컴퓨터 그래픽스
OpenGL 좌표계 변환

2016년 2학기

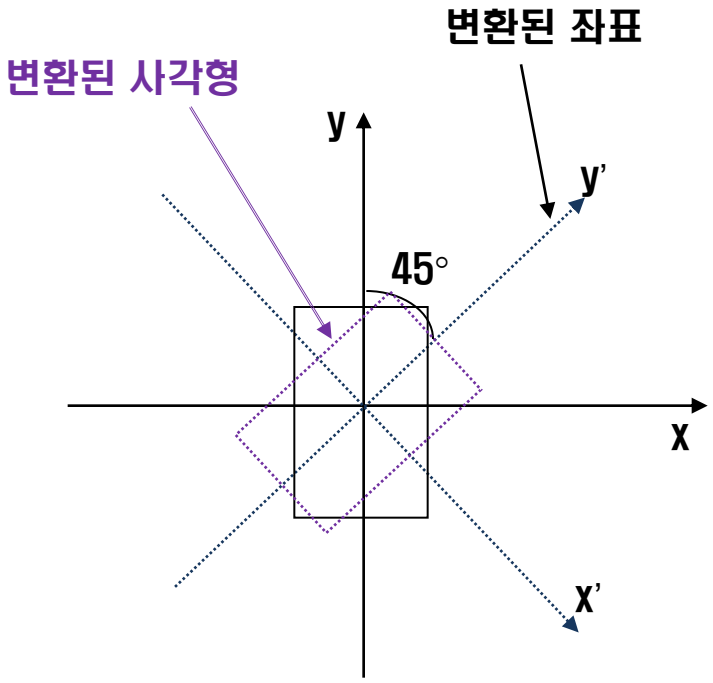
1. OpenGL 좌표계 변환 내용

- **좌표계 변환**
  - 좌표계
  - 변환
  - 모델링 변환
  - 행렬 스택 사용하기
  - 더블 버퍼링
  - 관측 (뷰잉) 변환
  - 투영 변환
  - 뷰포트 변환
  - 3차원 객체 만들기

좌표계

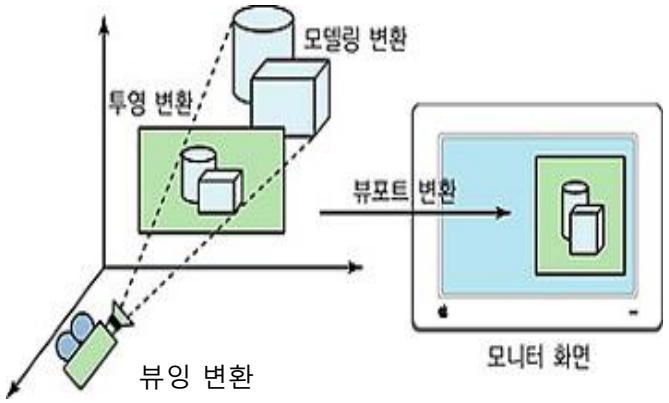
- 좌표계 변환
  - 그래픽 파이프라인에서 가장 중요한 개념
  - 물체는 좌표계에 따라 새로운 좌표 값으로 바뀌어 최종적으로 화면에 그려진다.
  - 시각 좌표계 (eye coordinate)
    - 변환에 관계없이 관측자의 시점에서 바라본 좌표
    - 스크린 좌표: 모든 변환에 영향을 받지 않는 절대적 화면 좌표

- OpenGL에서는
  - 직교 좌표계 (Cartesian Coordinate System)를 사용한다.
  - 어떠한 변환도 사용하지 않을 때:
    - 사용하는 좌표계는 시각 좌표계와 동일하다
  - 여러 가지 변환을 사용할 때:
    - 시각 좌표계의 관점으로 볼 때, 현재 좌표계가 변한다.
    - 시각 좌표계를 기준으로 좌표계를 이동, 회전시킨다.



변환

- OpenGL의 변환
  - **모델링 (Modeling) 변환:**
    - 3차원 공간에서 그래픽스 객체를 이동, 신축, 회전시켜주는 변환
    - 모델링 변환을 적용하는 순서에 따라 결과 값은 달라진다.
    - 물체를 뒤로 옮기는 것 = 좌표축을 앞으로 옮기는 것
      - 모델뷰 변환
  - **관측 (viewing) 변환 (뷰잉 변환):**
    - 관측자의 시점(viewpoint)을 설정하는 변환 (장면을 보는 위치를 결정)
    - 카메라의 위치를 잡는 것과 같은 효과를 내는 변환
    - 원하는 곳에 원하는 방향으로 관측점을 놓을 수 있다.
    - 기본적으로 관측점은 (0, 0, 0)이다. (z축의 음의 방향은 모니터의 안쪽)
  - **투영 (Projection) 변환:**
    - 3차원 그래픽스 객체를 2차원 평면으로 투영시키는 투영 변환
  - **뷰포트 (Viewport) 변환**
    - 투영된 그림이 출력될 위치와 크기를 정의하는 변환
    - 윈도우에 나타날 최종 화면의 크기 조절



변환

- OpenGL에서 변환은
  - **기본적으로 모든 변환은 현재의 행렬에 변환 행렬이 적용되어 최종 결과로 출력된다.**
    - 4 * 4 행렬을 사용한다.
      - 그리고자 하는 객체의 좌표에 변환 행렬이 적용된다.
      - 모든 변환은 변환 행렬로 대신한다 (모델 변환은 모델 행렬로, 시점 변환은 뷰 행렬로, 투영 변환은 투영 행렬을 사용)
      - 변환을 위하여 행렬을 지정해준다.

- **행렬 모드 설정**
  - 변환을 적용할 때 변환 행렬의 모드를 설정
    - void **glMatrixMode** (GLenum **mode**): 현재의 행렬 모드를 설정
      - GL_MODELVIEW: 모델링 및 뷰잉 변환 모드 (물체 이동 시)
        - » 디폴트 모드로 GL_MODELVIEW 로 설정되어 있다.
      - GL_PROJECTION: 투영 변환 모드 (클리핑 공간 정의)
      - GL_TEXTURE: 텍스처 매핑 모드 (텍스처 매핑 정의)
    - 변환 함수를 부를 때마다 현재의 모델 관측 행렬에 변환행렬이 곱해진다.

변환

- 선택된 변환 행렬의 값을 설정하고 조작을 수행하는 GL 함수들
  - void **glLoadIdentity** (): 현재의 변환 행렬을 단위행렬로 설정.
    - 시각 좌표계를 원점으로 초기화 한다
    - 행렬 변환을 수행하기 전에 좌표계를 초기화한다.
    - 초기화 결과: 모델 좌표계 = 전역 좌표계 = 시점 좌표계
  - void glLoadMatrixd (const GLdouble * *m*);
  - void glLoadMatrixf (const GLfloat * *m*);
    - m: 4*4 행렬 값 (16개의 연속된 값)
    - 현재의 행렬 (CTM, Current Transformation Matrix)을 m의 값으로 바꾼다.
      - 행렬은 열 우선 벡터를 사용한다.
  - void glMultMatrixf (const GLfloat *m);
  - void glMultMatrixd (const Gldouble *m);
    - 현재 행렬에 행렬 m을 곱한다.
  - 행 우선 행렬 사용
    - void glLoadTransposeMatrix{fd} (…);
    - void glMultTransposeMatrix{fd} (…);

변환


좌표계 변환
glLoadIdentity( );
glRotatef( );
glTranslatef( );
glVertex3f( );
물체 변환


좌표계 변환
지엘 함수호출
P' = I · R · T · P
물체 변환

변환: 모델링 변환

- 모델링 변환하기
  - 모델을 제어하는 과정으로 **이동, 회전, 신축**을 할 수 있다.
  - **glMatrixMode (GL_MODELVIEW)** 인자 설정

  - 이동: **glTranslate**
    - glTranslated (GLdouble x, GLdouble y, GLdouble z);
    - glTranslatef (GLfloat x, GLfloat y, GLfloat z);
      - y축의 양의 방향으로 10만큼 이동: glTranslatef (0.0f, 10.0f, 0.0f);

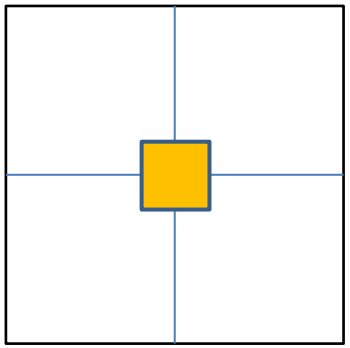
  - 회전: **glRotate**
    - glRotated (GLdouble angle, GLdouble x, GLdouble y, GLdouble z);
    - glRotatef (GLfloat angle, GLfloat x, GLfloat y, GLfloat z);
      - Angle: 도 (degree)단위로 나타낸다.
      - x, y, z: x, y, z 축 벡터 값
      - 원점과 (x, y, z)을 지나는 선을 축으로 angle만큼 회전시킨다.
    - x축에 대하여 45도만큼 회전: glRotatef (45.0f, 1.0f, 0.0f, 0.0f);

  - 신축: **glScale**
    - glScaled (GLdouble x, GLdouble y, GLdouble z);
    - glScalef (GLfloat x, GLfloat y, GLfloat z);
      - x축으로 2배, y축으로 0.5배만큼 신축: glScalef (2.0f, 0.5f, 1.0f);

변환: 모델링 변환

- 예) 육면체 그리기

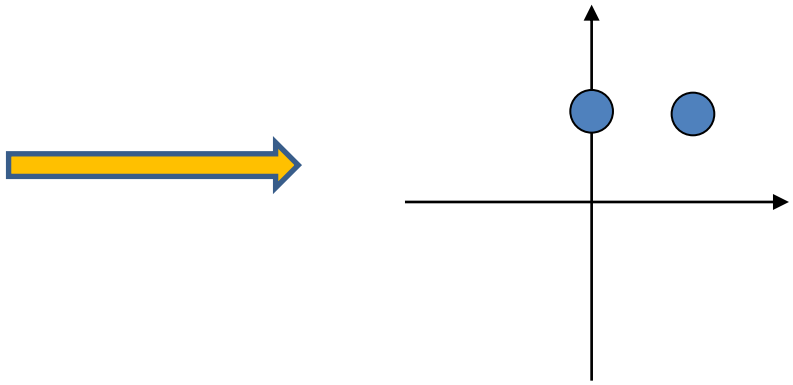
```
void DrawScene () {
    glClear (GL_COLOR_BUFFER_BIT);
    glMatrixMode (GL_MODELVIEW);

    glBegin (GL_LINES);
        glVertex2f (-1.0, 0.0);
        glVertex2f (1.0, 0.0);
        glVertex2f (0.0, -10.0);
        glVertex2f (0.0, 1.0);
    glEnd ();

    glLoadIdentity ();
    glutSolidCube (0.3);
    glFlush ()
}
```



변환: 모델링 변환

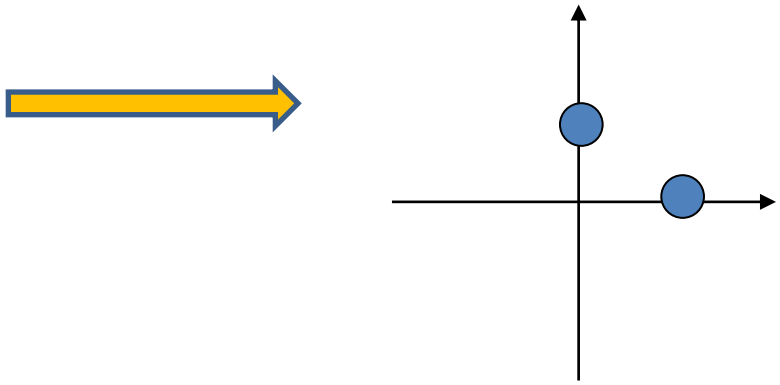
- 변환 함수는 누적 방식으로 수행된다.

```
glTranslatef (0.0f, 10.0f, 0.0f);
glutSolidSphere (1.0f, 15, 15);
glTranslatef (10.0f, 0.0f, 0.0f);
glutSolidSphere (1.0f, 15, 15);
```



- 우측과 같은 결과를 얻으려면

```
// 사용하는 행렬을 모델관측행렬로 설정하고 초기화한다.
glMatrixMode (GL_MODELVIEW);
glLoadIdentity ();
glTranslatef (0.0f, 10.0f, 0.0f);
glutSolidSphere (1.0f, 15, 15);

// 모델관측행렬을 다시 초기화한다.
glLoadIdentity ();
glTranslatef (10.0f, 0.0f, 0.0f);
glutSolidSphere (1.0f, 15, 15);
```

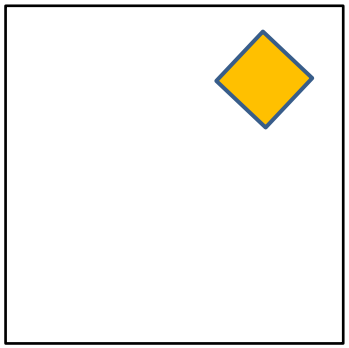


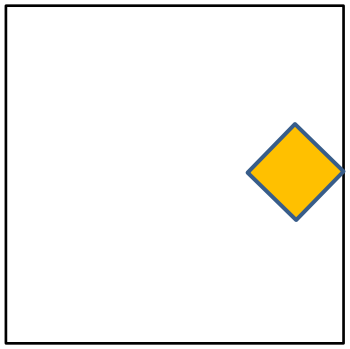
변환: 모델링 변환

- 예) 육면체에 회전, 이동 모델링 변환 적용

```
void DrawScene () {
    glClear (GL_COLOR_BUFFER_BIT);
    glmatrixMode (GL_MODELVIEW);
    glLoadIdentity ();
    glRotatef (45.0, 0.0, 0.0, 1.0);
    glTranslatef (0.6, 0.0, 0.0);
    glutSolidCube (0.3);
    glFlush ()
}
```

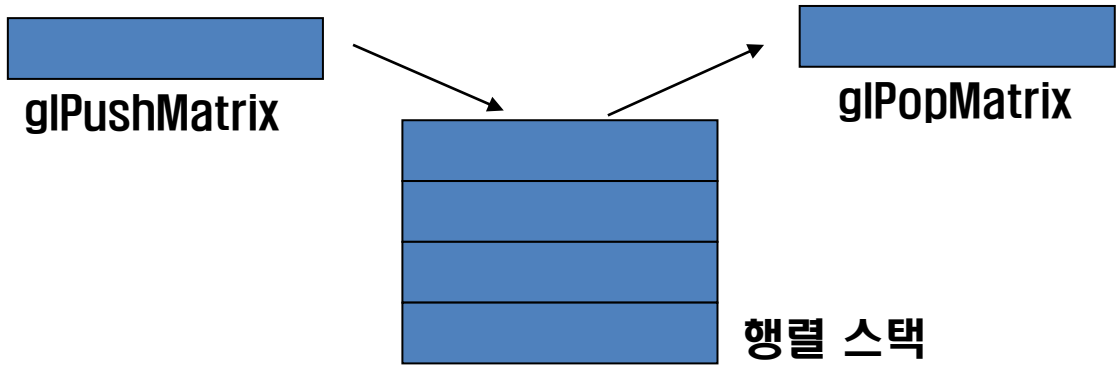
```
void DrawScene () {
   glClear (GL_COLOR_BUFFER_BIT);
   glmatrixMode (GL_MODELVIEW);
   glLoadIdentity ();
   glTranslatef (0.6, 0.0, 0.0);
   glRotatef (45.0, 0.0, 0.0, 1.0);
   glutSolidCube (0.3);
   glFlush ()
}
```



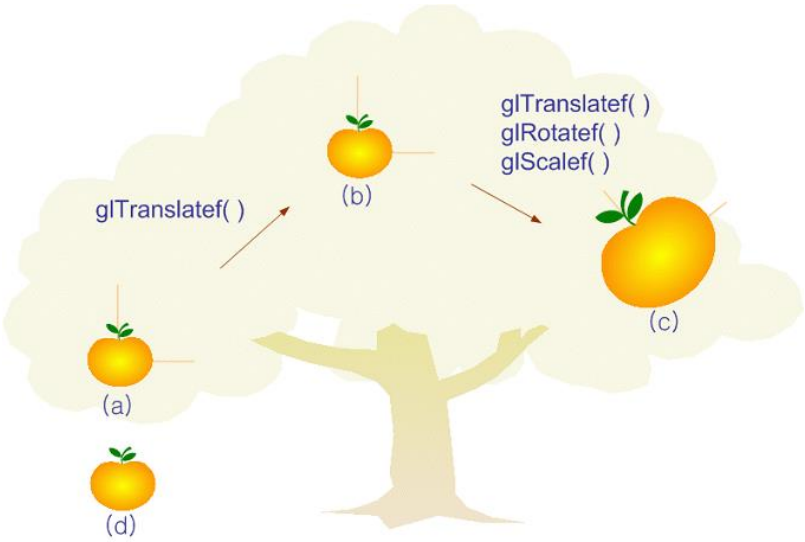


행렬 스택

- 행렬 스택 사용하기
  - 매번 모델관측행렬을 초기화하는 것은 바람직하지 않다.
  - 현재 변환상태를 저장하고 특정물체를 지정한 후 다시 복구하는 기능이 필요하다.
    - 모델뷰 행렬과 투영 행렬 모드에는 행렬을 저장하는 스택이 있는데, 현재 행렬이 스택의 맨 위에 저장되어 있다.
  - 행렬 스택(matrix stack)을 사용한다.
    - void **glPushMatrix** (): 스택에 행렬을 저장한다.
    - void **glPopMatrix** (): 스택의 행렬을 꺼낸다.

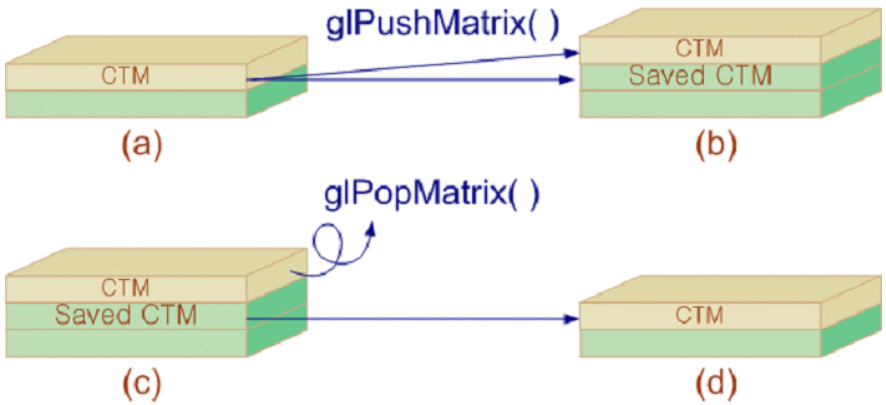


행렬 스택

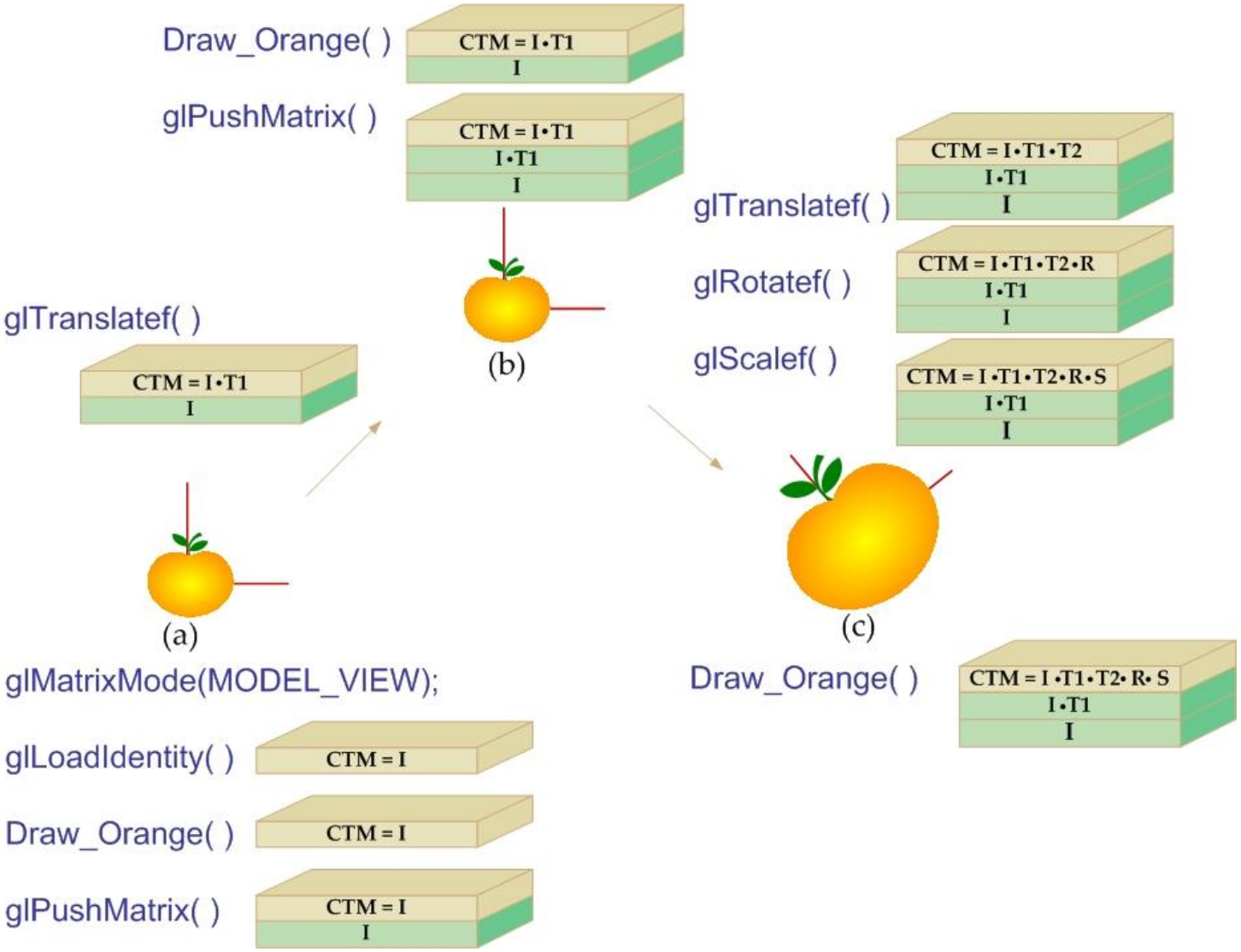


호출 함수	현 변환행렬	작업
glMatrixMode(GL_MODELVIEW);	CTM ← ModelView CTM	모델뷰 행렬 선택
① glLoadIdentity( );	CTM = I	초기화
② Draw_Orange( );	P' = CTM • P	(a)의 오렌지 그리기
③ glTranslatef(4.0, 4.0, 0.0);	CTM = CTM • T1	좌표계 이동
④ Draw_Orange( );	P' = CTM • P	(b)의 오렌지 그리기
⑤ glTranslatef(6.0, -2.0, 0.0);	CTM = CTM • T2	좌표계 이동
⑥ glRotatef(45, 0.0, 0.0, 1.0);	CTM = CTM • R	좌표계 회전
⑦ glScalef(2.0, 2.0, 2.0);	CTM = CTM • S	좌표계눈금 크기조절
⑧ Draw_Orange( );	P' = CTM • P	(c)의 오렌지 그리기

행렬 스택


glPushMatrix( )
CTM
CTM
Saved CTM
(a)
(b)
glPopMatrix( )
CTM
Saved CTM
CTM
(c)
(d)

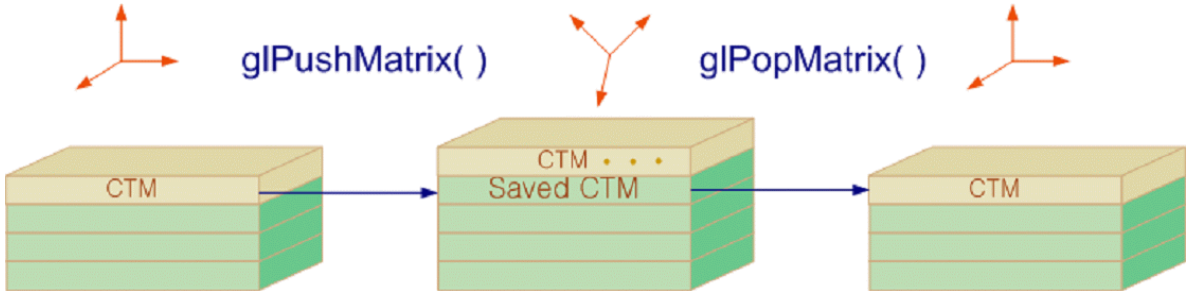
행렬 스택


glMatrixMode(MODEL_VIEW);
glLoadIdentity( )
CTM = I
Draw_Orange( )
CTM = I
glPushMatrix( )
CTM = I
I
(a)
glTranslatef( )
CTM = I•T1
I
Draw_Orange( )
CTM = I•T1
I
glPushMatrix( )
CTM = I•T1
I•T1
I
(b)
glTranslatef( )
CTM = I•T1•T2
I•T1
I
glRotatef( )
CTM = I•T1•T2•R
I•T1
I
glScalef( )
CTM = I•T1•T2•R•S
I•T1
I
(c)
Draw_Orange( )
CTM = I•T1•T2•R•S
I•T1
I

행렬 스택

- 일반적인 형태

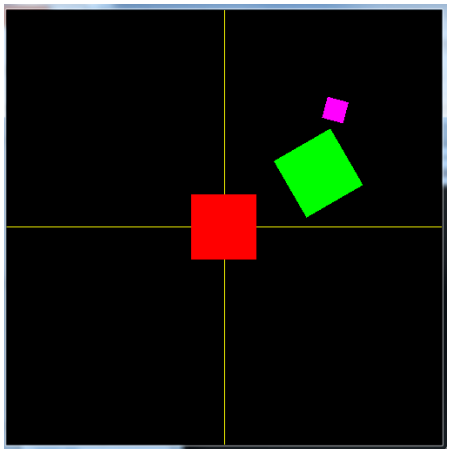
```
glPushMatrix( );
        glTranslatef( );
        glRotatef( );
        glScalef( );
        ...
        Draw_TransformedObject( );
glPopMatrix( );
```



행렬 스택

- Push/pop 사용 코드 예제

```
Void DrawScene () {
        glClear (GL_COLOR_BUFFER_BIT);
        glMatrixMode (GL_MODELVIEW);

         glLoadIdentity ();
         glColor3f (1.0, 0.0, 0.0);                    // 빨강색
         glutSolidCube (0.3);
         glPushMatrix ();
             glRotatef (30.0, 0.0, 0.0, 1.0);
             glTranslatef (0.5, 0.0, 0.0);
             glColor3f (0.0, 1.0, 0.0);                // 초록색
             glutSolidCube (0.3);
             glPushMatrix ();
                 glRotatef (45.0, 0.0, 0.0, 1.0);
                 glTranslatef (0.3, 0.0, 0.0);
                 glColor3f (1.0, 0.0, 1.0);            // 보라색
                 glutSolidCube (0.1);
             glPopMatrix ();
         glPopMatrix ();

         glFlush ();
}
```

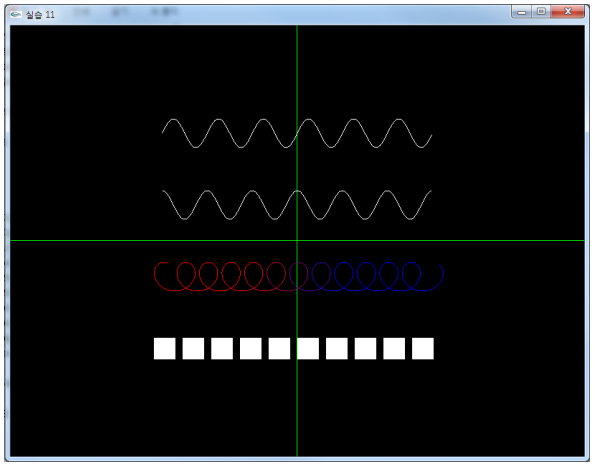


더블 버퍼링

- 더블 버퍼링 (double-buffering)
  - 그리기를 실행하는 동시에 화면에 나타나지 않는 버퍼(off screen)에 렌더링을 할 수 있다.
  - 스왑(swap) 명령으로 버퍼에 렌더링한 그림을 스크린 상에 즉시 나타낼 수 있다.
  - 더블 버퍼 사용
    - 시간이 오래 걸리는 복잡한 그림을 그린 후 완성된 그림을 화면에 보여줄 수 있다.
    - 애니메이션에서 사용할 수 있다.
  - 사용 방법
    - 출력 모드를 더블 버퍼링을 위해 설정한다.
      - glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
    - 그리기 함수에서 드로잉 명령을 실행하고 버퍼 교체를 설정한다.
      - glutSwapBuffers (); // 그리기 함수에서 glFlush 대신 사용
        - » glFlush 효과가 있으므로 glFlush를 사용할 필요가 없다.

실습 11

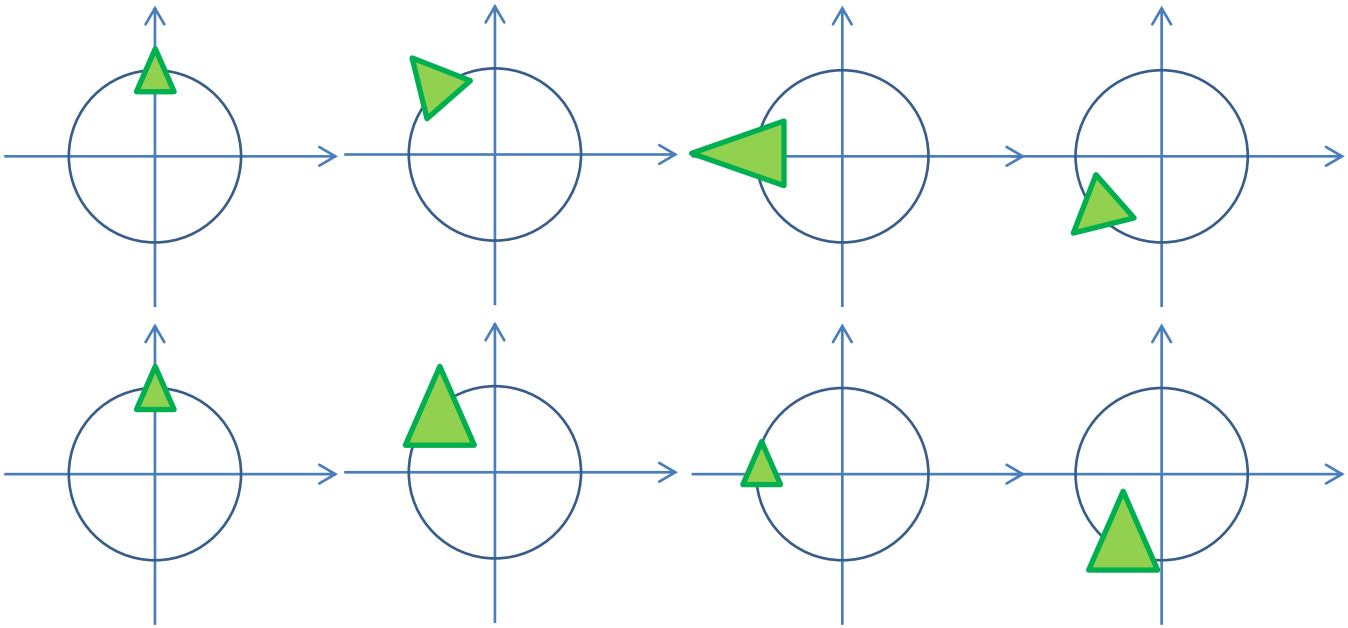
- 화면에 곡선 그리기
- 화면 전체에 사인, 코사인, 스프링, 직사각형 그리기
  - XY 평면에 위의 곡선 또는 직선을 위에서 아래로 순서대로 그린다.
  - 스프링을 그릴 때, GL_LINE_STRIP으로 설정
  - 스프링의 색은 그라데이션 (예, 빨강 → 파랑)을 넣기
  - (좌측 그림은 스프링 곡선, 우측 그림은 사인곡선)
  - 다음의 명령어를 수행한다.
    - x/y/z: x/y/z축에 대하여 회전
    - S/s: 제자리에서 확대/축소
    - 좌측 화살표/우측 화살표: x축을 따라 좌측/우측 이동
    - A/a: 애니메이션 시작/종료
    - i: 제자리로 리셋
    - q/Q: 프로그램 종료



실습 12

- **원의 경로를 따라 이동/회전하는 애니메이션 구현하기**
  - 오픈지엘 변환 함수를 사용한다.
  - 화면 중앙에 좌표축이 그려진다.
  - 좌표측의 원점을 중심으로 xy 평면에 원이 그려진다.
  - 키보드 명령에 따라 선택된 도형이 원의 경로를 따라 회전 또는 이동하고 크기가 확대→축소→확대 → …된다.
    - 도형 종류: 삼각형, 사각형
  - 또 키보드 명령어에 따라 경로 바꾸기
    - 1: 사인 곡선
    - 2: 회오리
    - 3: 지그재그
    - 4: 경로 그리기 (4번을 선택하면 왼쪽 마우스 클릭을 이용하여 직선 경로를 그리고, 오른쪽 마우스를 클릭하면 경로 그리기가 끝나고 삼각형이 경로를 따라 이동한다. 왼쪽 마우스는 최대 5번까지 클릭할 수 있다.)

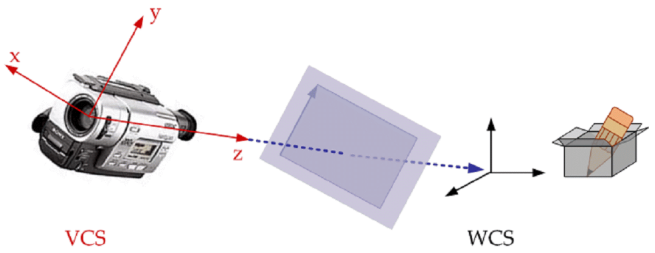
실습 12



실습 13

변환: 관측 (뷰잉) 변환

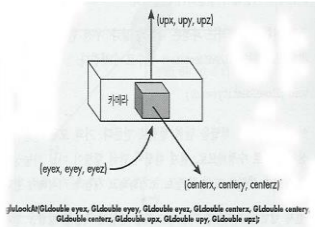
- 관측 변환하기
  - 카메라 위치 = 시점 좌표계 원점
    - 전역 좌표계 원점을 향한 방향이 시점 좌표계의 z축
    - z축에 수직으로 서 있는 면 = 투상면
    - 투상면 내부에 뷰 윈도우 = 카메라 필름
    - 시점 좌표계 y 축 = 뷰 윈도우의 y축과 평행
    - y-z 평면에 수직인 방향으로 x축



  - 관측 변환과 모델링 변환은 같이 생각할 수 있다.
    - **카메라를 오른쪽으로 이동 -> 객체를 왼쪽으로 이동**
    - **glMatrixMode (GL_MODELVIEW)**로 인자 설정
    - 모델링 변환 함수들을 사용하여 카메라의 위치를 바꿀 수도 있다.
      - glTranslate 나 glRotate를 사용할 수 있다.

변환: 관측 (뷰잉) 변환

- void **gluLookAt** (GLdouble eyeX, GLdouble eyeY, GLdouble eyeZ,
  GLdouble centerX, GLdouble centerY, GLdouble centerZ,
  GLdouble upX, GLdouble upY, GLdouble upZ);
  - 카메라의 위치와 방향 (카메라가 바라보는 방향)을 변경시킨다.
    - 파라미터: 카메라의 위치, 카메라가 바라보는 점, 카메라의 기울임
      - » eyeX, eyeY, eyeZ: 눈의 위치
      - » centerX, centerY, centerZ: 주시하고 있는 카메라의 주시점
      - » upX, upY, upZ: up-vector 방향



- glTranslated (-eyeX, -eyeY, -eyeZ)과 마찬가지 결과이다.
- **카메라의 디폴트 위치는 원점, 디폴트 방향은 z축의 음의 방향**,

- 디폴트 위쪽 방향은 y축의 양의 방향
  gluLookAt (0.0, 0.0, 0.0, 0.0, 0.0, -100.0, 0.0, 1.0, 0.0);

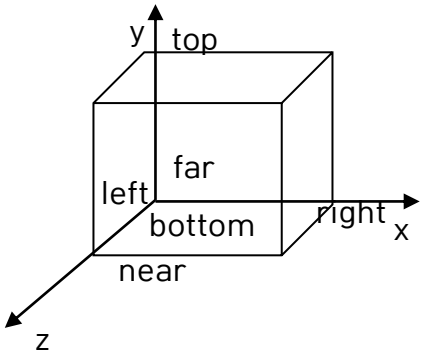
변환: 관측 (뷰잉) 변환

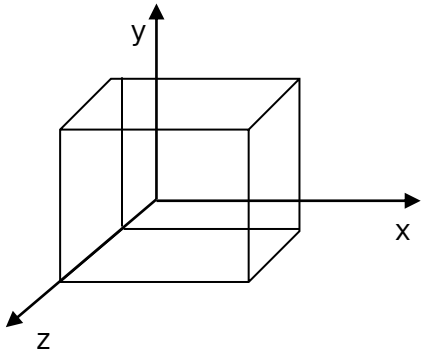
- 관측 변환 예

```
void DrawScene () {
        glClear (GL_COLOR_BUFFER_BIT);
        glMatrixMode (GL_MODELVIEW);
        glLoadIdentity ();
        gluLookAt (0.0, 0.0, 0.0,     0.0, 0.0, -1.0,     1.0, 1.0, 0.0);
        glutWireTeapot (1.0);
}
```

변환: 투영 변환

- 직각 투영
  - 투영 변환을 위해서 모드 설정
    - glMatrixMode (GL_PROJECTION)
  - 직각 투영을 사용하여 뷰포트의 공간을 설정한다.
    - void **glOrtho** (GLdouble left, GLdouble right,
      GLdouble bottom, GLdouble top, GLdouble near, GLdouble far);
      - 3D 좌표계 영역과 범위를 설정한다.
      - left, right: 투영 영역의 x축 좌측/우측 좌표
      - bottom, top: 투영 영역의 y축 아래측/위측 좌표
      - near, far: 투영 영역에서 가까운/먼 z축 거리



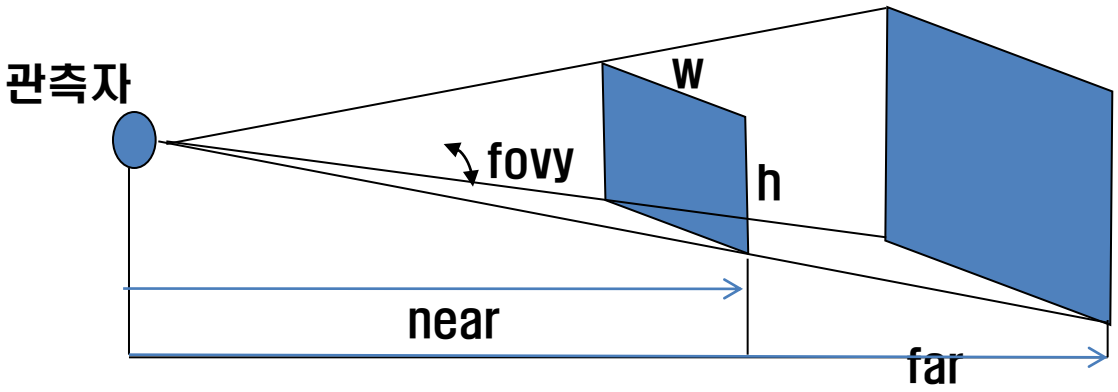


변환: 투영 변환

- 원근 투영 (Perspective Projection)
  - void **gluPerspective** (GLdouble fovy, GLdouble aspect,
    GLdouble zNear, GLdouble zFar);
    - 뷰잉 프러스텀을 월드 좌표계 시스템으로 지정한다.
    - **fovy**: 수직 방향의 보이는 각도 (y축 방향)
    - **aspect**: 종횡비 (앞쪽의 클리핑 평면의 폭(w)을 높이(h)로 나눈 값)
      - 종횡비: 화면의 가로방향에 대한 단위 길이를 나타내는 픽셀수에 대한 세로 방향의 단위길이를 나타내는 픽셀 수의 비율.
      - 예) "종횡비가 0.5": 가로길이의 두 픽셀이 세로길이의 한 픽셀에 대응한다.
    - **zNear**: 관측자에서부터 가까운 클리핑 평면까지의 거리 (항상 양의 값)
    - **zFar**: 관측자에서 먼 클리핑 평면까지의 거리 (항상 양의 값)

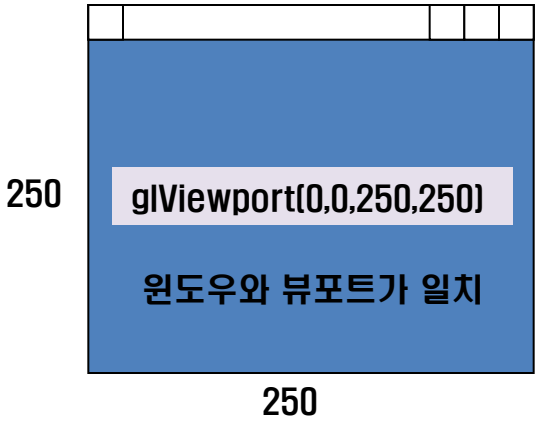
    - 예) gluPerspective (60.0f, w/h, 1.0, 400.0);

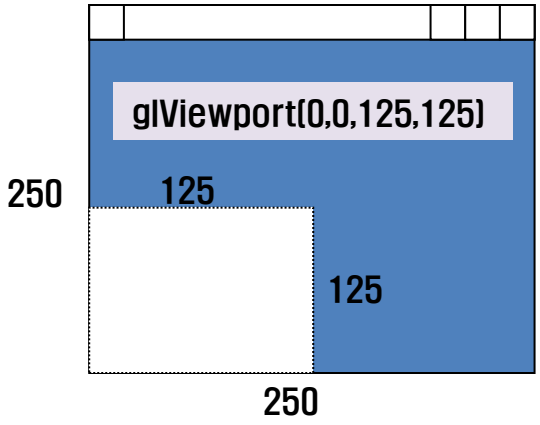
  - void glFrustum (left, right, bottom, top, near, far);



변환: 뷰포트 변환

- 뷰포트
  - 3차원 모델에서 2차원 평면으로 투영된 그림이 화면에서 최종적으로 출력될 영역 (사용할 영역)
  - 뷰포트 정하기:
    - void **glViewport** (GLint x, GLint y, GLsizei width, GLsizei height);
      - 윈도우의 영역을 설정한다.
      - x, y: 뷰포트 사각형의 왼쪽 아래 좌표
      - width, height: 뷰포트의 너비와 높이



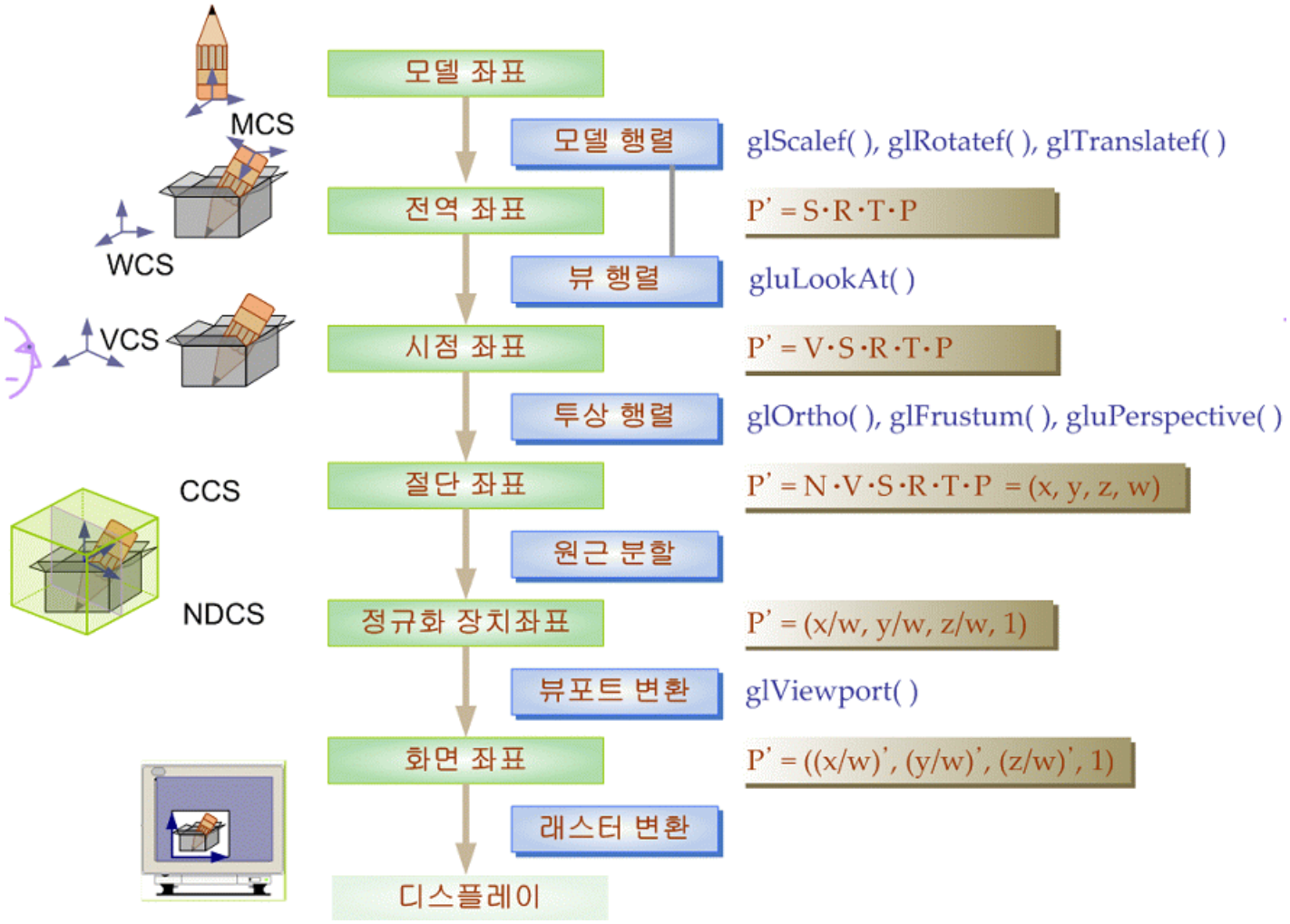


변환: reshape 함수에서 투영 변환, 뷰포트 변환 적용

- **투영 변환/뷰포트 변환 예제**
  - void Reshape (int w, int h) {

```
GLfloat nRange = 800.0f;

//--- 뷰포트 변환 설정
glViewport(0, 0, w, h);

// 투영 행렬 스택 재설정
glMatrixMode(GL_PROJECTION);
glLoadIdentity();

//-- 투영은 직각 투영 또는 원근 투영 중 한 개를 설정한다.
// 1. 클리핑 공간 설정: 원근 투영인 경우
gluPerspective (60.0f, w/h, 1.0, 1000.0);
glTranslatef (0.0, 0.0, -300.0);      // 투영 공간을 화면 안쪽으로 이동하여 시야를 확보한다.

// 2. 클리핑 공간 설정: 직각 투영인 경우
// glOrtho  (0, 800.0, 0.0, 600.0, -1.0, 200.0);


// 모델 뷰 행렬 스택 재설정
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
}
```

지엘의 변환 파이프라인


MCS
WCS
VCS
CCS
NDCS
모델 좌표
모델 행렬
glScalef( ), glRotatef( ), glTranslatef( )
전역 좌표
P' = S·R·T·P
뷰 행렬
gluLookAt( )
시점 좌표
P' = V·S·R·T·P
투상 행렬
glOrtho( ), glFrustum( ), gluPerspective( )
절단 좌표
P' = N·V·S·R·T·P = (x, y, z, w)
원근 분할
정규화 장치좌표
P' = (x/w, y/w, z/w, 1)
뷰포트 변환
glViewport( )
화면 좌표
P' = ((x/w)', (y/w)', (z/w)', 1)
래스터 변환
디스플레이

OpenGL 프로그램 작성하기

- 프로그램 기본 형태

```
#include <gl/glut.h>                                    // 헤더 파일
                                                        // 필요한 전역 변수 선언

void main ()
{
        // 윈도우 초기화 및 생성
        glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
        glutInitWindowSize ( 500, 500 );
        glutCreateWindow ("Points Drawing");

        // 상태 변수 초기화 함수
        SetupRC ()

        // 필요한 콜백 함수 설정
        glutDisplayFunc (DrawScene);                    // 출력 콜백 함수
        glutReshapeFunc (Reshape);                      // 다시 그리기 콜백 함수
        glutKeyboardFunc (Keyboard);                    // 키보드 입력 콜백 함수
        glutTimerFunc (100, TimerFunction, 1);                  // 타이머 콜백 함수

        glutMainLoop();                                 // 이벤트 루프 실행하기


}
```

OpenGL 프로그램 작성하기

```
// 초기화 함수 (Optional): 필요한 경우에 작성, 초기화해야 할 변수들이 많을 때는 만드는 것이 유리
void SetupRC ( )            {
      // 필요한 변수들, 좌표값 등의 초기화
      // 기능 설정 초기화
}

//-------------------------------------------------------------------------------------------------------
----------
// 렌더링을 위한 디스플레이 콜백 함수: 모든 그리기 명령은 이 함수에서 대부분 처리 함
void DrawScene ( )
{
      glClearColor(0.0f,0.0f,1.0f,1.0f);
      glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);          // 윈도우, 깊이 버퍼 클리어 하기

      // 필요한 변환 적용
      //--- 변환을 적용하기 위해서
      //        glPushMatrix 함수를 호출하여 기존의 좌표 시스템을 저장
      //        필요한 경우 행렬 초기화 ( glLoadIdentity ( ); )
      //        변환 적용: 이동, 회전, 신축 등 모델에 적용 할 변환 함수를 호출한다.
      //        변환이 끝난 후에는 원래의 좌표시스템을 다시 저장하기 위하여 glPopMatrix 함수 호출

      glutSwapBuffers ( );                              // 결과 출력
}
```

OpenGL 프로그램 작성하기

```
// 다시그리기 콜백 함수
//     처음 윈도우를 열 때, 윈도우 위치를 옮기거나 크기를 조절할 때 호출
//     뷰포트 설정, 투영 좌표계 설정, 관측 좌표 설정 등을 한다.
Void Reshape(int w, int h)
{
      // 뷰포트 변환 설정: 출력 화면 결정
      glViewport (0, 0, w, h);

      // 클리핑 변환 설정: 출력하고자 하는 공간 결정
      // 아래 3줄은 투영을 설정하는 함수
      glMatrixMode (GL_PROJECTION);
      glLoadIdentity ( );
      // 원근 투영을 사용하는 경우:
      gluPerspective (60.0, 1.0, 1.0, 1000.0);
      glTranslatef (0.0, 0.0, -300.0);

      // glOrtho (0.0, 800.0, 0.0, 600.0, -1.0, 1.0);

      // 모델링 변환 설정: 디스플레이 콜백 함수에서 모델 변환 적용하기 위하여 Matrix mode 저장
      glMatrixMode (GL_MODELVIEW);

      // 관측 변환: 카메라의 위치 설정 (필요한 경우, 다른 곳에 설정 가능)
      gluLookAt (0.0, 0.0, 0.0,   0.0, 0.0, 1.0,   0.0, 1.0, 0.0);

}
```

OpenGL 프로그램 작성하기

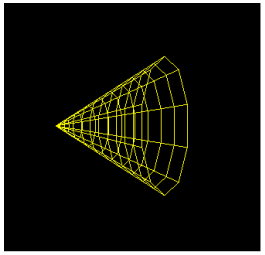
```
//  필요한 콜백 함수 구현: 키보드 입력, 마우스 입력, 타이머 등
Void Keyboard (unsigned char key, int x, int y)
{
        …
        glutPostRedisplay ( );                          // 화면 재출력을 위하여 디스플레이 콜백 함수 호출
}

Void TimerFunction (int value)
{
        glutPostRedisplay ();                  // 화면 재출력을 위하여 디스플레이 콜백 함수 호출
        glutTimerFunc (100, TimerFunction, 1);
}
```

OpenGL 프로그램 작성하기

- **변환**
  - 클리핑 영역 설정 // 다시 그리기 함수에서 처리
    - glMatrixMode (GL_PROJECTION); 설정 후
    - glOrtho (…) 또는 gluPerspective (…)
  - 카메라 위치 설정
    - gluLookAt (…);
  - 출력 영역 설정
    - glViewport (…);
  - 객체 위치 이동 // 그리기 함수에서 처리
    - glMatrixMode (GL_MODELVIEW); 설정 후
    - glTranslatef (…) / glRotatef (…) / glScalef (…);

객체 만들기

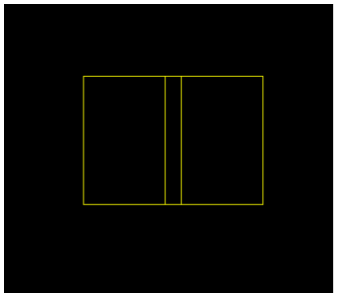
- 솔리드 혹은 와이어프레임 원뿔 그리기
  - void **glutSolidCone** (GLdouble base, GLdouble height, GLint slices, Glint stacks);
  - void **glutWireCone** (GLdouble base, GLdouble height, GLint slices, GLint stacks);
    - base: 원뿔의 밑면의 반지름
    - height: 원뿔의 높이
    - slices: z축 주위의 분할 개수
    - stacks: z축 방향의 분할 개수



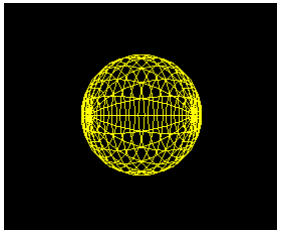
- 솔리드 혹은 와이어프레임 원뿔을 z축에 따라 그린다. 베이스는 z=0에, 원뿔의 제일 꼭대기는 z=height에 그리고, z축 주위로 slices갯수만큼 z축따라 stacks로 분할한다.

객체 만들기

- 솔리드 혹은 와이어프레임 육면체 그리기
  - void **glutSolidCube** (GLdouble size);
  - void **glutWireCube** (GLdouble size);
    - 원점을 중심으로 육면체를 그린다.
      - Size: 육면체의 각 모서리의 길이
      - 현재 좌표계의 중앙에 그린다.

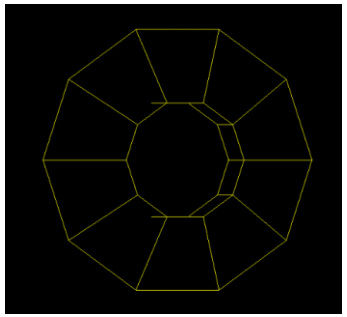


- 솔리드 혹은 와이어 프레임 구 그리기
  - void **glutWireSphere** (GLdouble radius, GLint slices, GLint stacks);
  - void **glutSolidSphere** (GLdouble radius, GLint slices, GLint stacks);
    - 구를 원점을 기준으로 그리며 반지름은 1.0이다
      - Radius: 구의 반지름
      - Slices: 구의 z축 둘레의 분할 개수 (경도와 비슷)
      - Stacks: 구의 z축따라 분할 개수 (위도와 비슷)
      - 현재 좌표계의 중앙에 radius 반지름으로 구를 그린다.

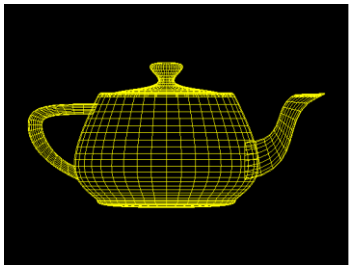


객체 만들기

- 솔리드 혹은 와이어프레임 토러스를 그린다.
  - void **glutSolidTorus** (GLdouble innerRadius, GLdouble outerRadius, GLint nSides, GLint nRings);
  - void **glutWireTorus** (GLdouble innerRadius, GLdouble outerRadius, GLint nSides, GLint nRings);
    - 도넛 형태의 입체 도형
      - innerRadius: 내부 원의 반지름
      - OuterRadius: 외부 원의 반지름
      - nSides: 각 radial section에 대한 분할의 개수
      - nRings: radial section의 수

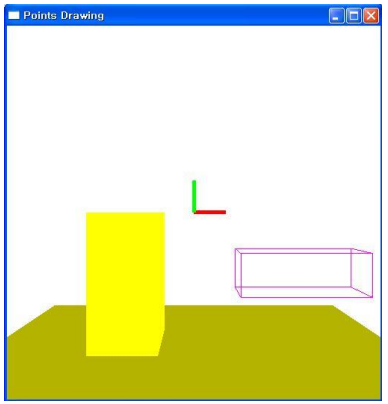


- 솔리드 혹은 와이어프레임 주전자를 그린다.
  - void **glutSolidTeapot** (GLdouble size);
  - void **glutWireTeapot** (GLdouble size);
    - Size: 주전자 반지름의 근사값



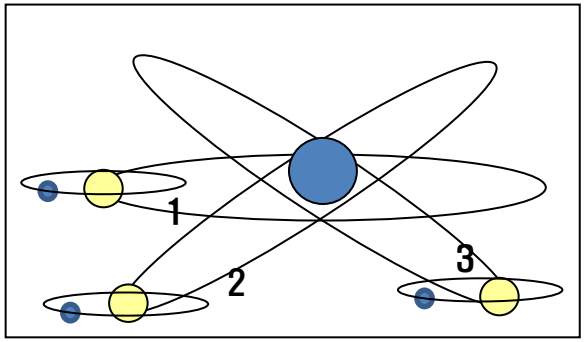
실습 14

- 도형 2개 그리고 변환 하기
  - 화면의 중앙에 좌표계를 그린다. (원근 투영 적용)
  - 화면에 도형을 좌우로 그린다.
    - 메뉴또는 키보드 명령에 의해 도형을 선택하면, 좌측에는 솔리드 모델이, 우측에는 와이어 모델이 그려진다.
    - 선택 모델: 구 / 육면체 / 원뿔 / 주전자
  - 키보드를 이용하여 회전한다.
    - 좌표계에 대한 회전
      - X: x축에 대하여 회전
      - Y: y축에 대하여 회전
      - Z: z축에 대하여 회전
      - 좌표계 전체가 회전하면서 두 개의 도형도 함께 회전한다.
    - 도형 회전
      - L: 왼쪽 도형이 제자리에서 임의의 회전축에 대하여 회전
      - R: 오른쪽 도형이 제자리에서 임의의 회전축에 대하여 회전



실습 15

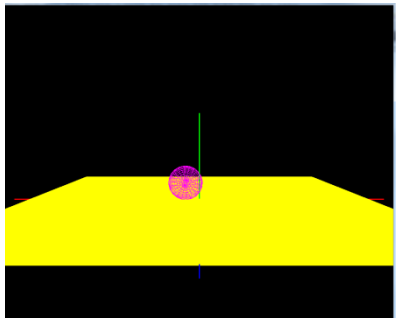
- 중심의 구를 중심으로 3개의 구가 다른 방향의 경로를 따라 회전하는 애니메이션 제작, 각 구에는 그 구를 중심으로 달이 공전한다.
  - 경로 1: xz 평면
  - 경로 2: xz 평면이 반시계방향으로 45도 기울어져 있다.
  - 경로 3: xz 평면이 시계방향으로 45도 기울어져 있다.
  - 3개의 구는 다른 속도로 중심의 구를 공전한다.
  - 3개의 구에는 각각 공전하는 달을 가지고 있다.
  - 메뉴를 이용하여 구의 모델을 선택할 수 있게 한다.
    - 솔리드 모델 / 와이어 모델
  - 원근 투영을 적용한다.
  - 다음의 키보드 명령을 수행한다.
    - x/X: x축 따라 카메라의 위치가 양/음의 방향으로 이동
    - y/Y: y축 기준으로 양/음의 방향으로 중점 이동
    - z/Z: z축 기준으로 양/음의 방향으로 카메라 회전
    - +/-: z축 방향으로 양/음 이동
    - i: 초기화

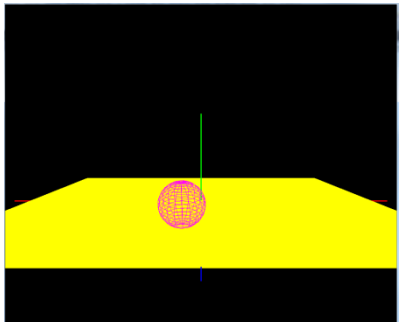


  - 카메라 조정 함수 또는 변환 함수 중 선택하여 사용하기

실습 16

- 구르는 공 그리기
  - 화면에 바닥을 그린다.
  - 중심에 구를 그린다.
  - 구가 회전한다
    - x/X: x축 따라 공이 회전하면서 회전 방향으로 이동한다.
    - y/Y: y축 따라 공이 회전한다.
    - z/Z: z축 따라 공이 회전하면서 회전 방향으로 이동한다.
    - 구가 가장자리에 도달하면 회전만 한다.
  - +/-: z축 방향으로 양/음 이동
  - i: 초기화

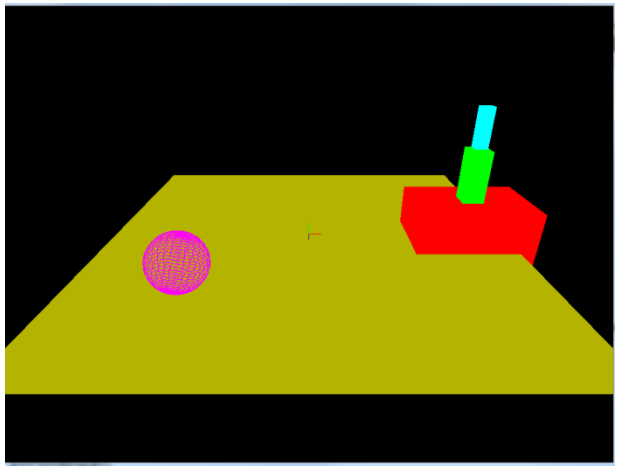


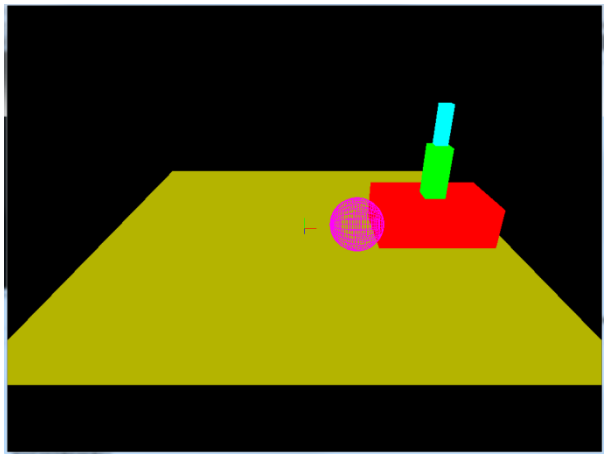


실습 17

실습 18

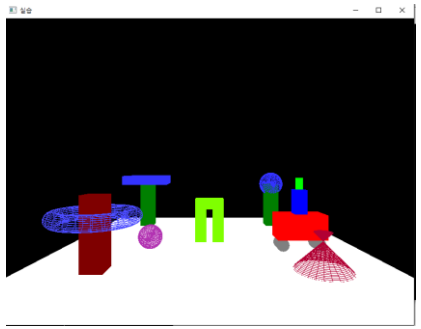
- 실습 16과 17을 융합하기
  - 바닥 위에 크레인과 공을 그린다.
  - 크레인은 임의의 방향으로 자동이동한다.
  - 공은 키보드를 이용하여 바닥위를 좌/우/앞/뒤로 이동한다.
  - 공이 크레인과 부딪치면 둘 다 움직이지 않는다.
  - 카메라를 y축으로 회전하여 볼 수 있다.
    - 전체 화면이 회전한다.
  - 카메라를 z축으로 줌인/줌아웃 하여 볼 수 있다.
  - 보너스: 크레인 아래의 바퀴를 붙여서 회전하게 한다.





실습 19

- 다양한 객체 애니메이션 만들기
  - 바닥을 그리고 바닥의 네 구석과 중앙에 다양한 객체 그린다.
    - 육면체를 이용한 나무
    - 육면체와 구를 이용한 나무
    - 토러스와 육면체를 이용한 건물
    - 두 개의 콘을 이용한 꼭지점 건물
    - 중앙: ㄇ 모양의 건물
  - 객체 애니메이션
    - 육면체를 이용한 나무: 나무 윗 부분이 제자리에서 y축 기준으로 회전
    - 육면체와 구를 이용한 나무: 나무 위 부분이 커졌다 작아졌다 신축
    - 토러스와 육면체를 이용한 건물: 밖의 토러스가 제자리에서 y축 기준으로 위/아래 이동
    - 두 개의 콘을 이용한 꼭지점 건물: 위 아래의 콘이 반대로 신축
    - 중앙: ㄇ 모양의 건물: 문이 열렸다 닫혔다,
      건물 중앙의 y축 기준으로 회전
  - 육면체 외의 객체들은 와이어 객체로 그린다.



실습 20