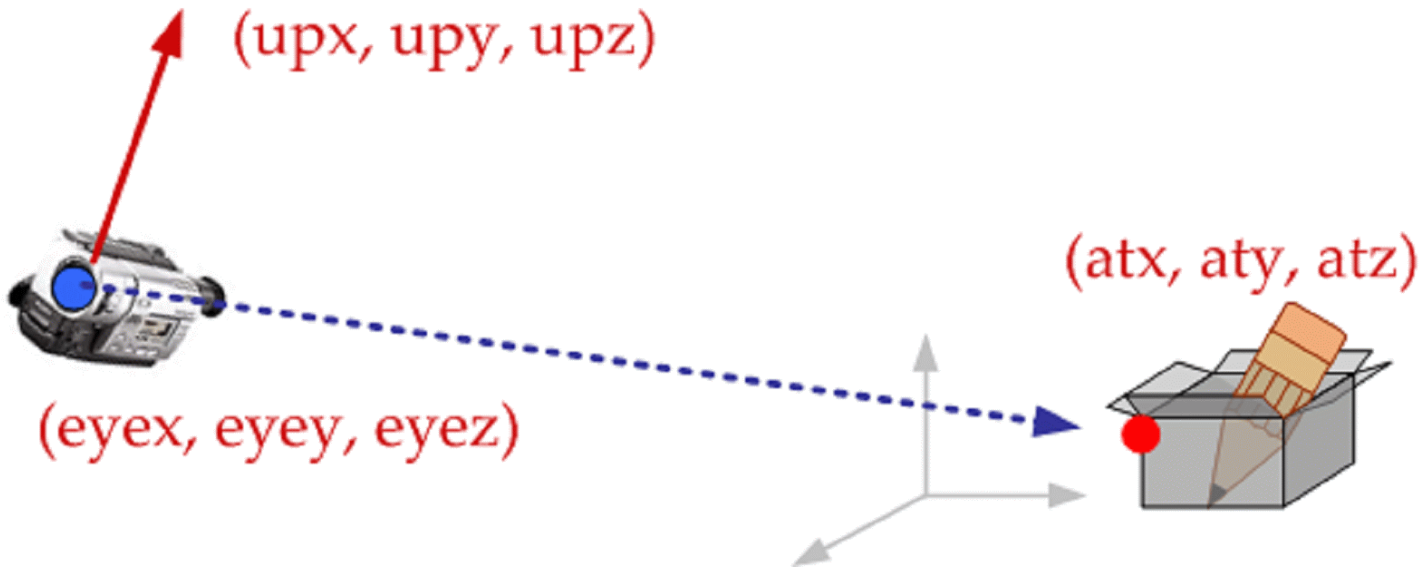
CGL
COMPUTER GRAPHICS LABORATORY

컴퓨터 그래픽스
[05]

2023학년도 1학기

담당교수 : 마 준

- Viewing & Modeling Transformation
- Projection Transformation
- Viewport Transformation

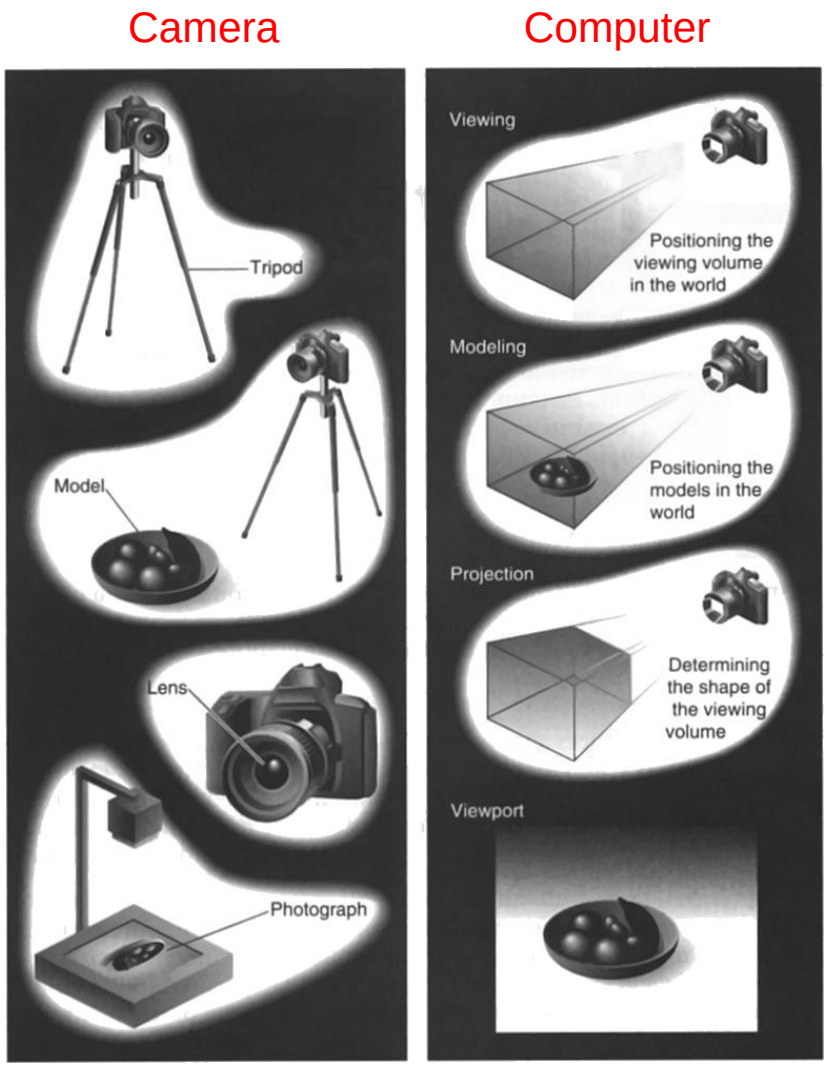


카메라와 OpenGL

- OpenGL을 이용한 렌더링 절차는 사진기를 이용한 작업과 유사함

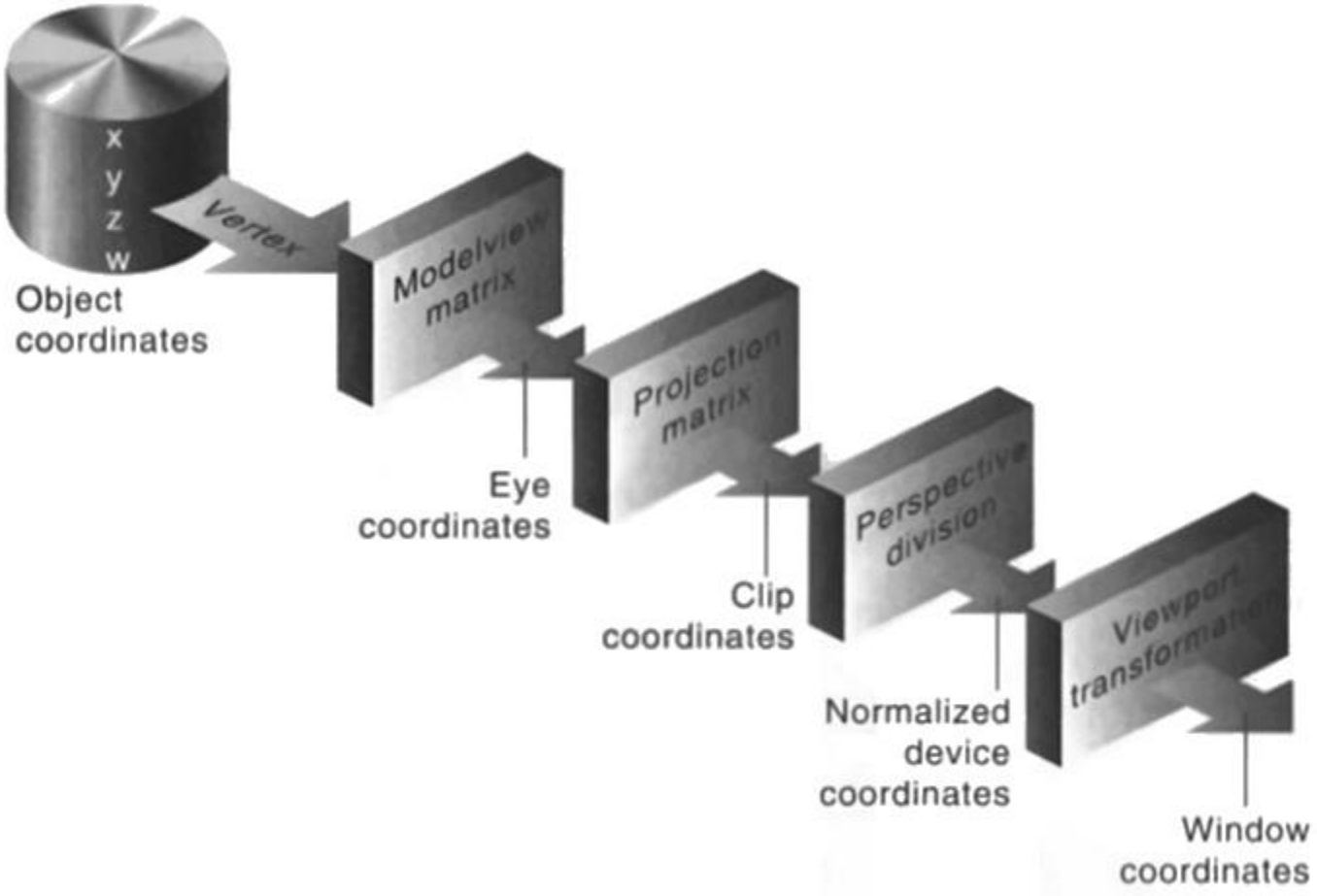
- 카메라 설치
- 방향 조정
- **촬영**
- 결과물 확인

- 카메라 설치
- 방향 조정
- **투영**
- 결과물 확인



OpenGL의 렌더링 파이프라인

- 물체 정보 => 정점 정보 => 모델/뷰 변환 => 조명 처리 => 투영 => 클리핑 => 화면 매핑



OpenGL의 렌더링 파이프라인

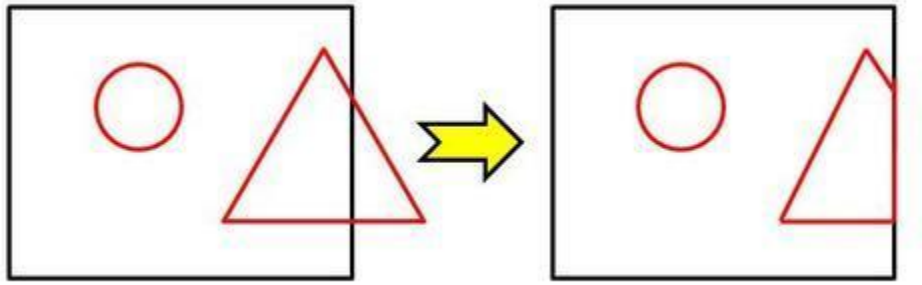
- 모델/뷰 변환 : 모델의 로컬 위치 및 뷰(카메라)의 위치에 따라 화면상에 보여질 시야의 위치를 결정하는 단계

- 조명 처리 : 조명의 위치 설정, 법선 벡터 처리, 재질 속성 처리를 거쳐 물체의 색상을 결정하는 단계

- 투영 : 3차원 상의 공간을 2차원 평면(모니터)에 투영하는 단계

OpenGL의 렌더링 파이프라인

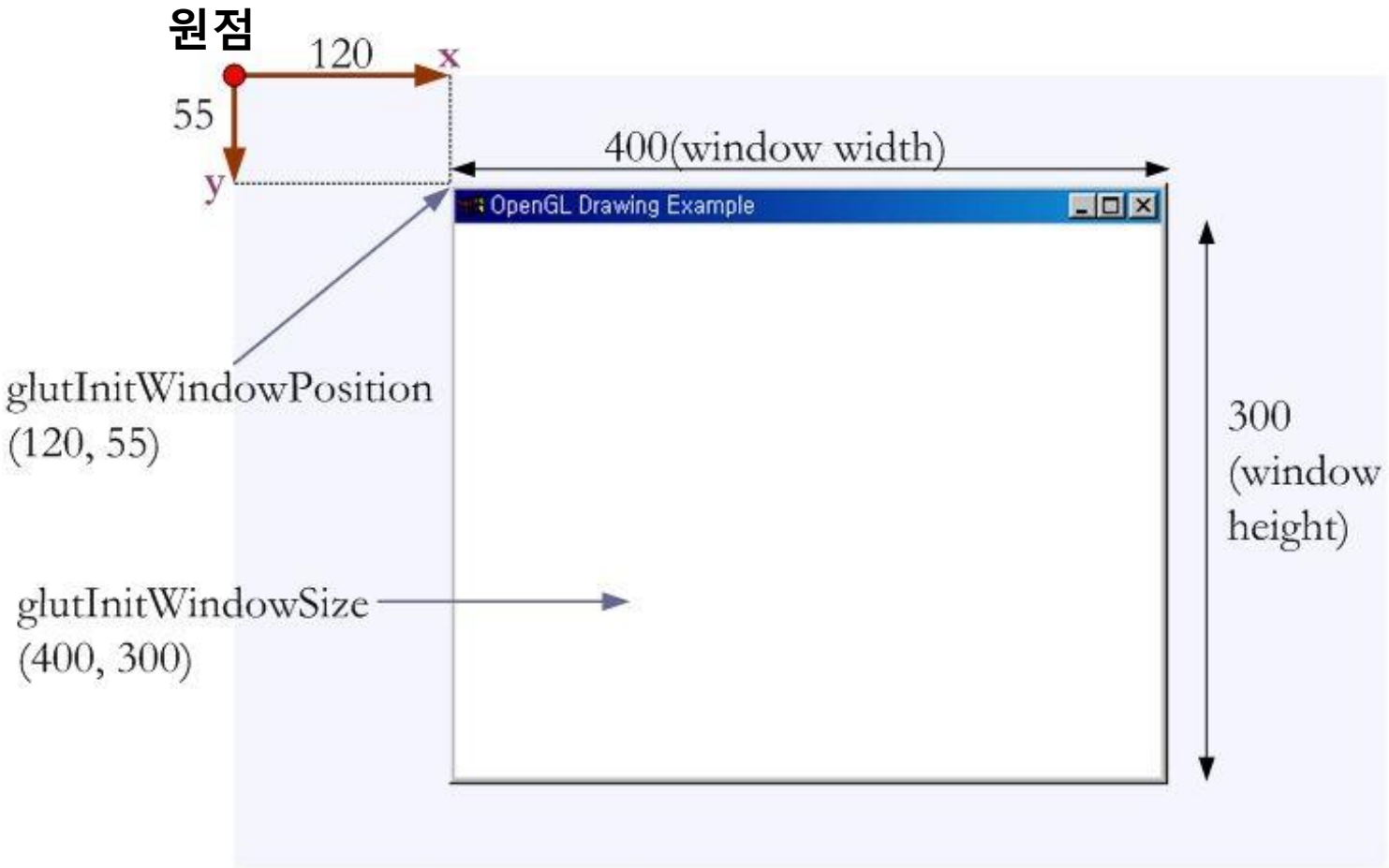
- 클리핑 : 투영 후 2차원 평면의 화면이 결정되면, 시야에서 벗어나는 객체를 잘라내는 단계



- 화면 매핑 : 클리핑된 마지막 요소들을 화면에 매핑하는 단계로 윈도우 창 내부에 화면을 보여주는 단계

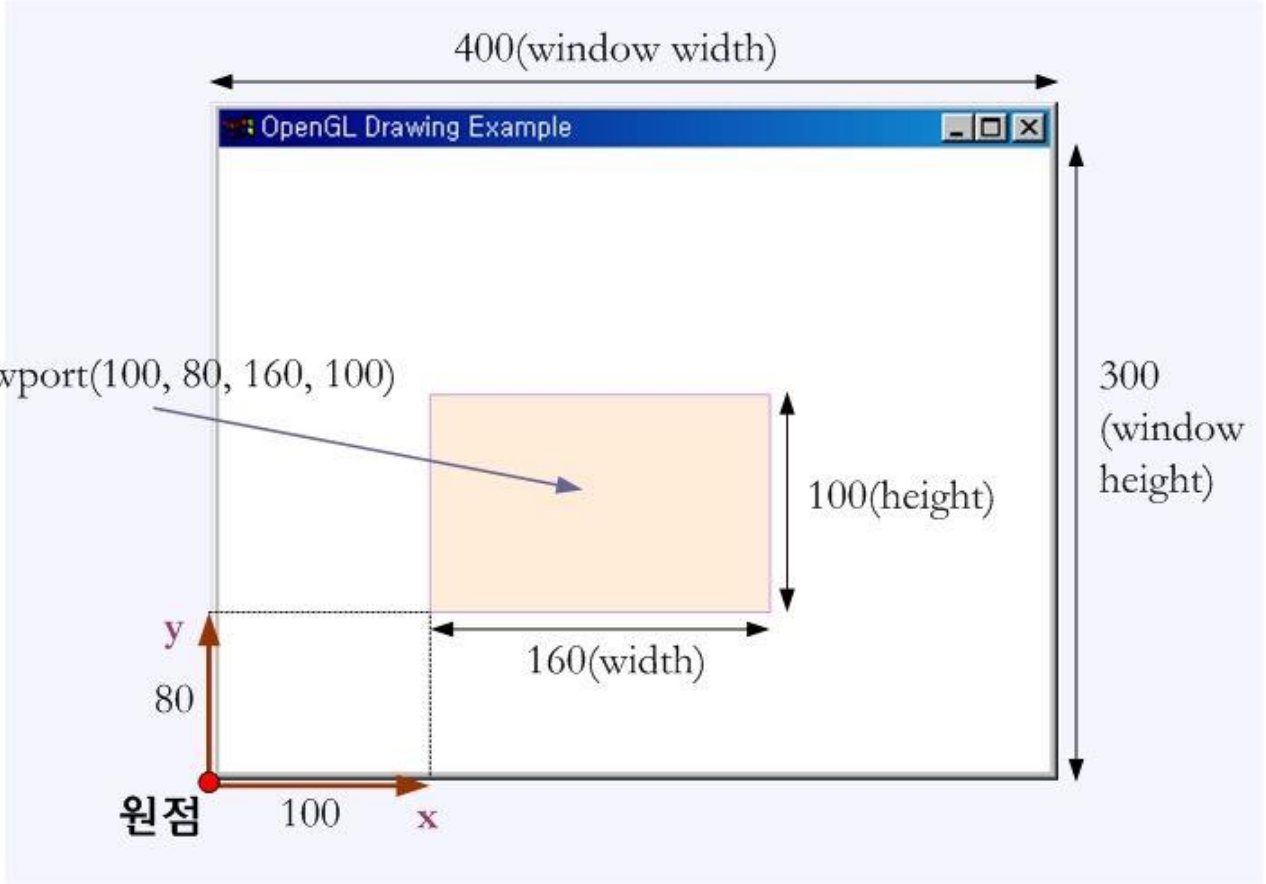
GLUT의 윈도우 제어

- Width = 400, Height = 300인 윈도우를 화면좌표(120, 55)에 위치시키는 경우



OpenGL의 Viewport 설정

- GLUT에서 사용하는 화면 좌표계는 GL이 사용하는 화면 좌표계와 달라서 이벤트처리를 할 때 유의해야함!!



```
void reshape(int new_w, int new_h)
{
    //0,0부터 시작해서 윈도우 전체를 뷰포트로 설정
    glViewport(0, 0, new_w, new_h);

    //0,0부터 시작해서 윈도우의 1사분면을 뷰포트로 설정
    glViewport(0, new_h/2, new_w / 2, new_h / 2);

    //0,0부터 시작해서 윈도우의 2사분면을 뷰포트로 설정
    glViewport(new_w / 2, new_h / 2, new_w / 2, new_h / 2);

    //0,0부터 시작해서 윈도우의 3사분면을 뷰포트로 설정
    glViewport(0, 0, new_w / 2, new_h / 2);

    //0,0부터 시작해서 윈도우의 4사분면을 뷰포트로 설정
    glViewport(new_w / 2, 0, new_w / 2, new_h / 2);

    float WidthFactor = (float)new_w / 250.0;
    float HeightFactor = (float)new_h / 250.0;

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    gluOrtho2D(-5.0 * WidthFactor, 5.0 * WidthFactor, -5.0 * HeightFactor, 5.0 * HeightFactor);
}
```

03_2 Viewport 응용(여러 개의 물체 그리기)

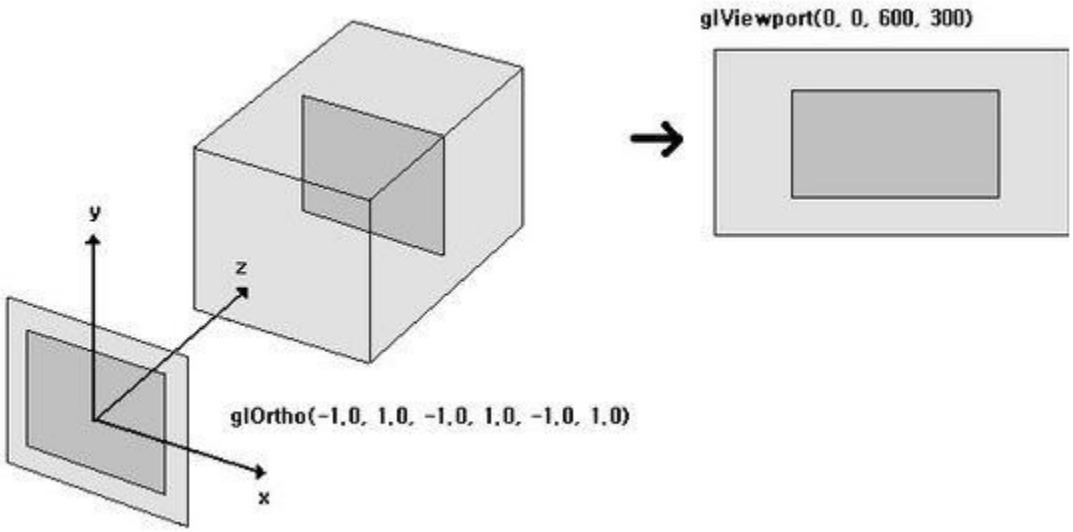
CGL
COMPUTER GRAPHICS LABORATORY

```
#include <gl\glut.h>
#include <stdio.h>
#include <stdlib.h>

float angle = 0.0f;

int width;
int height;

void idleProcess()
{

    angle += 0.0001;
    if (angle > 360.0) angle = 0.0f;

    glutPostRedisplay();
}

void draw_box(float R, float G, float B)
{
    glRotatef(angle, 0, 0, 1);
    glColor3f(R, G, B);
    glBegin(GL_POLYGON);
    glVertex3f(-0.75, -0.75, 0.0);
    glVertex3f(0.75, -0.75, 0.0);
    glVertex3f(0.75, 0.75, 0.0);
    glVertex3f(-0.75, 0.75, 0.0);
    glEnd();
}
```

```
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    //0,0부터 시작해서 윈도우의 1사분면을 뷰포트로 설정
    glViewport(0, height / 2, width / 2, height / 2);

    draw_box(1.0f, 1.0f, 1.0f);

    //0,0부터 시작해서 윈도우의 2사분면을 뷰포트로 설정
    glViewport(width / 2, height / 2, width / 2, height / 2);

    draw_box(1.0f, 0.0f, 0.0f);

    //0,0부터 시작해서 윈도우의 3사분면을 뷰포트로 설정
    glViewport(0, 0, width / 2, height / 2);

    draw_box(0.0f, 1.0f, 0.0f);

    //0,0부터 시작해서 윈도우의 4사분면을 뷰포트로 설정
    glViewport(width / 2, 0, width / 2, height / 2);

    draw_box(0.0f, 0.0f, 1.0f);

    glutSwapBuffers();
}
```

```
void reshape(int new_w, int new_h)
{
    width = new_w;
    height = new_h;

    float WidthFactor = (float)new_w / 250.0;
    float HeightFactor = (float)new_h / 250.0;

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    gluOrtho2D(-5.0 * WidthFactor, 5.0 * WidthFactor, -5.0 * HeightFactor, 5.0 * HeightFactor);
}
```

- Void glViewport(int x, int y, int width, int height)
  - 창 내에서 OpenGL의 렌더링이 진행될 영역 설정
  - x, y는 렌더링 영역의 시작 위치
  - width, height는 렌더링 영역의 넓이와 높이



03_3 3차원 큐브

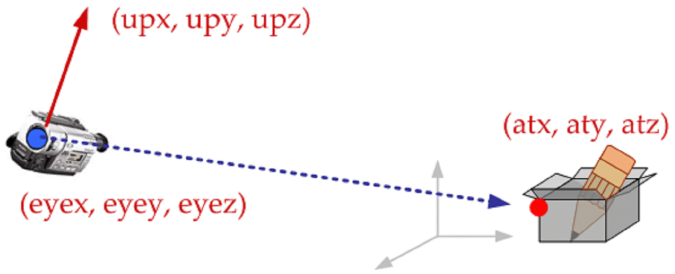
CGL
COMPUTER GRAPHICS LABORATORY

```
#include <gl\glut.h>
#include <stdio.h>
#include <stdlib.h>

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0f, 1.0f, 1.0f);
    glLoadIdentity();   // 모델뷰 행렬 초기화

    // Viewing Transformation
    gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

    // Modeling Transformation
    glScalef(1.0, 2.0, 1.0);
    glutWireCube(1.0);
    glutSwapBuffers();
}
void init()
{
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT);
}

void reshape(int new_w, int new_h)
{
    glViewport(0, 0, new_w, new_h);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();   // 투영 행렬 초기화

    glFrustum(-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
    glMatrixMode(GL_MODELVIEW);
}
```

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);

    glutInitWindowSize(500, 500);
    glutInitWindowPosition(100, 100);

    glutCreateWindow("03_3 3차원 큐브");
    init();

    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();

    return 0;
}
```

- void glLoadIdentity(void);
  - 행렬을 단위행렬로 초기화해주는 함수
  - 이전의 변환에 영향을 받지 않도록 설정함

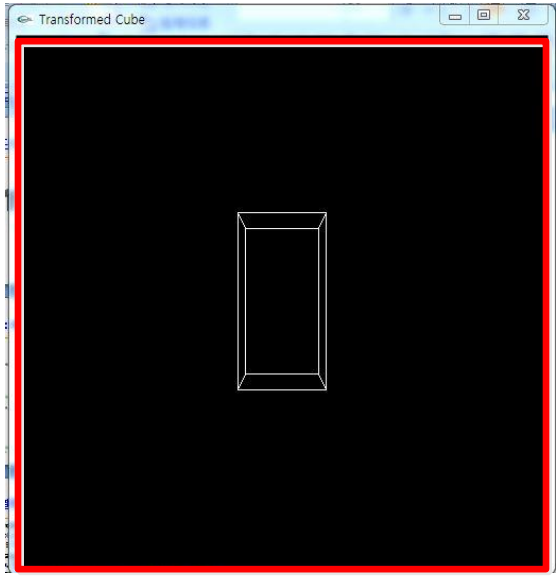
- Void gluLookAt(double eyex, double eyey, double eyez, double centerx, double centery, double centerz, double upx, double upy, double upz);
  - 카메라(눈)의 위치를 설정하는 함수
  - eye(x,y,z)는 카메라(눈)의 위치를 지정
  - center(x,y,z)는 카메라(눈)가 바라보는 곳을 지정
  - up(x,y,z)는 카메라(눈)의 기울임 정도를 지정

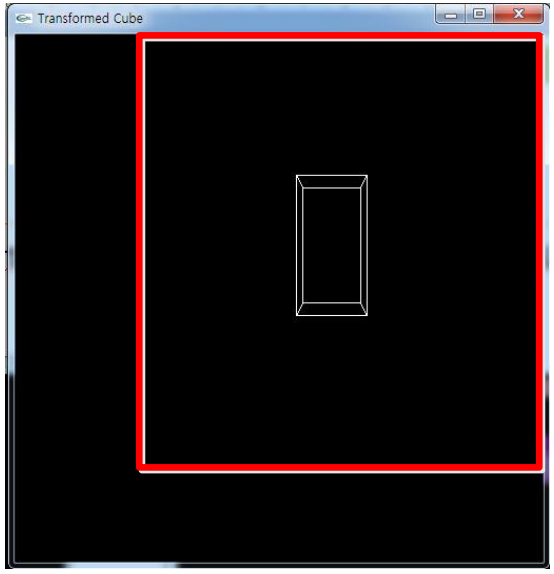


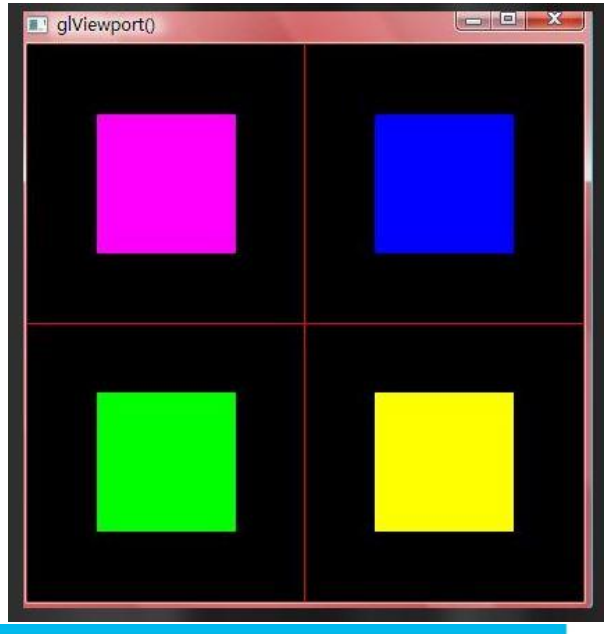
주요 함수 설명

- void glScalef(float x, float y, float z);
  - 물체의 크기를 변환하는 함수
  - x, y, z 각 인자를 이용하여 1.0을 기준으로 확대/축소 가능

- void glutWireCube(double size);
  - 와이어 형태의 큐브를 그려주는 함수
  - size 인자는 큐브의 크기를 지정(1.0이 기본)

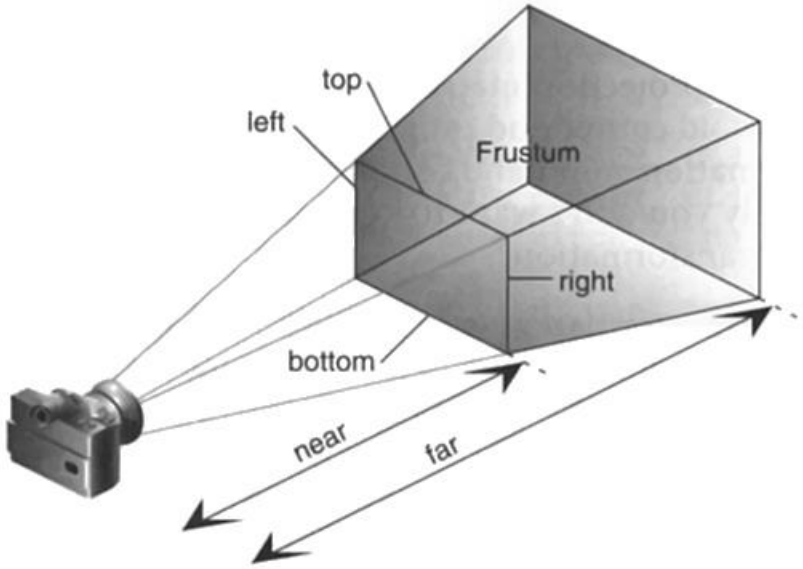






glViewport(0, 0, w, h); glViewport(100, 100, 400, 400);

- void glFrustum(double left, double right, double bottom, double top, double near, double far);
  - 원근 투영을 위한 원근 행렬을 생성해주는 함수
  - left, right : 절두체 앞쪽의 좌/우(너비) 위치를 지정
  - bottom, top : 절두체 앞쪽의 상/하(높이) 위치를 지정
  - near, far : 절두체 앞, 뒤쪽의 클리핑 플레인 간 거리를 지정

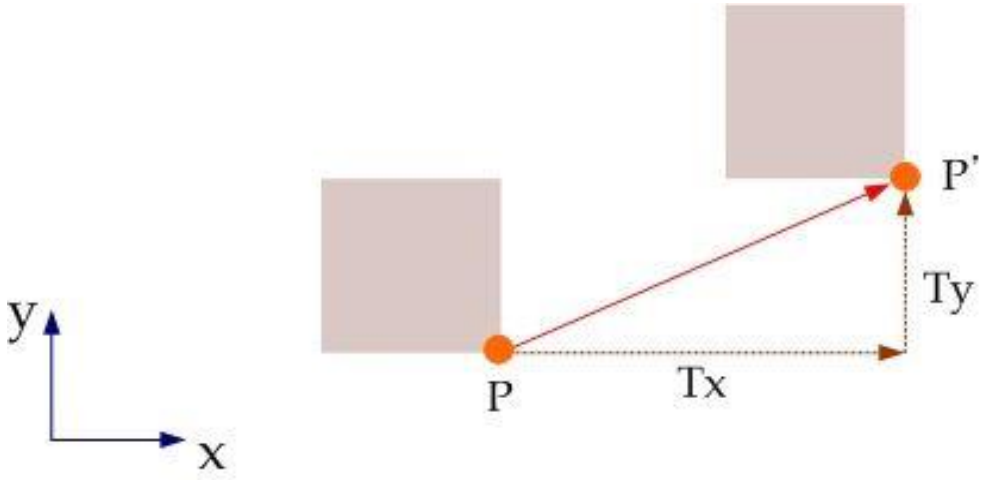


OpenGL Transformation

CGL
COMPUTER GRAPHICS LABORATORY

- Viewing 변환
  - gluLookAt() 함수를 이용하여 카메라(시점)의 위치를 설정
- Modeling 변환
  - 이동(translate), 회전(rotate), 크기(scale) 지정을 통해 물체의 위치 및 크기 조정을 지정
  - glTranslatef(), glRotatef(), glScalef()
- Projection 변환
  - 투영 방법(직교, 원근)을 설정하는 함수
  - glFrusturm(), gluPerspective(), glOrtho(), gluOrtho2D()
- Viewport 변환
  - 이미지가 그려질 윈도우의 크기를 설정하는 함수
  - glViewport()

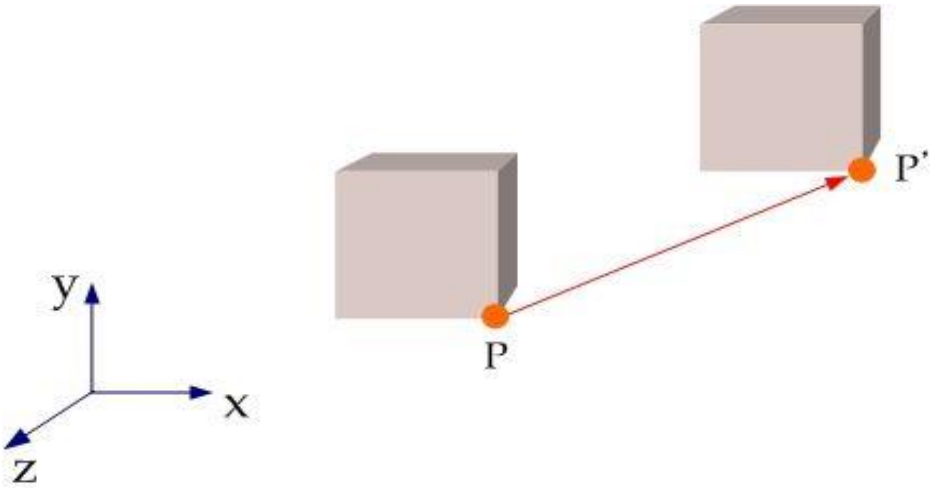
- x' = 1Xx + 0Xy + TxX1 $x' = x + t_x$
- y' = 0Xx + 1Xy + TyX1 $y' = y + t_y$



2차원 이동

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & Tx \\ 0 & 1 & Ty \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

OpenGL Transformation: 3차원 이동

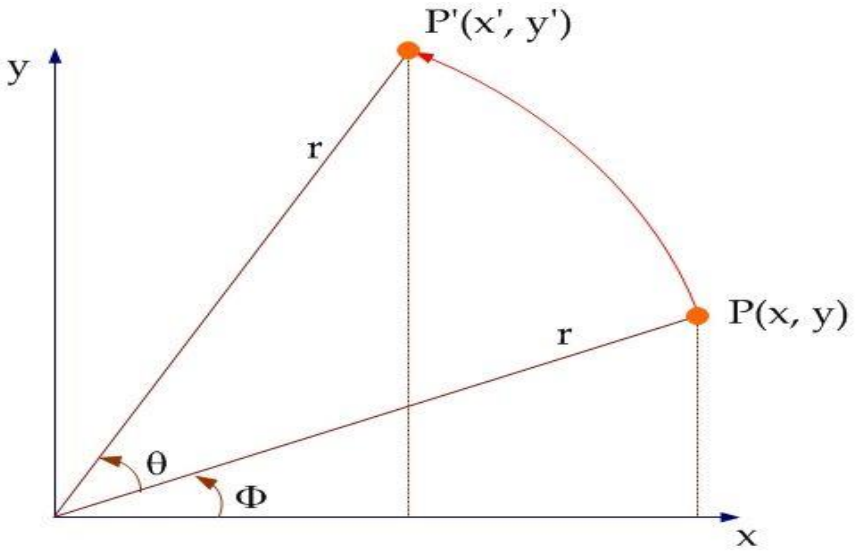


3차원 이동

$$P' = T \cdot P$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & Tx \\ 0 & 1 & 0 & Ty \\ 0 & 0 & 1 & Tz \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

OpenGL Transformation: 2차원 회전



2차원 회전

$$x' = r\cos(\phi+\theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta = x\cos\theta - y\sin\theta$$

$$y' = r\sin(\phi+\theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta = x\sin\theta + y\cos\theta$$

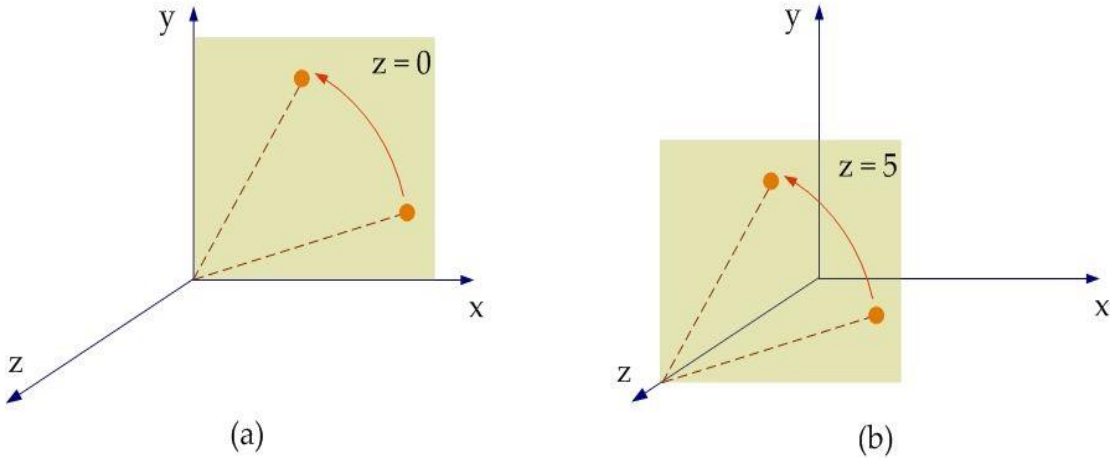
삼각함수 덧셈정리...

$$P' = R \cdot P$$

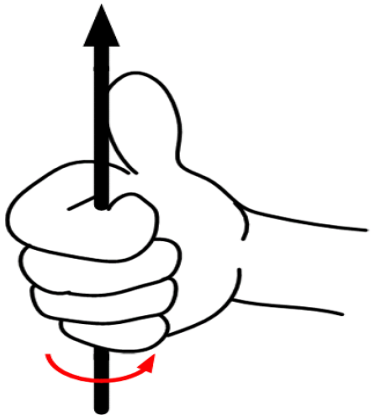
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

OpenGL Transformation: 3차원 회전

- 회전축 기준의 회전으로 정의
- 반 시계 방향의 회전각



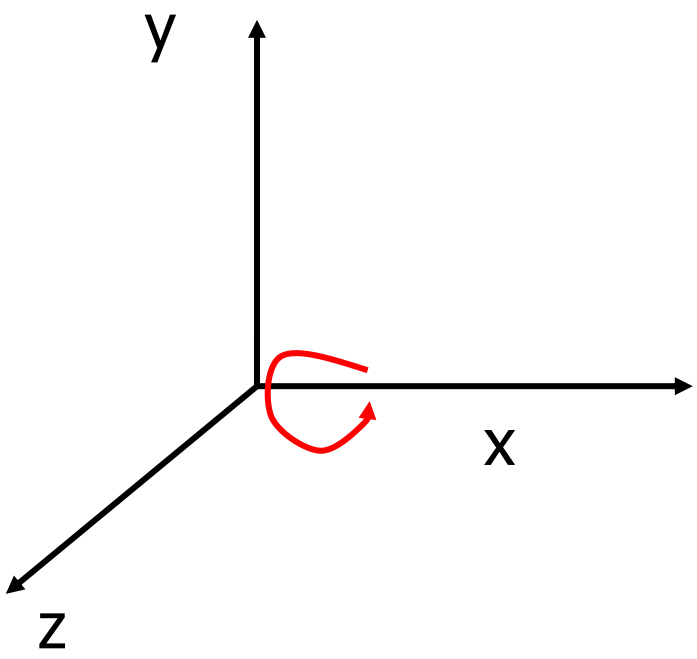
3차원 회전



반시계 방향

$$P' = Rz(\theta) \cdot P$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} cos\theta & -sin\theta & 0 & 0 \\ sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

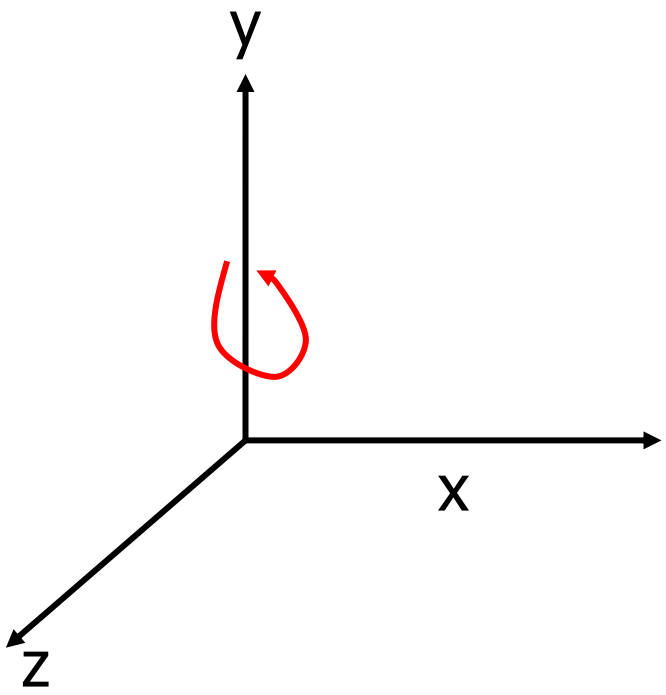


$$y' = y\cos\theta - z\sin\theta$$

$$z' = y\sin\theta + z\cos\theta$$

$$x' = x$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

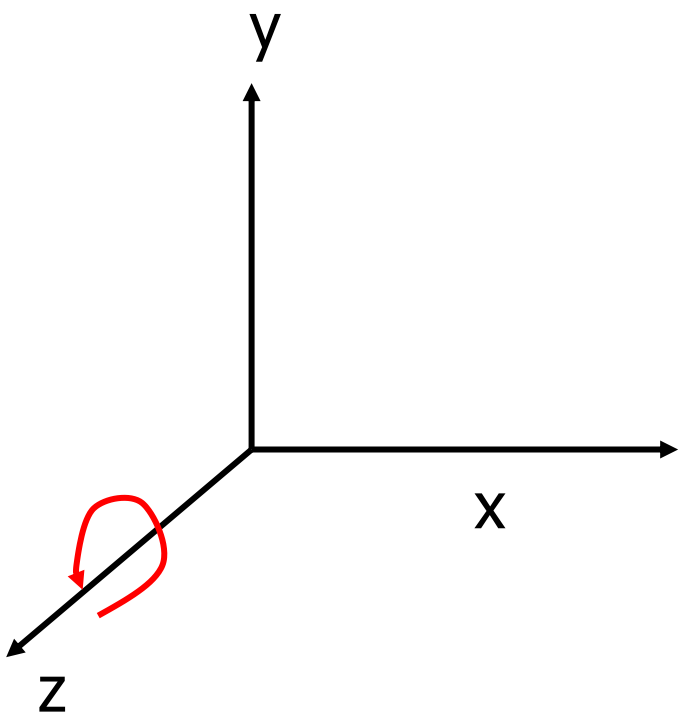


$$z' = z\cos\theta - x\sin\theta$$

$$x' = z\sin\theta + x\cos\theta$$

$$y' = y$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



$$x' = x\cos\theta - y\sin\theta$$

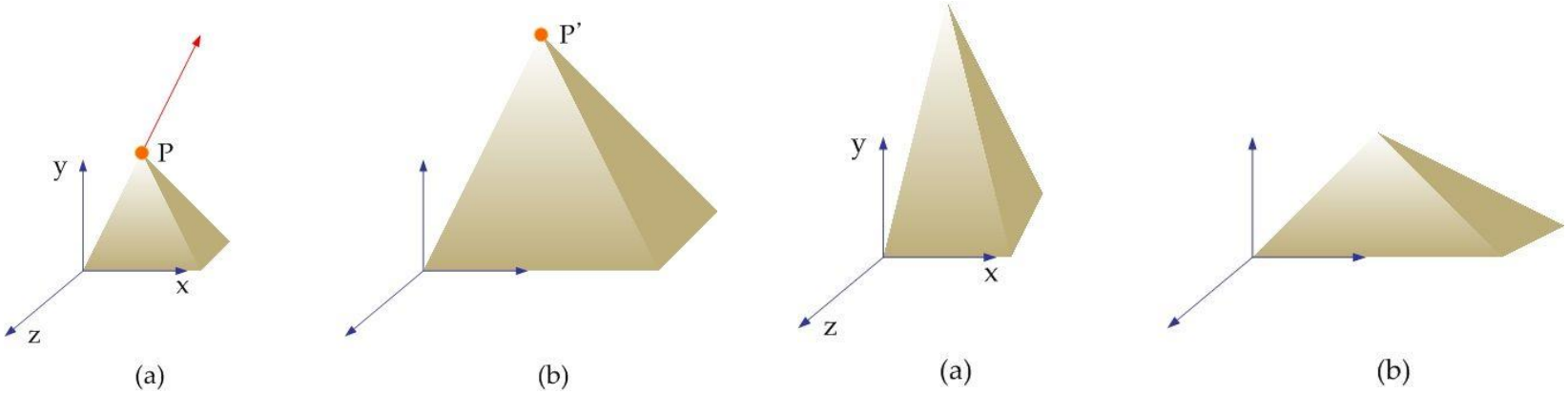
$$y' = x\sin\theta + y\cos\theta$$

$$z' = z$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

OpenGL Transformation: 크기 조절 (Scaling)

- 균등 크기조절(Uniform Scaling) vs. 차등 크기조절(Non-Uniform Scaling)



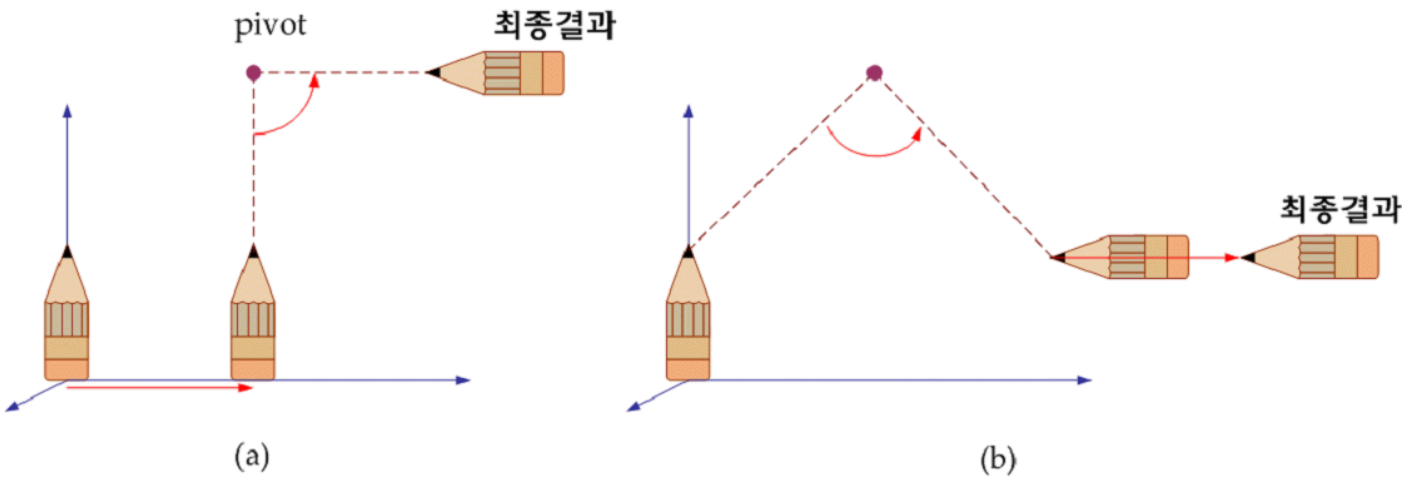
균등 크기조절

차등 크기조절

$$P' = S \cdot P$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

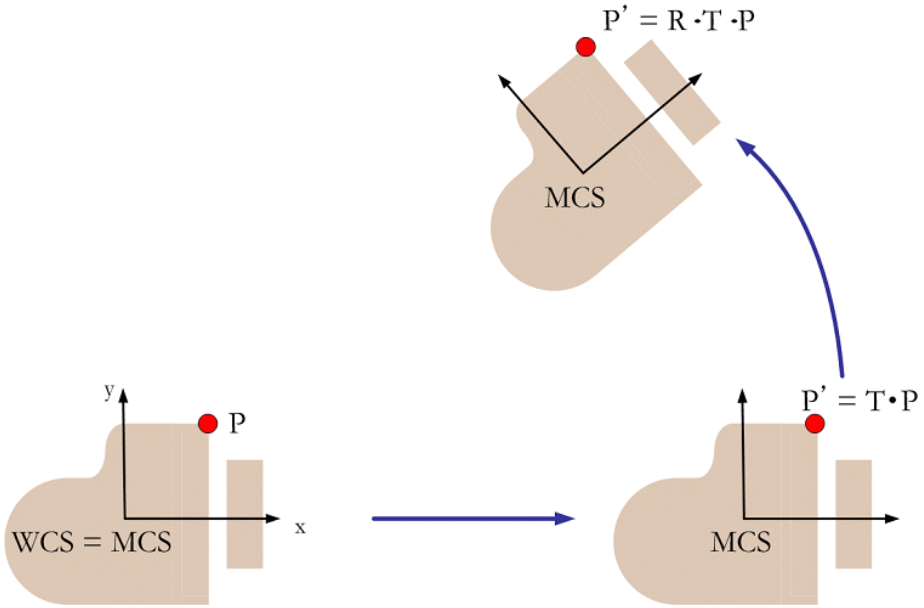
- 크기조절(S1) 후, 결과 물체를 회전(R1)한 후, 다시 크기조절(S2)
  - P' = S2 • R1 • S1 • P
  - 행렬곱셈의 순서에 유의
  - P' = C • P 복합행렬 C는 한번만 계산. 모든 정점에 적용
- 복합 변환에 대한 교환법칙은 성립하지 않음
  - R • T 와 T • R 은 서로 다른 결과를 가져옴
  - 이동 후 회전과 회전 후 이동의 결과는 다름



복합 변환 순서

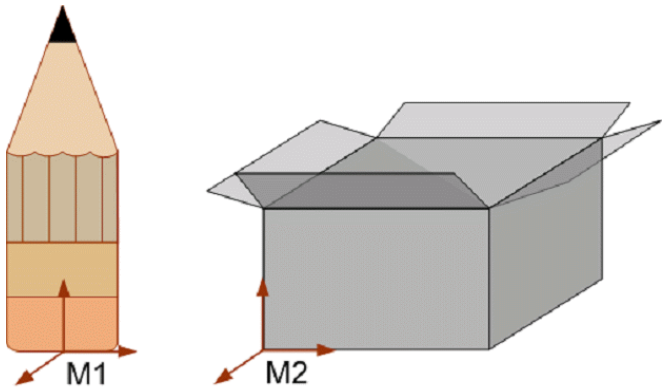
- 코드
  - glMatrixMode(GL_MODELVIEW);
  - glLoadIdentity( );
  - glRotatef(45, 0.0, 0.0, 1.0); 물체 변환의 역순
  - glTranslatef(10.0, 0.0, 0.0);
  - glVertex3f(Px, Py, Pz);

- 전역좌표 기준의 물체변환
  - P' = T•P
  - P'' = R•P' = R•T•P



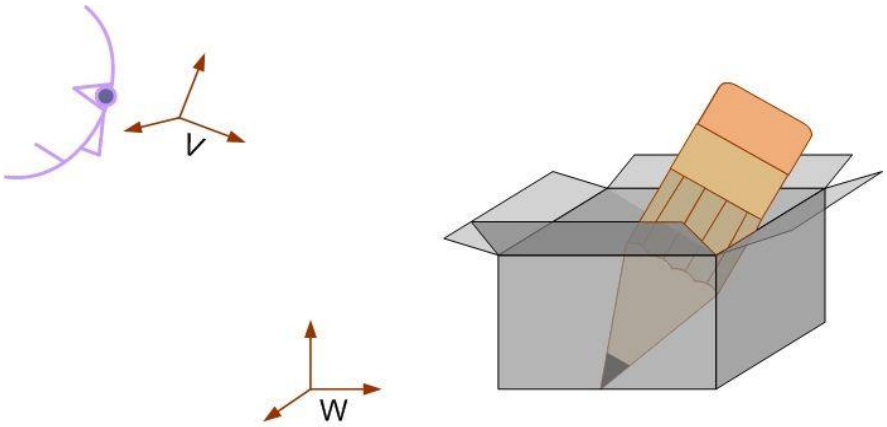
모델 좌표계

- 모델링
  - 물체의 형태를 정점들로 설계
- 모델 좌표계(MCS, Modeling Coordinate System) 또는 지역 좌표계(LCS, Local Coordinate System)
  - 물체마다 각자의 좌표계 사용
  - 설계상 편리



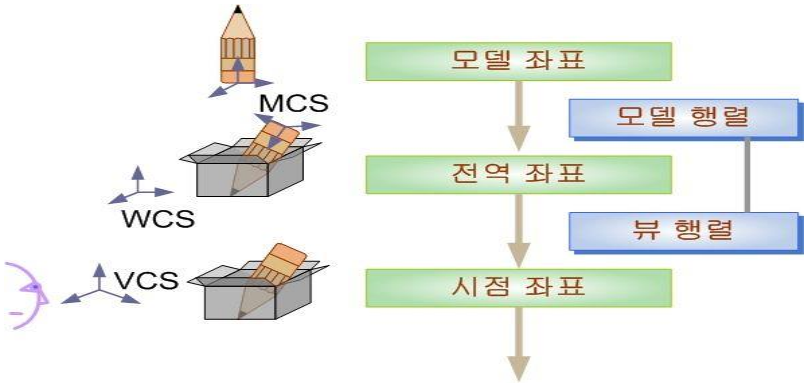
전역 좌표계 및 시점 좌표계

- 장면
  - 여러 개의 물체가 존재 (여러 개의 지역 좌표계가 존재)
  - 기준 좌표계가 필요
  - 전역 좌표계 (WCS, World Coordinate System)
- 시점
  - 바라보는 위치에 따라서 장면은 다르게 보임
  - 시점 좌표계(VCS, View Coordinate System)



OpenGL 파이프라인

- 모델 변환: 모델 행렬
  - 물체에 가해지는 기하변환: 이동, 회전 등
  - 모델 좌표에 모델 행렬을 곱하면 전역 좌표임
- 뷰 변환: 뷰 행렬
  - 카메라 위치와 방향 설정
  - 전역 좌표에 뷰 행렬을 곱하면 시점 좌표임
- OpenGL은 모델 행렬과 뷰행렬을 하나로 통합하여 모델뷰 행렬로 취급
  - 물체를 뒤로 이동하는 것과 카메라를 이동하는 것이 같음



실습 Code: 04_3 카메라 이동 1 예제

CGL
COMPUTER GRAPHICS LABORATORY

```
#include <GL/glut.h>
#include <stdlib.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);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity(); // 행렬 초기화

    /* Viewing Transformation */
    gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

    //glRotatef(90, 1.0, 0.0, 0.0);  // 카메라의 위치를 x축을 기준으로 90도 회전
    //glTranslatef(0.0, 0.0, -2.0);  // z축으로 -2.0만큼 이동. (물체가 작아짐)
    glutWireTeapot(1.0);

    glFlush();
}

void reshape(int w, int h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity(); // 행렬 초기화
    glFrustum(-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);

}
```

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

실습 Code: 04_4 카메라 이동 2 예제

```
#include <GL/glut.h>
#include <stdlib.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);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity(); // 행렬 초기화

    /* Viewing Transformation */
    gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
    //gluLookAt(0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -1.0);   // x축으로 90도 회전

    glutWireTeapot(1.0);

    glFlush();
}
```

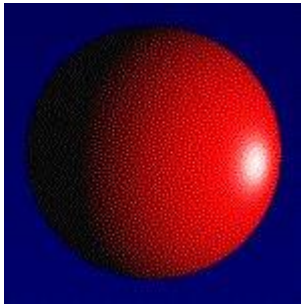
```
void reshape(int w, int h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity(); // 행렬 초기화
    glFrustum(-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);

}

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

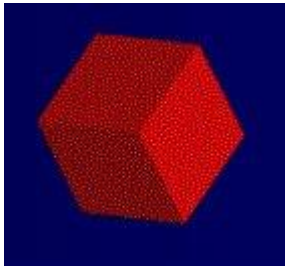
GLUT 3D Primitives

- glutWireSphere(GLdouble radius, GLint slices, GLint stacks)
- glutSolidSphere(GLdouble radius, GLint slices, GLint stacks)

- glutWireCube(GLdouble size)
- glutSolidCube(GLdouble size)

- glutWireTeapot(GLdouble size)
- glutSolidTeapot(GLdouble size)





CGL
COMPUTER GRAPHICS LABORATORY

THANK YOU

마 준 | majun@sch.ac.kr