Graphics Programming

Viewing

HyoungSeok Kim

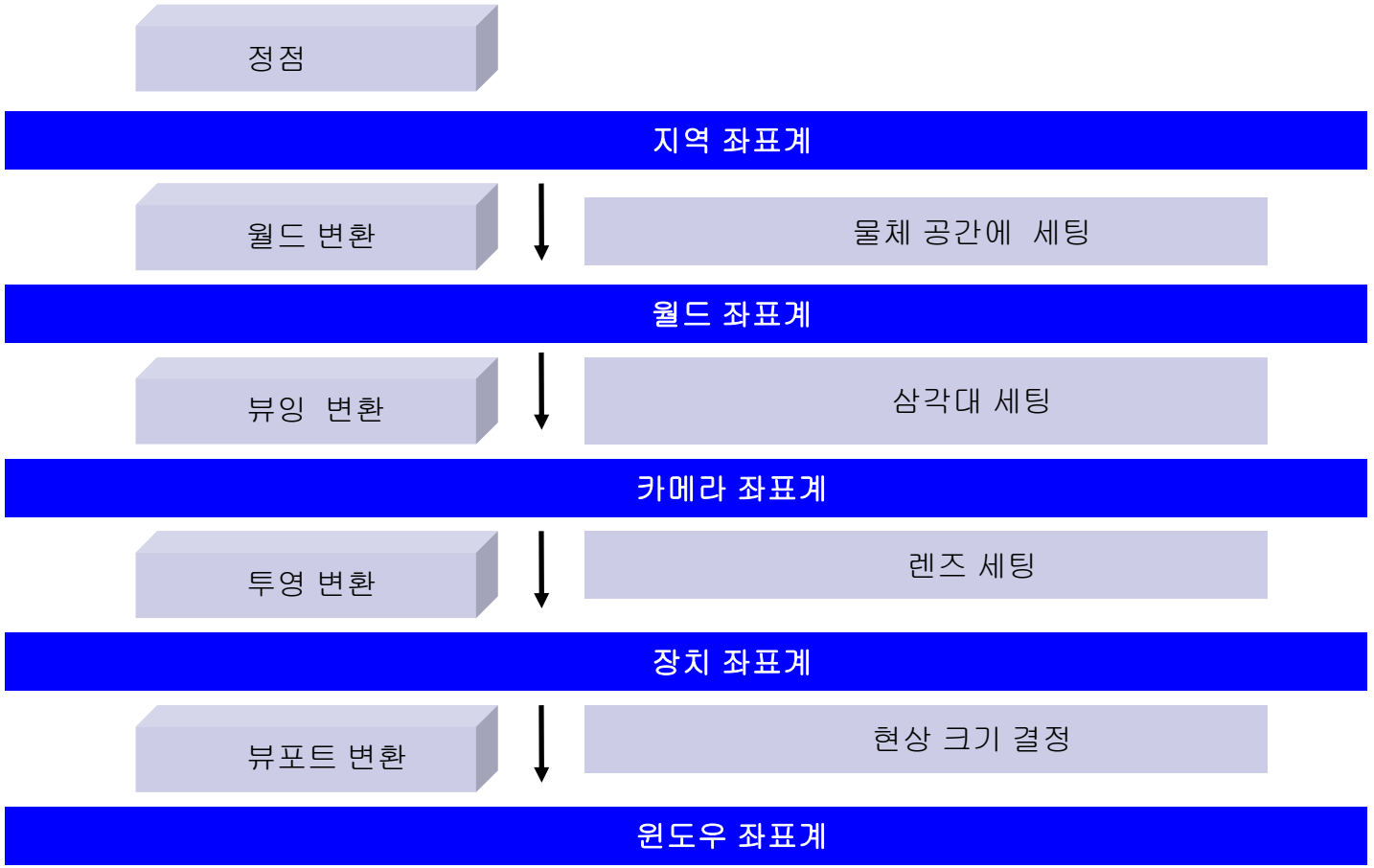
- 컴퓨터그래픽스
  - 컴퓨터를 사용하여 실 세계나 가상 세계를 구축하고 이를 모니터에 나타내게 하는 모든 기술

- 3차원 컴퓨터그래픽스
  - 3차원 객체를 2차원 이미지로 나타내는 모든 과정 및 기술
  - 모델링 + 렌더링
  - 모델링 : 물체를 생성하는 과정
  - 렌더링 : 물체를 카메라로 찍어서 사진으로 현상하는 과정
    - 빛(조명) + 사진(뷰잉)

- 카메라 처리 단계
  1. 삼각대를 세운다. 카메라가 원하는 장면을 향하도록 세운다.
  2. 원하는 장면을 화면에 담도록 모델을 구성한다.
  3. 사용할 카메라 렌즈를 선택하거나 줌을 조절한다.
  4. 사진의 크기를 결정한다.

- 컴퓨터 처리 단계
  1. **Viewing** : 카메라 놓기
  2. **Displacement** : 모델 놓기→ **ModelView** 변환
  3. **Projection** : 관측 공간의 모양 결정하기
  4. **Viewport** : 나타내어지는 이미지의 크기 지정

Viewing Pipeline



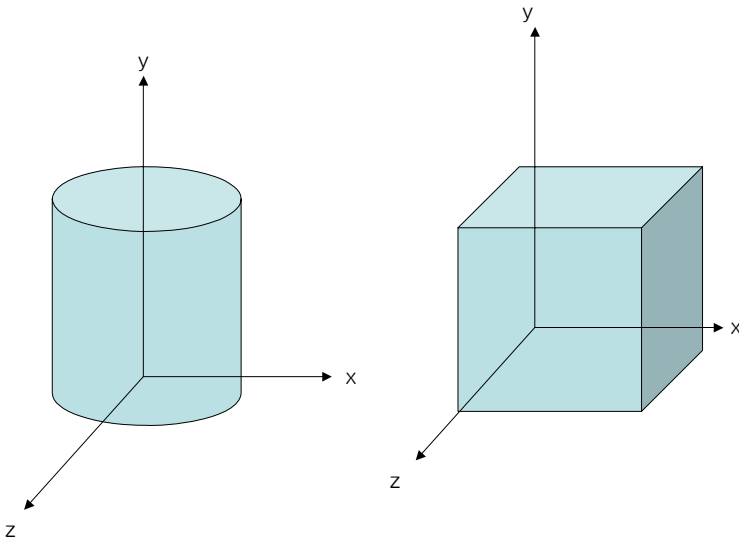
Example


```
#include <gl/glut.h>   // (or others, depending on the system in use)
#include <math.h>
#include <iostream.h>

void init (void)
{
        glClearColor (0.0, 0.0, 0.0, 0.0) ;
        glShadeModel(GL_FLAT);
}

void display(void)
{
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0, 1.0, 1.0);
        glLoadIdentity();
        gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
        glScalef(1.0, 2.0, 1.0);
        glutWireCube(1.0);
        glFlush();
}
```

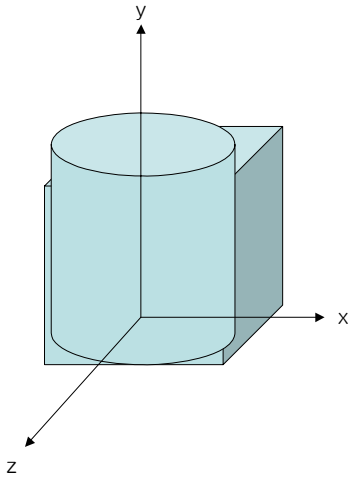
Example


```
void reshape(int w, int h)
{
        glViewport(0, 0, (GLsizei) w, (GLsizei) h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glFrustum(-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
        glMatrixMode (GL_MODELVIEW);
}

void main (int argc, char** argv)
{
        glutInit (&argc, argv);
        glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
        glutInitWindowPosition (100, 100);
        glutInitWindowSize (500, 500);
        glutCreateWindow (argv[0]);
        init();
        glutDisplayFunc (display);
        glutReshapeFunc (reshape);
        glutMainLoop();
}
```

지역 좌표계

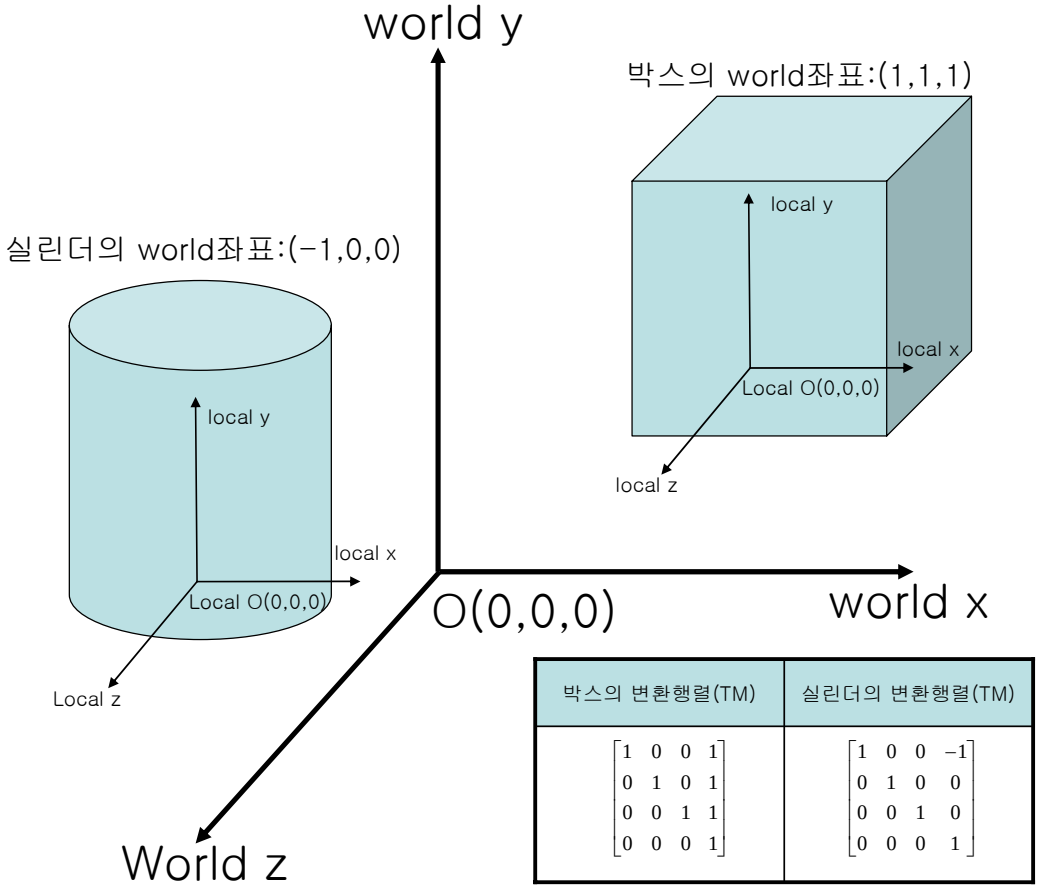


실린더와 박스



원점을 공유한 두 개의 물체

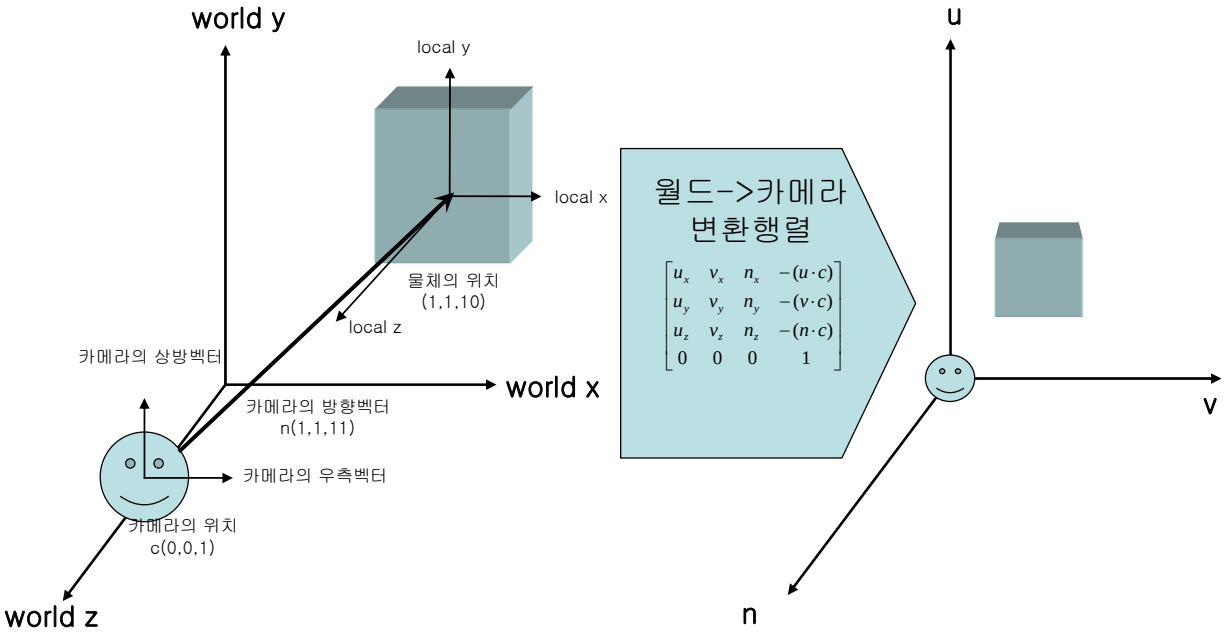
월드 변환



박스의 변환행렬(TM)	실린더의 변환행렬(TM)
$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

월드변환을 적용해서 그린 경우

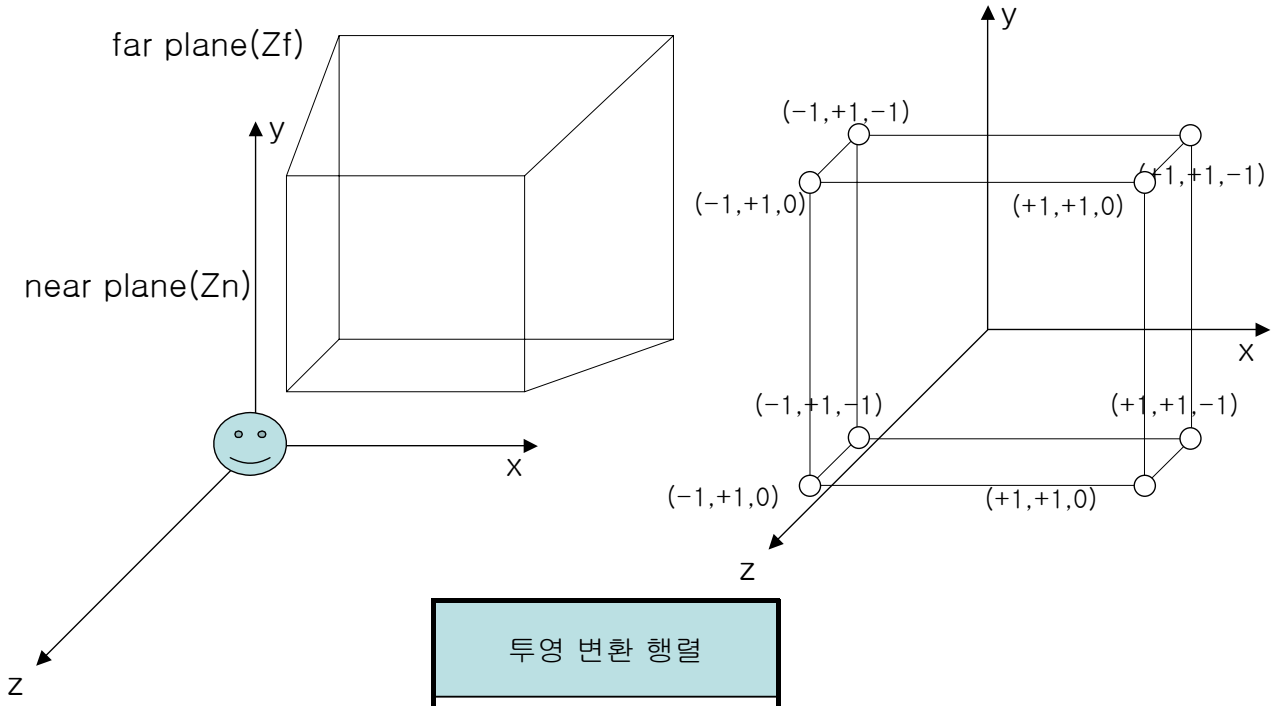
카메라 변환



월드좌표계 카메라좌표계

월드 -> 카메라 좌표계 변환

투영 변환



카메라 좌표계

투영 변환 행렬

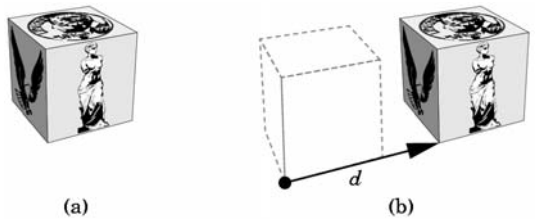
$$\begin{bmatrix} \frac{2\times Z_n}{S_w} & 0 & 0 & 0 \\ 0 & \frac{2\times Z_n}{S_h} & 0 & 0 \\ 0 & 0 & \frac{Z_f}{Z_f - Z_f} & \frac{-Z_f \times Z_n}{Z_f \times Z_n} \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

투영 좌표계

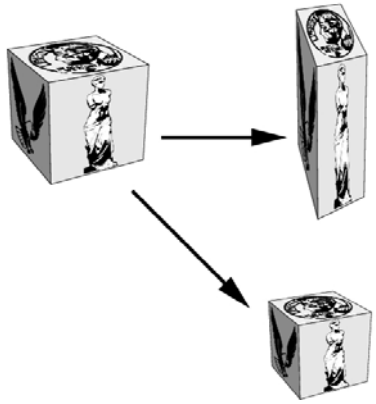
카메라 -> 투영 좌표계 변환

- 변환
  - 새로운 위치로 바꾸는 작업
    - 뷰잉 변환 : 카메라를 설치하는 작업
    - 투영 변환 : **3**차원 좌표를 **2**차원 좌표로 바꾸는 작업
  - 기본 변환의 조합
    - 이동변환, 크기변환, 회전변환
  - 행렬로 표현
    - **3**차원 : **4 x 4** 행렬 **(Why 4x4 ?)**
    - **2**차원 : **3 x 3** 행렬
  - **OpenGL**에서 사용되는 행렬 모드
    - 종류 : **GL_MODELVIEW, GL_PROJECTION**
    - 모드 지정
      - **glMatrixMode(GL_MODELVIEW);**
      - **glMatrixMode(GL_PROJECTION);**

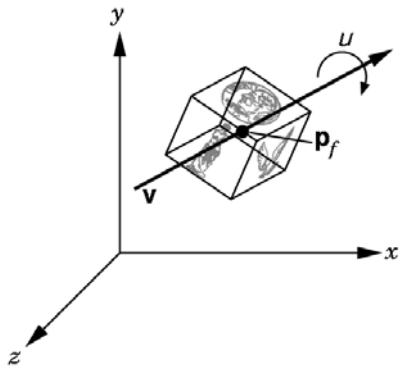
- 이동변환 : **glTranslatef(dx, dy, dz);**



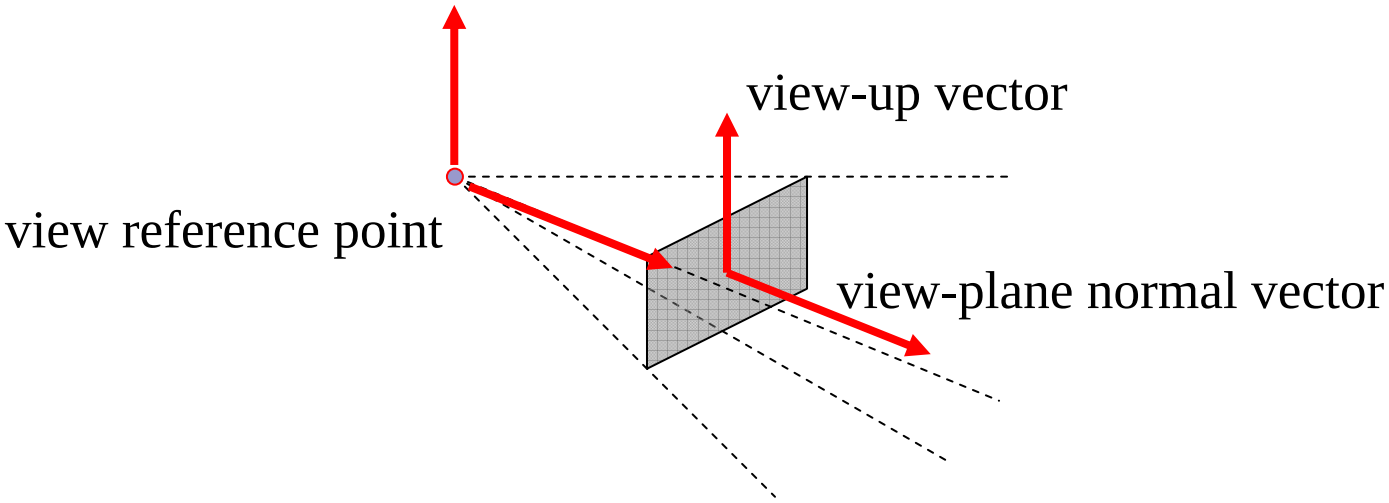
- 크기변환 : **glScalef(sx, sy, sz);**



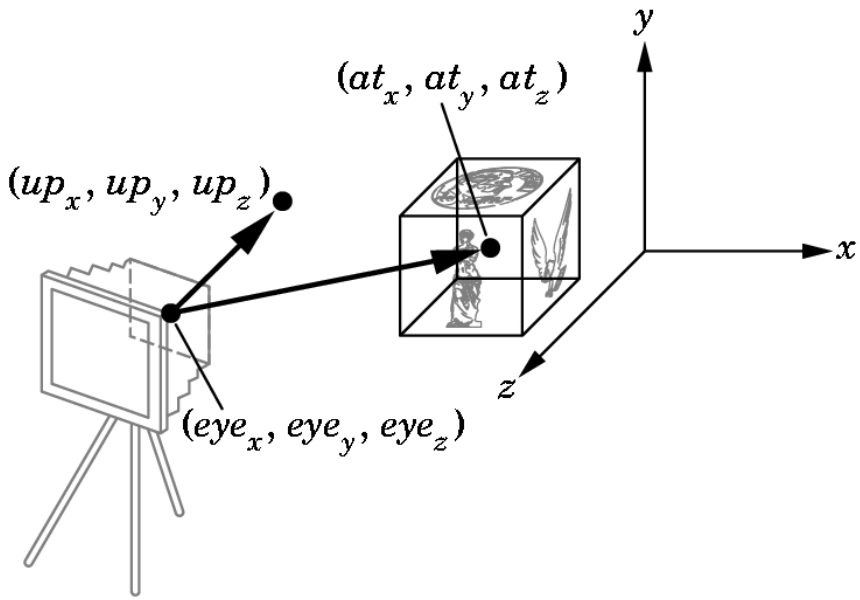
- 회전변환 : **glRotatef(θ , rx, ry, rz);**



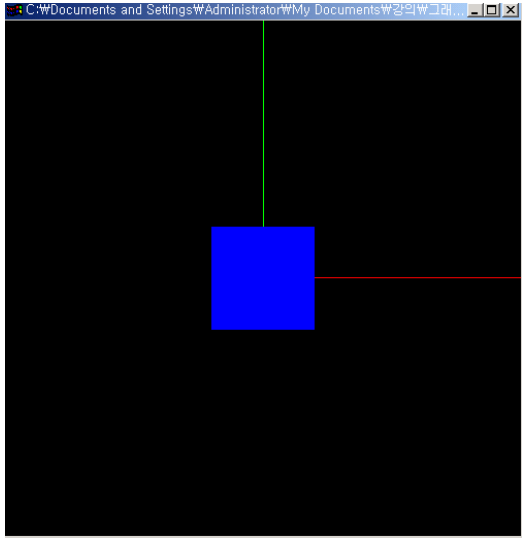
- 뷰잉 변환
  - 카메라를 설치하는 작업
    - 카메라 위치 : VRP (view reference point)
    - 카메라 방향 : VPN (view plane normal)
    - 카메라 앵글 방향 : VUP (view up vector)

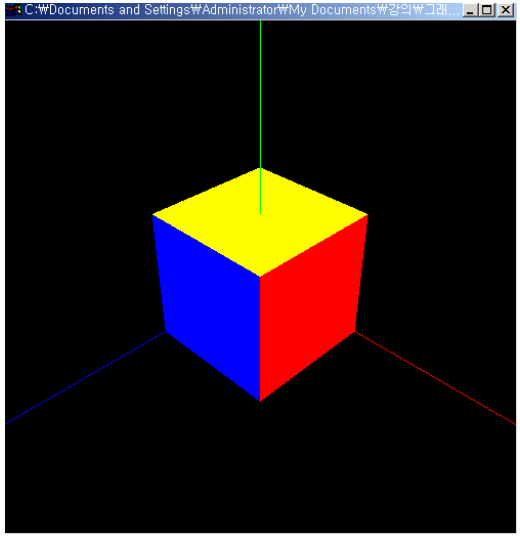


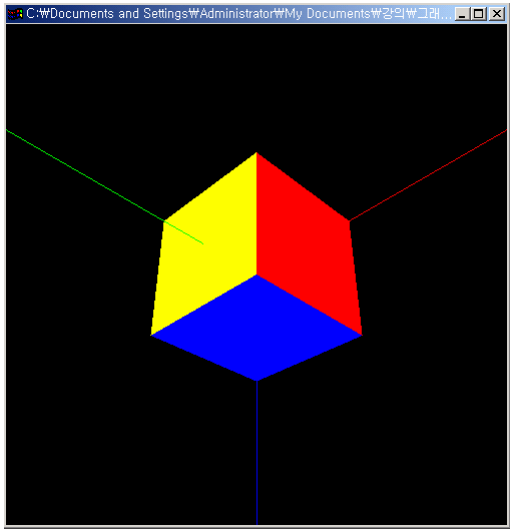
- 뷰잉 변환
  - **OpenGL 함수 : Look-At Function**
    - gluLookAt(eyeX, eyeY, eyeZ, atX, atY, atZ, upX-eyeX, upY-eyeY, upZ-eyeZ);
    - 예) gluLookAt(0, 0, 5, 0, 0, 0, 0, 1, 0);



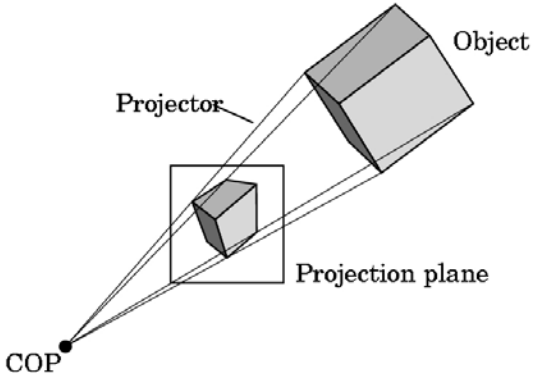
- 예제 **1 :** gluLookAt(0.0, 0.0, 13.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
- 예제 **2 :** gluLookAt(5.0, 5.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
- 예제 **3 :** gluLookAt(5.0, 5.0, 5.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0);

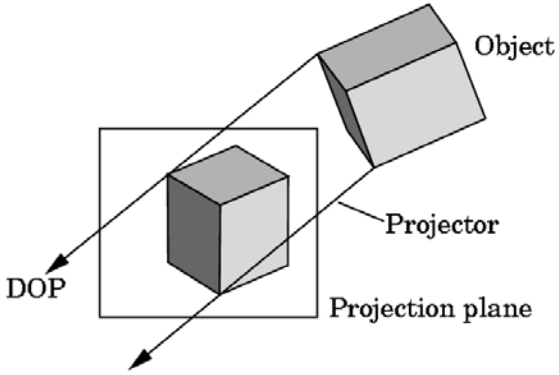






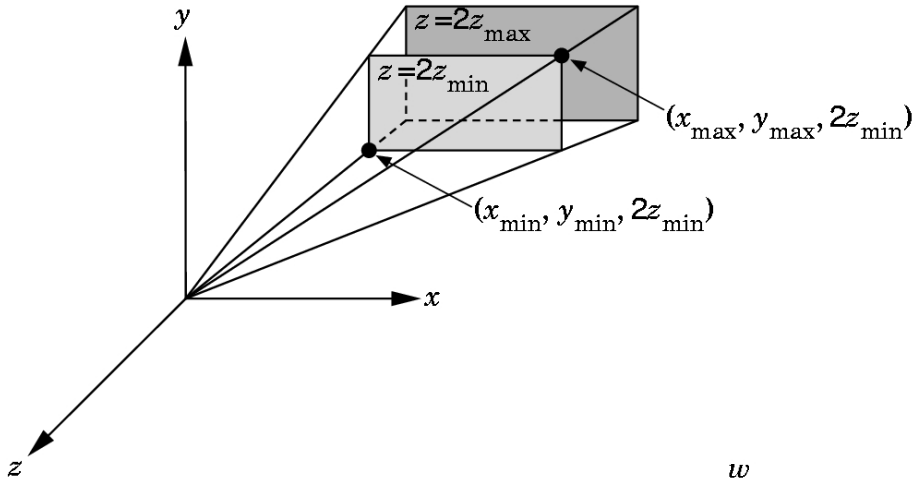
- 카메라의 렌즈 선택
  - 관측 공간 (Viewing Volume) 결정
    - 원근 투영 (Perspective Viewing)
      - finite COP
      - center of projection
    - 직교 투영 (Orthographic Viewing)
      - finite COP
      - center of projection



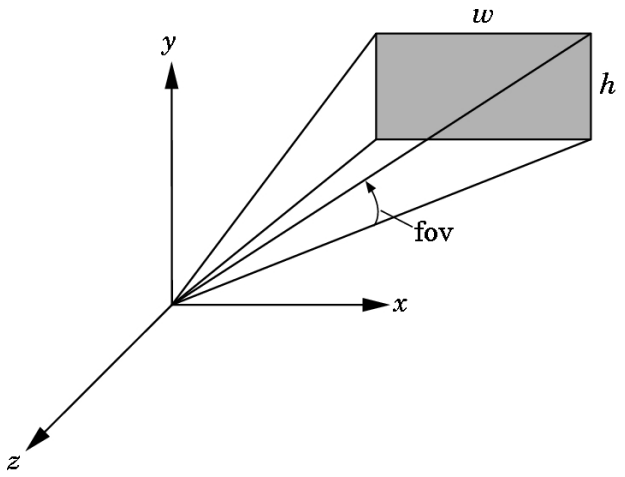


Perspective Projection

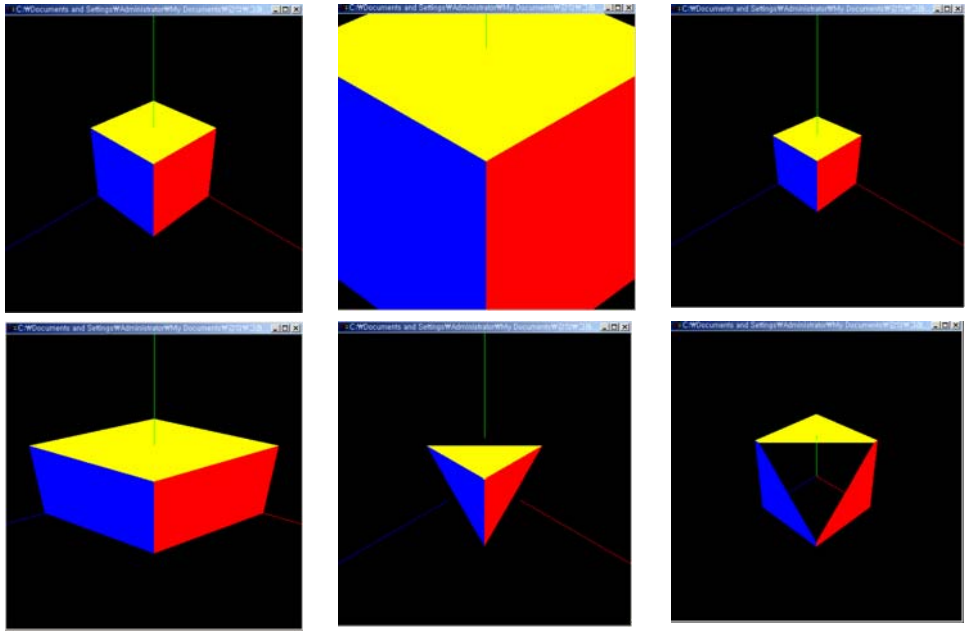
- glFrustum(xmin, xmax, ymin, ymax, znear, zfar);
  - znear, zfar > 0
  - zmin = –znear, zmax = –zfar



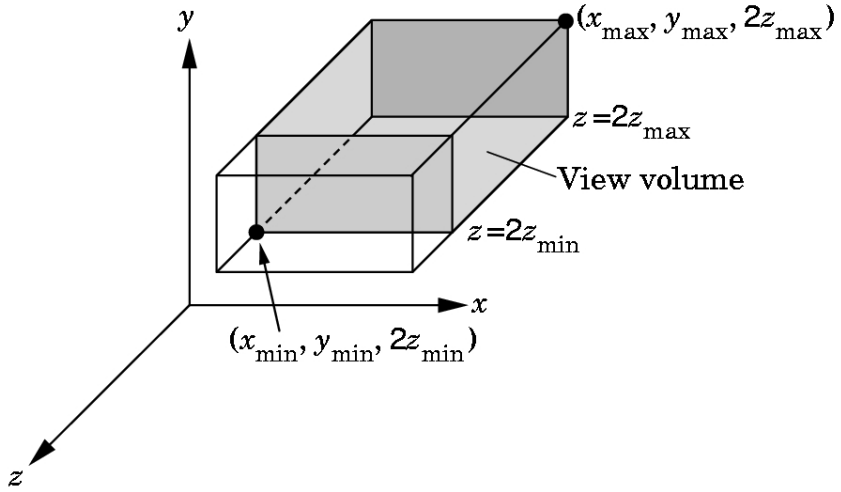
- gluPerspective(fovy, aspect, znear, zfar)
  - fovy : field of view in y-direction, in degree
  - aspect : width / height



- 카메라 위치 : gluLookAt(5.0, 5.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
  - 예제 **1 :** gluPerspective(45.0, 1.0, 1, 30)
  - 예제 **2 :** gluPerspective(15.0, 1.0, 1, 30)
  - 예제 **3 :** gluPerspective(60.0, 1.0, 1, 30)
  - 예제 **4 :** gluPerspective(45.0, 0.5, 1, 30)
  - 예제 **5 :** gluPerspective(45.0, 1.0, 1, 8)
  - 예제 **6 :** gluPerspective(45.0, 1.0, 8, 30)

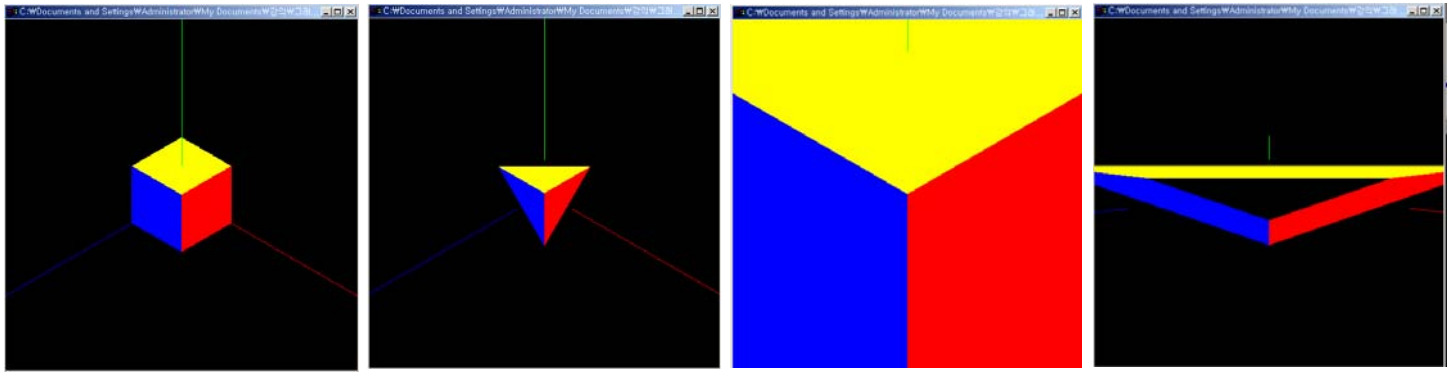


- **Orthographic Projection**
  - glOrtho(xmin, xmax, ymin, ymax, zmin, zmax);
  - gluOrtho2D(xmin, xmax, ymin, ymax);
    - glOrtho(xmin, xmax, ymin, ymax, -1, 1);

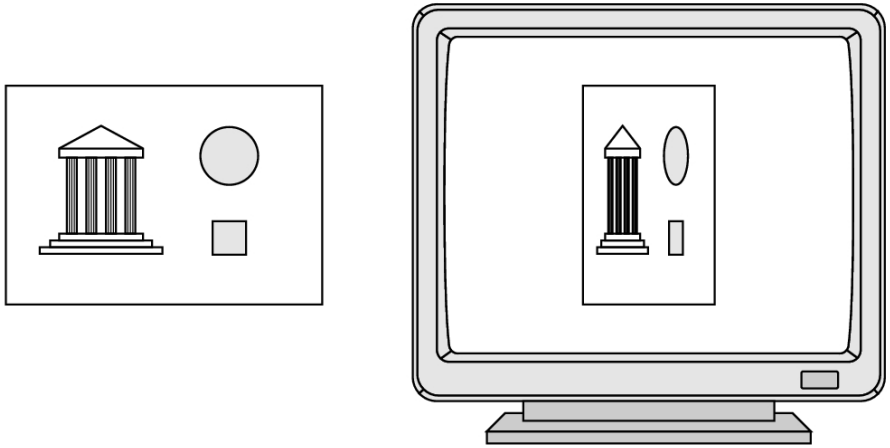


- 카메라 위치 : gluLookAt(5.0, 5.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

  - **예제 1 :** glOrtho(-5.0, 5.0, -5.0, 5.0, -5.0, 25.0)
  - **예제 2 :** glOrtho(-5.0, 5.0, -5.0, 5.0, 1.0, 8.0)
  - **예제 3 :** glOrtho(-1.0, 1.0, -1.0, 1.0, 1.0, 30.0)
  - **예제 4 :** glOrtho(-1.0, 1.0, -5.0, 5.0, 7.0, 8.0)



- **Viewport 설정**
  - void glViewport(GLint x, y, GLsizei w, h);
    - (x, y) 위치에서, w × h 영역에 출력

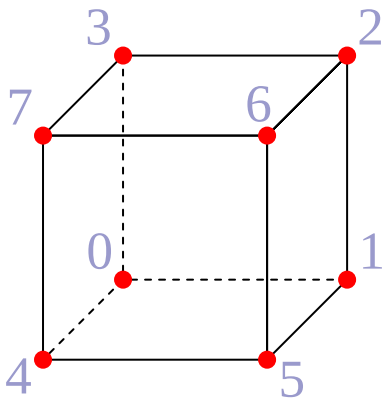


(a) (b)

Modeling a Colored Cube

- Modeling a Cube
  - vertex 정의

```
GLfloat vertices[8][3] = {
        { –1.0, –1.0, –1.0 },  // 0
        { 1.0, –1.0, –1.0 },   // 1
        { 1.0, 1.0, –1.0 },    // 2
        { –1.0, 1.0, –1.0 },   // 3
        { –1.0, –1.0, 1.0 },   // 4
        { 1.0, –1.0, 1.0 },    // 5
        { 1.0, 1.0, 1.0 },     // 6
        { –1.0, 1.0, 1.0 }     // 7
};
```

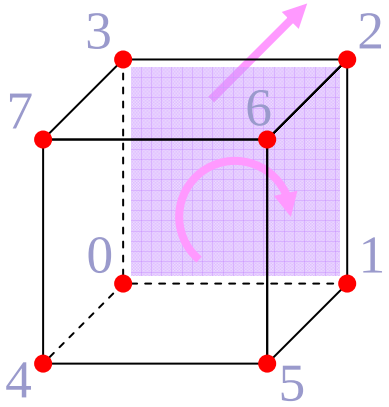


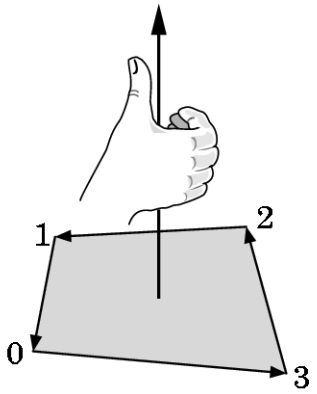
Modeling a Colored Cube

- Modeling a Cube
  - face 정의

```
glBegin(GL_POLYGON);
        glVertex3fv(vertices[0]);
        glVertex3fv(vertices[3]);
        glVertex3fv(vertices[2]);
        glVertex3fv(vertices[1]);
glEnd( );
```

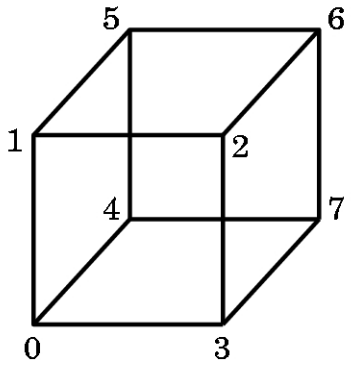
  - outward facing
    - normal vector 는 항상 외부를 향한다.
    - right-hand rule

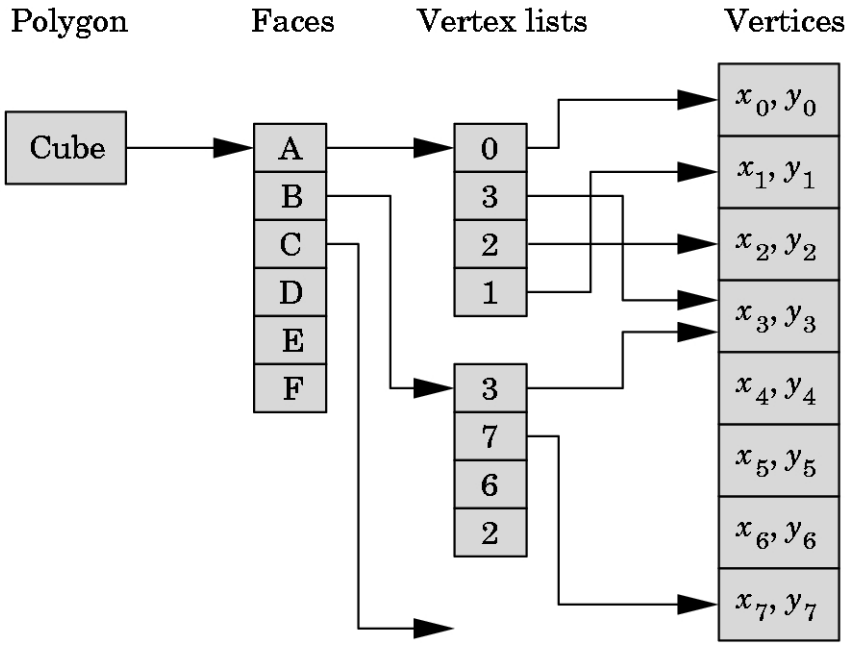




Data Structure

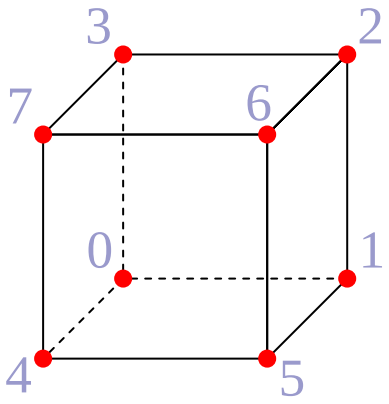
- vertex-list representation
  - 1 개의 vertex를 3개의 face가 공유 가능
  - list 구조 : index 로 vertex 를 share

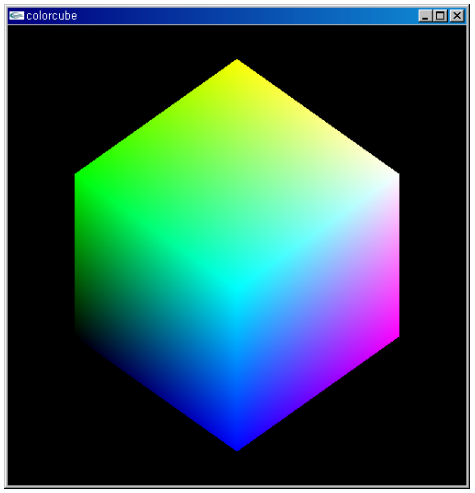




- Color Cube
  - vertex 마다 color 설정

```
GLfloat colors[8][3] = {
        { 0.0, 0.0, 0.0 },  // black
        { 1.0, 0.0, 0.0 },  // red
        { 1.0, 1.0, 0.0 },  // yellow
        { 0.0, 1.0, 0.0 },  // green
        { 0.0, 0.0, 1.0 },  // blue
        { 1.0, 0.0, 1.0 },  // magenta
        { 1.0, 1.0, 1.0 },  // white
        { 0.0, 1.0, 1.0 }   // cyan
};
```





Modeling a Colored Cube

- Face 출력 함수

```
void  quad(int a, int b, int c , int d) {
        glBegin(GL_POLYGON);
                glColor3fv(colors[a]);
                glVertex3fv(vertices[a]);
                glColor3fv(colors[b]);
                glVertex3fv(vertices[b]);
                glColor3fv(colors[c]);
                glVertex3fv(vertices[c]);
                glColor3fv(colors[d]);
                glVertex3fv(vertices[d]);
        glEnd();
}
```

```
/* face 별로 출력 */

void colorcube(void) {
   quad(0,3,2,1);
   quad(2,3,7,6);
   quad(0,4,7,3);
   quad(1,2,6,5);
   quad(4,5,6,7);
   quad(0,1,5,4);
}
```

Example


```
#include <gl/glut.h>
    GLfloat vertices[8][3] = {
        { –1.0, –1.0, –1.0 },      // 0
        { 1.0, –1.0, –1.0 },       // 1
        { 1.0, 1.0, –1.0 },        // 2
        { –1.0, 1.0, –1.0 },       // 3
        { –1.0, –1.0, 1.0 },       // 4
        { 1.0, –1.0, 1.0 },        // 5
        { 1.0, 1.0, 1.0 },         // 6
        { –1.0, 1.0, 1.0 }         // 7
    };

    GLfloat colors[8][3] = {
        { 0.0, 0.0, 0.0 },  // black
        { 1.0, 0.0, 0.0 },  // red
        { 1.0, 1.0, 0.0 },  // yellow
        { 0.0, 1.0, 0.0 },  // green
        { 0.0, 0.0, 1.0 },  // blue
        { 1.0, 0.0, 1.0 },  // magenta
        { 1.0, 1.0, 1.0 },  // white
        { 0.0, 1.0, 1.0 }   // cyan
    };
```

Example


```
void init (void)
{
        glClearColor (0.0, 0.0, 0.0, 0.0) ;
        glShadeModel(GL_FLAT);
}
void  quad(int a, int b, int c , int d) {
                glBegin(GL_POLYGON);
                        glColor3fv(colors[a]);
                        glVertex3fv(vertices[a]);
                        glColor3fv(colors[b]);
                        glVertex3fv(vertices[b]);
                        glColor3fv(colors[c]);
                        glVertex3fv(vertices[c]);
                        glColor3fv(colors[d]);
                        glVertex3fv(vertices[d]);
                glEnd();
}
```

Example


```
void colorcube(void) {
  quad(0,3,2,1);
  quad(2,3,7,6);
  quad(0,4,7,3);
  quad(1,2,6,5);
  quad(4,5,6,7);
  quad(0,1,5,4);
}

void display(void)
{
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0, 1.0, 1.0);
        glLoadIdentity();
        gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
        glScalef(1.0, 2.0, 1.0);
        colorcube();
        glFlush();
}
```

Example


```
void reshape(int w, int h)
{
        glViewport(0, 0, (GLsizei) w, (GLsizei) h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glFrustum(-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
        glMatrixMode (GL_MODELVIEW);
}

void main (int argc, char** argv)
{
        glutInit (&argc, argv);
        glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
        glutInitWindowPosition (100, 100);
        glutInitWindowSize (500, 500);
        glutCreateWindow (argv[0]);
        init();
        glutDisplayFunc (display);
        glutReshapeFunc (reshape);
        glutMainLoop();
}
```

Example

- 문제점
  - 육면체가 제대로 그려지지 않는다.
    - 그리는 순서대로 그려지기 때문
      ➔사각형 그리는 함수 Quad의 순서를 바꾸어 볼 것

      ➔ Z-buffer 를 사용하여야 함
      glEnable(GL_DEPTH_TEST);
      glClear(GL_DEPTH_BUFFER_BIT);
      glutInitDisplayMode (GLUT_DEPTH);
  - 색상이 예제 결과와 다르다.
    - 한 면에 하나의 색만 나타난다.
      ➔Shading 방법을 바꾸어 볼 것

      ➔glShadeModel(GL_SMOOTH);

Matrix Direct Control Commands

- 변환 행렬 직접 지정
  - **glMatrixMode(Glenum mode)**
    - 모델뷰, 투영, 텍스쳐 행렬을 수정할 지 여부 지정
    - GL_MODELVIEW, GL_PROJECTION, GL_TEXTURE
  - **glLoadIdentity(void)**
    - 현재수정 가능한 행렬을 4x4 단위 행렬로 설정
  - **glLoadMatrix{fd}(const TYPE *m)**
    - 현재 행렬의 16개의 값을 m 으로 지정된 행렬의 값으로 설정
  - **glMultMatrix{fd}(const TYPE *m)**
    - 행렬 m을 현재 행렬과 곱하고, 그 결과를 현재 행렬로 둔다.

$$m = \begin{pmatrix} m_1 & m_5 & m_9 & m_{13} \\ m_2 & m_6 & m_{10} & m_{14} \\ m_3 & m_7 & m_{11} & m_{15} \\ m_4 & m_8 & m_{12} & m_{16} \end{pmatrix} \qquad m = \begin{pmatrix} m_{11} & m_{21} & m_{31} & m_{41} \\ m_{12} & m_{22} & m_{32} & m_{42} \\ m_{13} & m_{23} & m_{33} & m_{43} \\ m_{14} & m_{24} & m_{34} & m_{44} \end{pmatrix}$$

변환 순서 과 코드 순서

- Code
  - glMatrixMode(GL_MODELVIEW);
  - glLoadIdentity();
  - glMultMatrixf(N); // 변환 N 을 적용
  - glMultMatrixf(M); // 변환 M 을 적용
  - glMultMatrixf(L); // 변환 L 을 적용
  - glBegin(GL_POINTS);
    - glVertex3f(x, y, z);
  - glEnd();

- 모델뷰 행렬
  - I → N → NM → NML
- 결과
  - N(M(L(v)))
- 변환순서
  - 변환 L → 변환 M → 변환 N 적용

➔ 변환 순서와 코드 순서는 바뀐다.

- glTranslatef(4.0, 0.0, 0.0);
- glRotatef(45.0, 0.0, 0.0, 1.0);
- colorcube();

- glRotatef(45.0, 0.0, 0.0, 1.0);
- glTranslatef(4.0, 0.0, 0.0);
- colorcube();