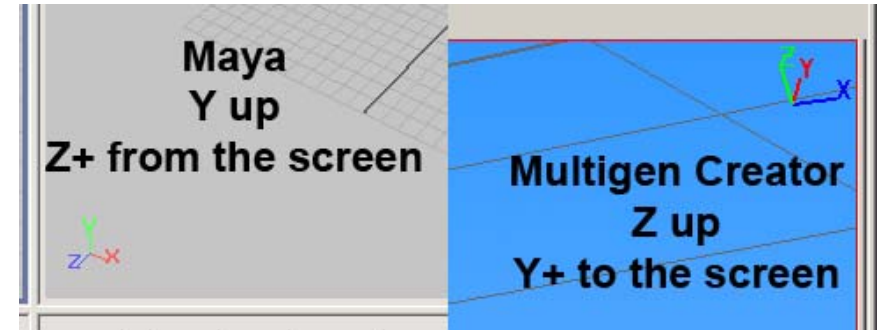
- Open Scene Graph (OSG)은 오른손 좌표계 (Right-Handed Coordinate System) x+ 오른쪽. y+ 화면 안으로 향하는 방향. z+ 위쪽.

3D Coordinate Systems



- 3D Studio Max은 오른손 좌표계 (Right-Handed Coordinate System) x+ 오른쪽, y+ 화면 안으로 향하는 방향, z+ 위쪽.

3D Coordinate Systems



- Maya - same as OpenGL
- Multigen Creator - same as OSG

Graphics Programming

- OpenGL & GLUT API를 사용한 그래픽스 프로그래밍 소개
- OpenGL Programming Guide "red book"
- OpenGL Reference Manual "blue book"
- OpenGL Shading Language "orange book"

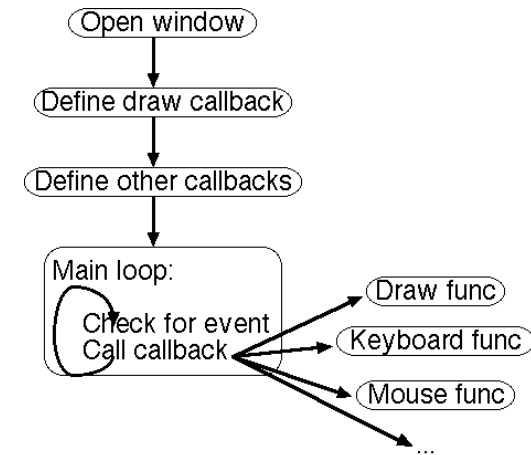
OpenGL

- OpenGL은 그래픽스 파이프라인 (graphics pipeline)을 모델로 사용하는 3D graphics library이다.
- OpenGL은 window system에 독립적인 API
- OpenGL은 operating system에 독립적인 API
- OpenGL libraries:
 - GL - graphics
 - GLU - utilities
 - GLX (X window)/ WGL (Windows)/ AGL (Mac) / PGL (IBM) - windowing toolkits
 - GLUT - multi-platform windowing API

GLUT (OpenGL Utility Toolkit)

- Mark J. Kilgard가 개발한 portable windowing and interaction API
 - Prefix “glut”로 시작
 - 대부분의 window system에 보편적인 기능들을 wrapping한 상위 interface제공 (portable across all PC and workstation OS platforms)
 - OpenGL이 제공하는 범위보다 상위 수준의 utility function도 제공
 - UNIX/X-window에서 개발된 code를 그대로 재사용 가능
 - Win32, MFC, Xlib을 알 필요가 없음
 - 그러나 Window system의 기능을 제한적으로만 이용 가능

GLUT Program Structure



OpenGL API

- 그래픽스 함수 (Graphics Functions)
 - 기본 요소 함수 (Primitive functions)
 - 속성 함수 (Attribute functions)
 - 관측 함수 (Viewing functions) - chap5
 - 변환 함수 (Transformation functions) - chap4
 - 입력 함수 (Input functions) - GLUT chap3
 - 제어 함수 (Control functions) - GLUT
 - 질의 함수 (Query functions)

Graphics Functions

- 기본 요소 함수 (Primitive functions)
 - 저수준 객체 (Low-level object)을 정의
 - Points, line segments, polygons, pixels, texts, curves, surfaces
- 속성 함수 (Attribute functions)
 - 기본 요소가 화면상에 나타날 수 있는 방법을 제어
 - Color, line thickness, pattern to fill the polygon, ..
- 관측 함수 (Viewing functions)
 - 합성카메라 (synthetic camera)모델 - 카메라 위치, 방향, 렌즈
 - Viewport, clipping, ..
- 변환 함수 (Transformation functions)
 - 물체의 회전(rotation), 이동(translation), 크기변환 (scaling)

Graphics Functions

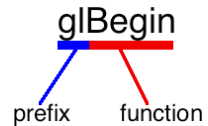
- 입력 함수 (Input functions)
 - 키보드, 마우스, 태블릿 등 입력장치를 다루는 함수
- 제어 함수 (Control functions)
 - 윈도우 시스템과 통신을 가능하게 하고, 프로그램 초기화, 에러처리 등
 - Close, open, resolution setting, mode setting, ..
- 질의 함수 (Query functions)
 - 그래픽스 시스템의 내부적인 상태 (internal states)이나 속성 (properties) 정보

OpenGL Syntax

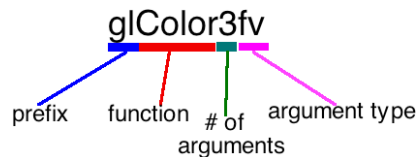
- Basic OpenGL Syntax
 - 함수(function)에 prefix "gl" 사용
 - E.g.: glBegin, glClear, glCopyPixels, glPolygonMode
 - 기호상수(symbolic constant)에 prefix "GL" 사용 & 각각의 단어는 '_'로 연결
 - E.g.: GL_2D, GL_RGB, GL_CCW, GL_POLYGON, GL_AMBIENT_AND_DIFFUSE
 - 자료타입 (data type)에 prefix "GL" 사용
 - E.g. GLbyte, GLshort, GLint, GLfloat, GLdouble, GLboolean

OpenGL Functions

- OpenGL 함수 이름은 "gl"로 시작
- 함수 인자 (argument)가 하나인 경우:

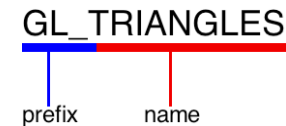


- 함수 인자 (argument)를 여러 개 받는 경우:



OpenGL Constants

- OpenGL의 상수 (Constant)는 함수 인자에 사용
- 상수의 이름은 전부 대문자를 사용하며 "GL"로 시작:



e.g. glBegin()에 들어갈 수 있는 인자들 (arguments)의 예:

- GL_POINTS
- GL_LINES
- GL_TRIANGLES
- GL_POLYGON

OpenGL State

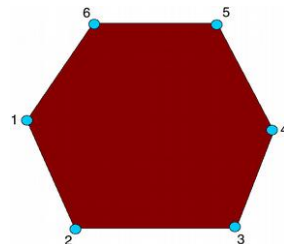
- GL rendering consists of geometry + state
- Both are passed to graphics hardware via function calls
- To draw something, the necessary state attributes (e.g. color) are set first. Then, the geometry (e.g. triangle data) is passed
- State is retained until changed.
- State changes do not affect any geometry already drawn.

OpenGL State

- State:
 - Color
 - Drawing style
 - Material properties
 - Light sources
 - Texture
 - Transformations

OpenGL Geometry

- 가상의 공간을 구성하는 각 물체를 표현하는데 있어 가장 기본이 되는 요소
- 실시간 그래픽스에서는 주로 가장 단순한 형태의 표현 방법인 **linear primitives**를 사용
 - Point, vertex
 - Line segments
 - Polygon
 - Polyhedron



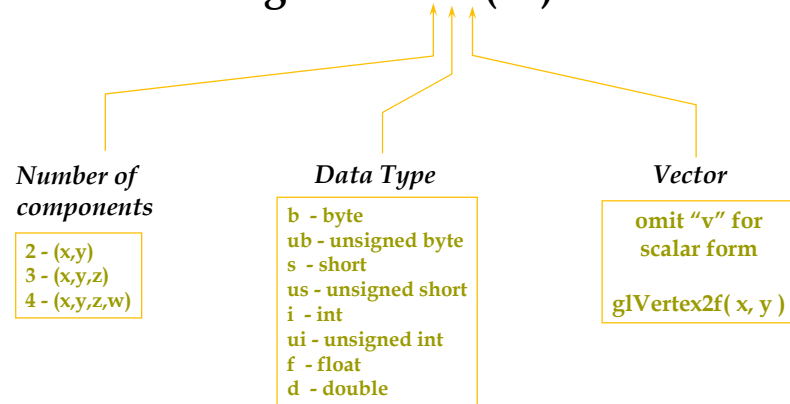
OpenGL Geometry

- 기하학적 객체를 정의하기 위해서는:

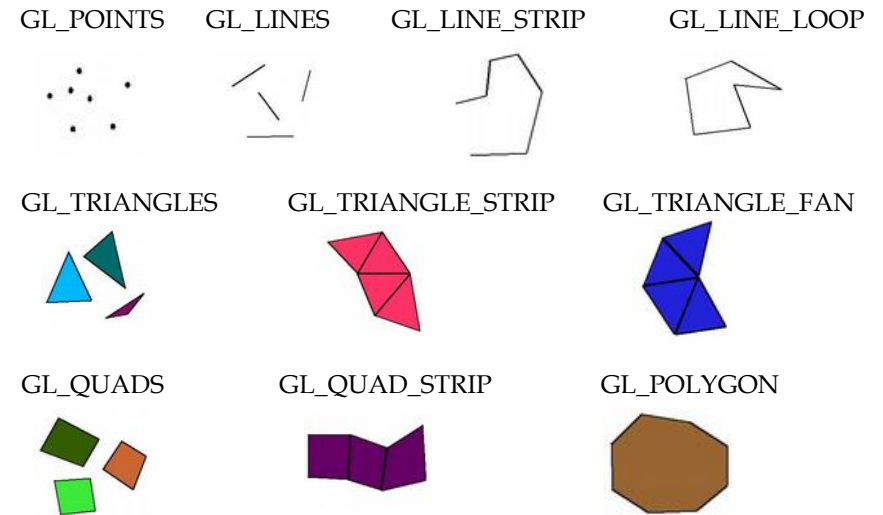
```
glBegin(...);  
glVertex*(...);  
...  
glEnd();
```
- `void glBegin(GLenum mode)`
`void glEnd(void)`
 - 도형을 그리려면, `glBegin()`과 `glEnd()` 사이에 그 도형의 각 정점(vertex)의 좌표치를 설정하는 함수를 둬. `mode`에 `GL_POINTS`, `GL_LINES`, `GL_POLYGON` 등 도형의 타입을 지정.
- `void glVertex*()`
 - 이 함수는 2차원의 좌표치를 설정하는 사용. 인수의 형태는 `Gldouble`임. Float 형태는 `glVertex2f(..)`를 int 형태는 `glVertex2i(..)`를 사용함.

OpenGL Geometry

glVertex3fv(v)



OpenGL Geometry Primitives



OpenGL Point Function

```
glBegin(GL_POINTS);
    glVertex2i(50, 100);
    glVertex2i(75, 150);
    glVertex2i(100, 200);
glEnd();
```

```
GLint pt1[2] = {50, 100};
GLint pt2[2] = {75, 150};
GLint pt3[2] = {100, 200};
glBegin(GL_POINTS);
    glVertex2iv(pt1);
    glVertex2iv(pt2);
    glVertex2iv(pt3);
glEnd();
```

OpenGL Point Function

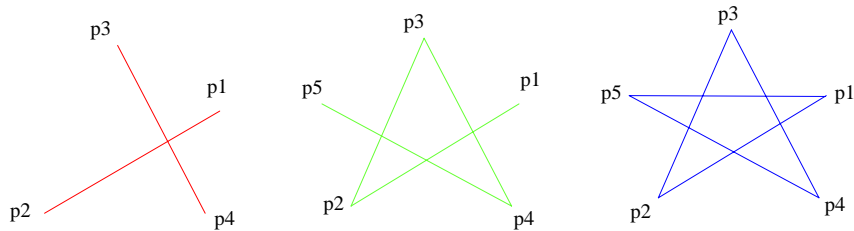
```
glBegin(GL_POINTS);
    glVertex3f(-78.05, 909.72, 14.60);
    glVertex3f(261.91, -5200.67, 188.33);
glEnd();
```

```
class wcPt2D {
public:
    GLfloat x, y;
};
wcPt2D pointPos;
```

```
pointPos.x = 120.75;
pointPos.y = 45.30;
glBegin(GL_POINTS);
    glVertex2f(pointPos.x, pointPos.y);
glEnd();
```

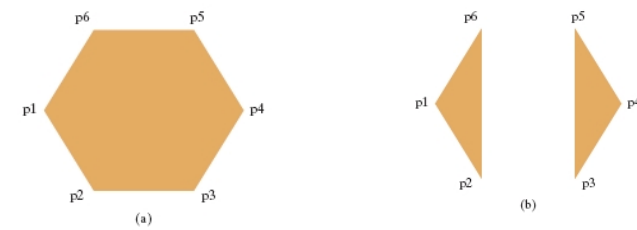
OpenGL Line Function

| | | |
|---------------------------------|--------------------------------------|-------------------------------------|
| <code>glBegin(GL_LINES);</code> | <code>glBegin(GL_LINE_STRIP);</code> | <code>glBegin(GL_LINE_LOOP);</code> |
| <code>glVertex2iv(p1);</code> | <code>glVertex2iv(p1);</code> | <code>glVertex2iv(p1);</code> |
| <code>glVertex2iv(p2);</code> | <code>glVertex2iv(p2);</code> | <code>glVertex2iv(p2);</code> |
| <code>glVertex2iv(p3);</code> | <code>glVertex2iv(p3);</code> | <code>glVertex2iv(p3);</code> |
| <code>glVertex2iv(p4);</code> | <code>glVertex2iv(p4);</code> | <code>glVertex2iv(p4);</code> |
| <code>glVertex2iv(p5);</code> | <code>glVertex2iv(p5);</code> | <code>glVertex2iv(p5);</code> |
| <code>glEnd();</code> | <code>glEnd();</code> | <code>glEnd();</code> |



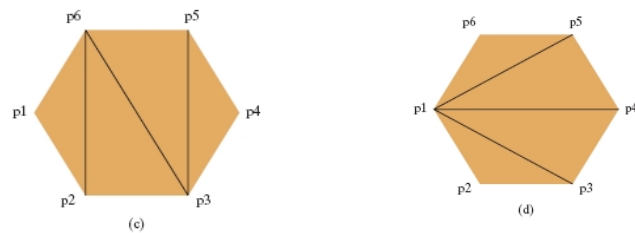
OpenGL Filling Polygon

| | |
|------------------------------------|-------------------------------------|
| <code>glBegin(GL_POLYGONS);</code> | <code>glBegin(GL_TRIANGLES);</code> |
| <code>glVertex2iv(p1);</code> | <code>glVertex2iv(p1);</code> |
| <code>glVertex2iv(p2);</code> | <code>glVertex2iv(p2);</code> |
| <code>glVertex2iv(p3);</code> | <code>glVertex2iv(p6);</code> |
| <code>glVertex2iv(p4);</code> | <code>glVertex2iv(p3);</code> |
| <code>glVertex2iv(p5);</code> | <code>glVertex2iv(p4);</code> |
| <code>glVertex2iv(p6);</code> | <code>glVertex2iv(p5);</code> |
| <code>glEnd();</code> | <code>glEnd();</code> |



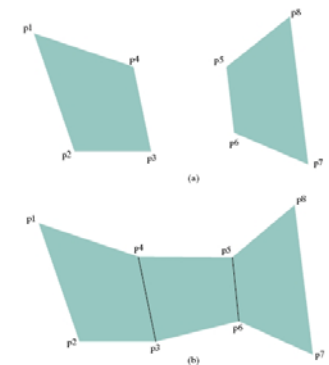
OpenGL Filling Polygon

| | |
|--|--|
| <code>glBegin(GL_TRIANGLE_STRIP);</code> | <code>glBegin(GL_TRIANGLE_FAN);</code> |
| <code>glVertex2iv(p1);</code> | <code>glVertex2iv(p1);</code> |
| <code>glVertex2iv(p2);</code> | <code>glVertex2iv(p2);</code> |
| <code>glVertex2iv(p6);</code> | <code>glVertex2iv(p3);</code> |
| <code>glVertex2iv(p3);</code> | <code>glVertex2iv(p4);</code> |
| <code>glVertex2iv(p5);</code> | <code>glVertex2iv(p5);</code> |
| <code>glVertex2iv(p4);</code> | <code>glVertex2iv(p6);</code> |
| <code>glEnd();</code> | <code>glEnd();</code> |



OpenGL Filling Polygon

| | |
|---------------------------------|--------------------------------------|
| <code>glBegin(GL_QUADS);</code> | <code>glBegin(GL_QUAD_STRIP);</code> |
| <code>glVertex2iv(p1);</code> | <code>glVertex2iv(p1);</code> |
| <code>glVertex2iv(p2);</code> | <code>glVertex2iv(p2);</code> |
| <code>glVertex2iv(p3);</code> | <code>glVertex2iv(p4);</code> |
| <code>glVertex2iv(p4);</code> | <code>glVertex2iv(p3);</code> |
| <code>glVertex2iv(p5);</code> | <code>glVertex2iv(p5);</code> |
| <code>glVertex2iv(p6);</code> | <code>glVertex2iv(p6);</code> |
| <code>glVertex2iv(p7);</code> | <code>glVertex2iv(p8);</code> |
| <code>glVertex2iv(p8);</code> | <code>glVertex2iv(p7);</code> |
| <code>glEnd();</code> | <code>glEnd();</code> |



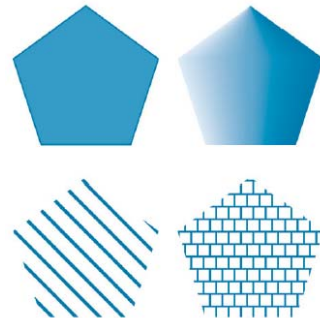
OpenGL Attributes

- 각 기하학적 기본요소 (geometry primitive)는 속성을 갖고 있다. 속성은 기본 요소가 화면상에 나타날 수 있는 방법을 제어한다.

- Color
- Line thickness
- Line styles
- Polygon patterns



선의 두께나 스타일



다각형 표시방법

OpenGL Attributes

- Vertex
 - 점의 크기 - `glPointSize(GLfloat size)`
- Line
 - 선분의 폭 - `glLineWidth(GLfloat width)`
 - 점선의 스타일 - `glLineStipple(GLint factor, GLushort pattern)`
- Polygon
 - 외형과 사용면 지정 - `glPolygonMode(GLenum face, GLenum mode)`
 - Face: `GL_FRONT_AND_BACK`, `GL_FRONT`, `GL_BACK`
 - Mode: `GL_POINT`, `GL_LINE`, `GL_FILL`
 - 앞면 지정 - `glFrontFace(GLenum mode)`
 - Mode: `GL_CCW`, `GL_CW`
 - 스티플링 - `glPolygonStipple(const GLubyte *mask)`
 - Mast: 32x32 비트 윈도우 정렬 스티플 패턴

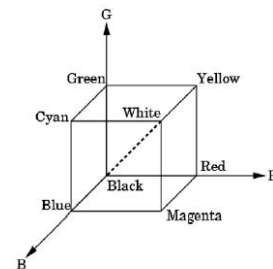
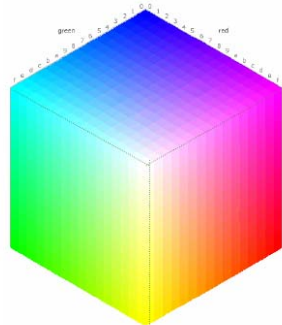
OpenGL Attributes

□ OpenGL Color Model

- RGB (Red Green Blue) or RGBA (Red Green Blue Alpha)
- RGB 색이 따로 분리되어 framebuffer에 저장되어 있음.

`glColor3f(1.0, 0.0, 0.0); // 색값은 0.0 ~ 1.0`

`glColor3ub(255, 0.0, 0.0); // 색값은 0~255`



OpenGL Attributes

□ OpenGL Color Model

- Color Index는 메모리를 적게 사용함. 현재는 크게 쓰이지 않음.

`glIndexi(element);`

