OpenGL 5

가상현실론

2021/03/15

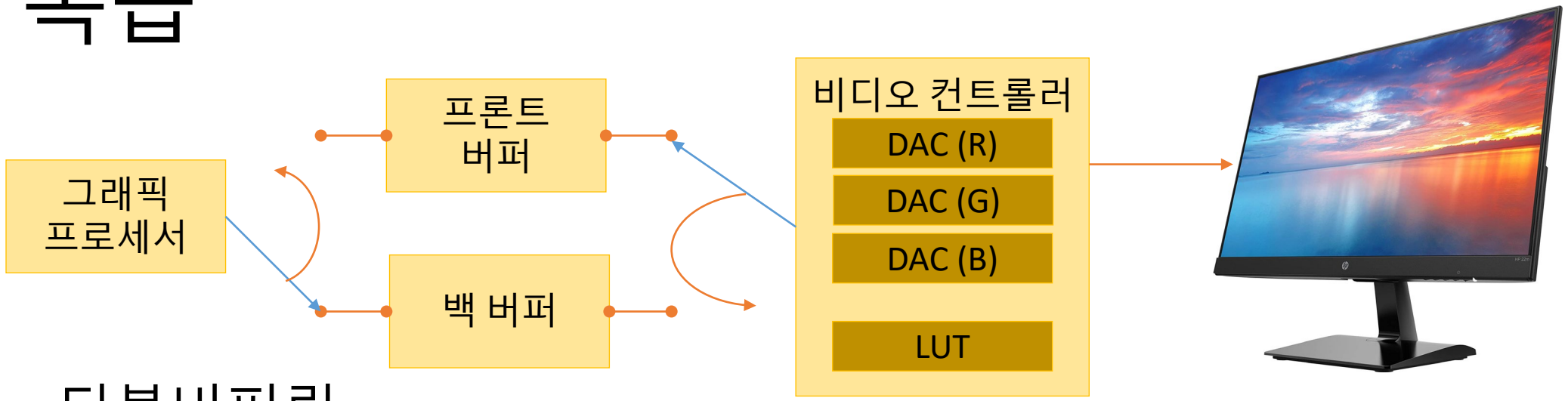
목차

- 복습
  - 콜백프로그래밍
  - 더블버퍼링
  - 모델변환과 시점변환 – 동차좌표, 기하변환
- 모델변환과 시점변환
  - 복합변환
  - 변환의 분류
- GL의 모델변환
  - GL 파이프라인
  - 모델 변환
  - 복합변환에 의한 모델링

복습

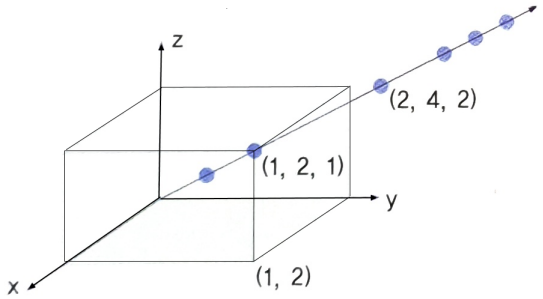
- 콜백프로그래밍
  - 마우스 콜백
    - void glutMouseFunc(void(*func)(int button, int state, int x, int y));
    - void glutMotionFunc(void(*func)(int x, int y));
  - 메뉴 콜백
    - int glutCreateMenu(void(*func)(int value));
    - void glutSetMenu(int id);
    - void glutAddMenuEntry(char *name, int value);
    - void glutAttachMenu(int button);
  - 타이머콜백
    - void glutTimerFunc(unsigned int msecs, void(*func)(int value), int value);

복습



- 더블버퍼링
  - void glutInitDisplayMode(unsigned int mode);
  - void glutSwapBuffers();
- 디스플레이 리스트
  - GLuint glGenLists(GLsizei range);
  - glNewList(GLuint list GLenum mode);

복습



- 모델변환과 시점변환 – 동차좌표
  - 동차좌표는 3차원 요소(점, 벡터)를 4차원으로 표현

$$v = 4V_1 + 2V_2 + V_3 + 0 \cdot r$$
$$P = 4V_1 + 2V_2 + V_3 + 1 \cdot r$$

  - 마지막 요소가 0이면 벡터를 1이면 점으로 표현
  - 3차원 동차좌표를 4차원 (x, y, z, w)로 표현하면 3차원 실제 좌표는 (x/w, y/w, z/w)

복습

- 기하변환 – 이동, 회전, 크기조절 등
- 이동

$$P'=T+P$$

$$\begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

- 회전

$$P'=R\cdot P$$

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

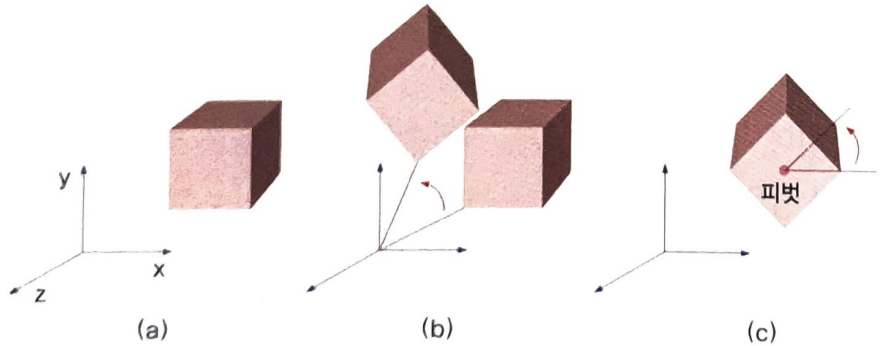
- 크기조절

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- 전단

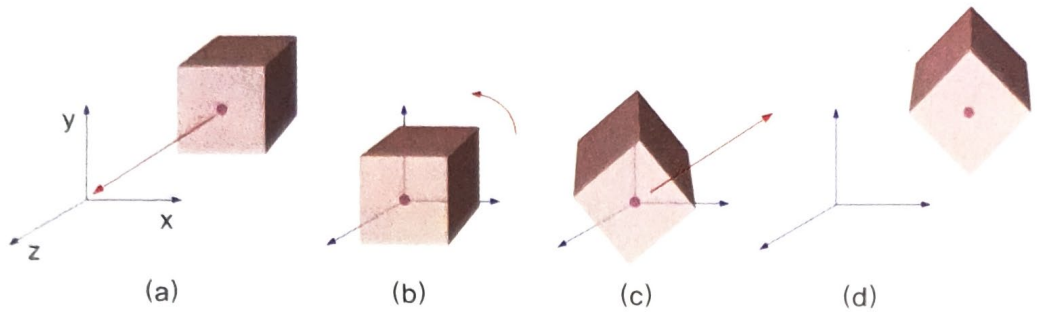
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & Sh_y & 0 & 0 \\ Sh_x & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

복합변환 (Composite Transformation)



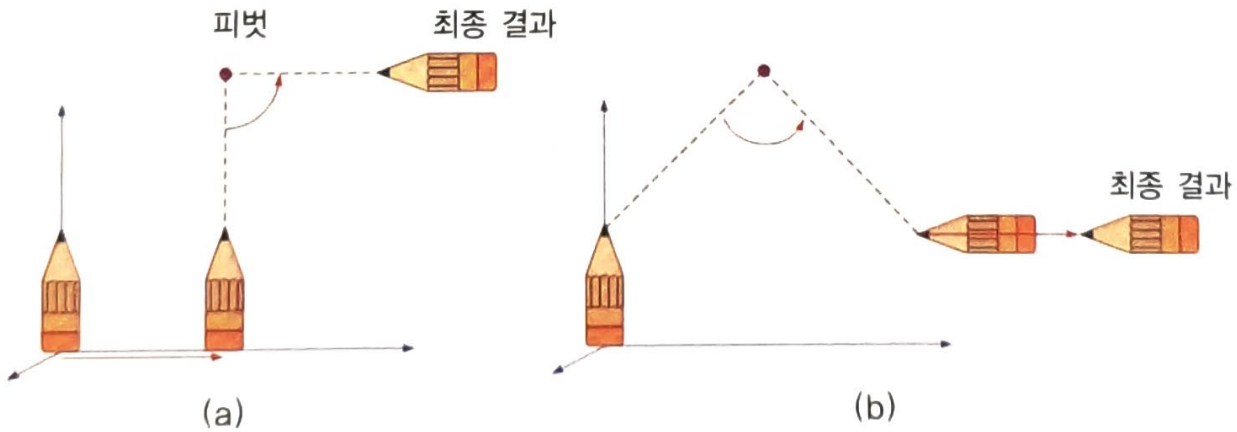
- z축 중심의 회전(b)와 피벗 중심의 회전(c)는 다른 결과를 가져옴
- 피벗 회전은 아래와 같은 3가지 변환이 복합되어 실행
  1. 피벗이 좌표계 원점에 일치하도록 물체를 이동
  2. 물체를 원점 중심으로 기준 축 주위로 회전
  3. 회전된 물체를 1에서 이동한 방향의 반대 방향으로 이동

복합변환-중심축 기준 회전



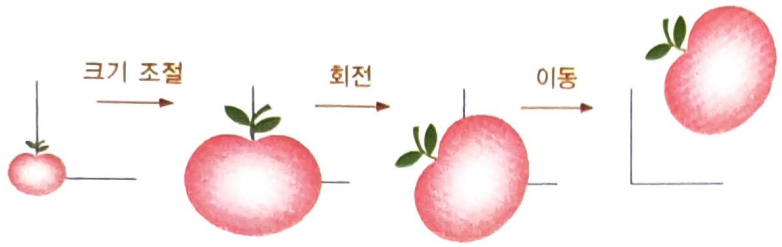
- (a)-(b): 물체의 중심을 원점에 일치시킴
- (b)-(c): z 축 중심 회전
- (c)-(d): 물체의 중심점으로 다시 이동
- $\mathrm{C} = (\mathrm{X_p}, \mathrm{Y_p}, \mathrm{Z_p}) \cdot \mathrm{R_z}(\theta) \cdot (-\mathrm{X_p}, -\mathrm{Y_p}, -\mathrm{Z_p})$

복합변화



- 행렬의 곱셈에 교환법칙은 성립하지 않음에 유의 ($A \cdot B \neq B \cdot A$)
- (a)는 이동 후 회전을 가한 것이며 (b)는 회전 후 이동을 가한 것. (a)와 (b)의 경우 회전을 위한 피벗의 위치는 동일하나 결과는 다르다는 것에 유의

복합변환



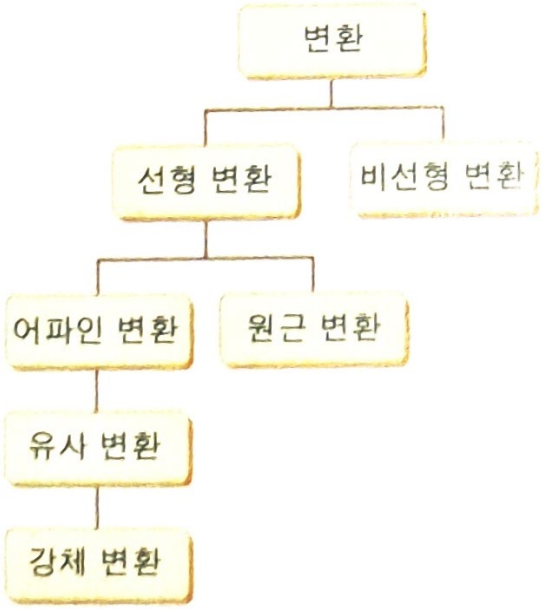
[그림 6-31] 크기 조절, 회전, 이동

- 변환된 물체 하나를 원래 물체의 인스턴스(instance)라고 함
- 물체 인스턴스는 크기 조절, 회전 이동 순으로 변환을 가함
- 크기 조절에이나 회전을 먼저 가하는 이유는 이 상태에서 물체 중심이 피벗과 원점이 일치한 상태이므로 이를 일치시키기 위한 변환이 불필요하기 때문.
- C=T·R·S

변환의 분류

- 강체 변환(Rigid Body Transformation)
  - 이동변환+ 회전변환
  - 변환 전후에 내부 정점 간의 거리가 그대로 유지됨
- 유사변환 (Similarity Transformation)
  - 강체변환+균등 크기 조절 변환+반사 변환
  - 변환 전후에 물체면 사이의 각이 유지되고, 물체 내부 정점 간의 거리도 일정한 비율로 유지됨 (크기 조절 변환 때문에 거리 자체가 유지되는 것은 아니라는 데 유의)
- 어파인 변환(Affine Transformation)
  - 유사변환+차등 크기 조절 변환+전단 변환
  - 공학이나 자연과학 문제 해결에 사용되는 거의 모든 변환
  - 물체의 타입이 유지 (직선-> 직선, 다각형-> 다각형, 곡면 -> 곡면)
  - 물체 내부의 평행한 선분이 변환 후에도 평행하게 유지
  - 행렬의 마지막 행이 항상 (0, 0, 0, 1)

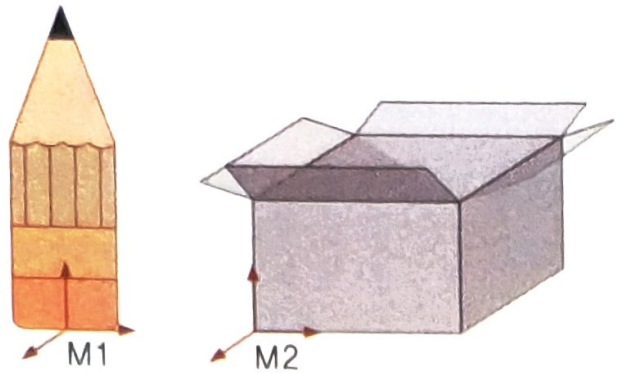
변환의 분류



- 원근 변환 (Perspective Transformation)
  - 직선이 직선으로 변환된다는 정도의 속성만 유지
  - 변환 행렬의 마지막 행이 (0, 0, 0, 1)이 아님
- 선형 변환(Linear Transformation)
  - 어파인 변환+원근 변환
  - 선형 조합(linear combination)으로 표시되는 변환

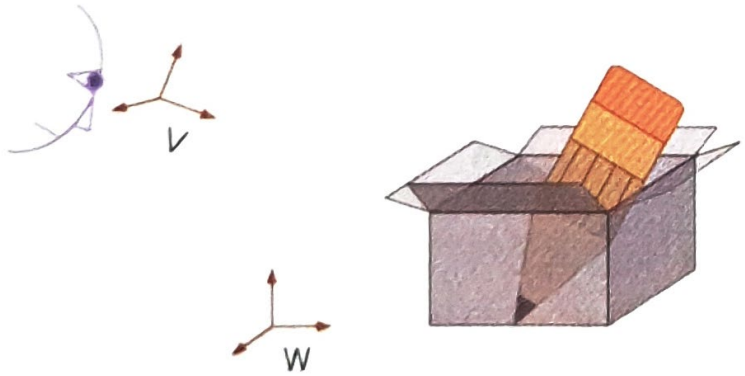
GL의 모델 변환

- 물체 정점은 물체 하나를 설계할 때의 좌표계, 한 장면에 여러 물체를 모아놓았을 때의 좌표계, 그 장면을 바라보는 시점에 따른 좌표계 등의 좌표계를 거치면서 새로운 좌표값으로 바뀌며 최종적으로 화면에 그려짐.
- 그래픽에서 모델링은 물체 정점을 결정하는 작업. 정점을 조함하여 다각형 메쉬를 만들고, 다각형 메쉬를 조합하여 물체를 만들고, 다시 물체를 조합하여 장면(Scene)을 구성할 수 있기 때문.



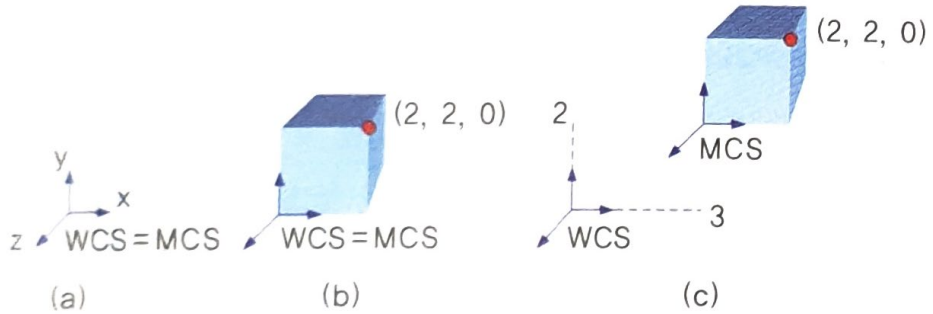
- 물체 좌표계(MCS: Modelling Coordinate System)/시역 좌표계(LCS: Local Coordinate System): 각 물체별로 설계상의 편의를 위주로 설정된 좌표계

GL의 모델 변환



- 연필과 박스가 각각 M1, M2의 지역좌표계를 가지고 있기 때문에 모든 물체를 한꺼번에 아우를 수 있는 기준 좌표계가 필요. 이를 전역 좌표계 (WCS: World Coordinate System)이라고 함.
- 그림에서는 W를 전역좌표계로 설정. 전역 좌표계가 설정되면 각 모델 좌표계를 기준으로 표시된 물체 좌표가 전역 좌표계를 기준으로 바뀌어 물체 간의 상대적인 위치가 명확해짐.
- 화면에 보이는 것은 시점(Viewpoint)가 좌우함. 사용자의 시점을 기준으로하는 시점 좌표계(VCS: View Coordinate System)으로 WCS가 다시 변환됨 (그림에서 V).

전역좌표계의 설정

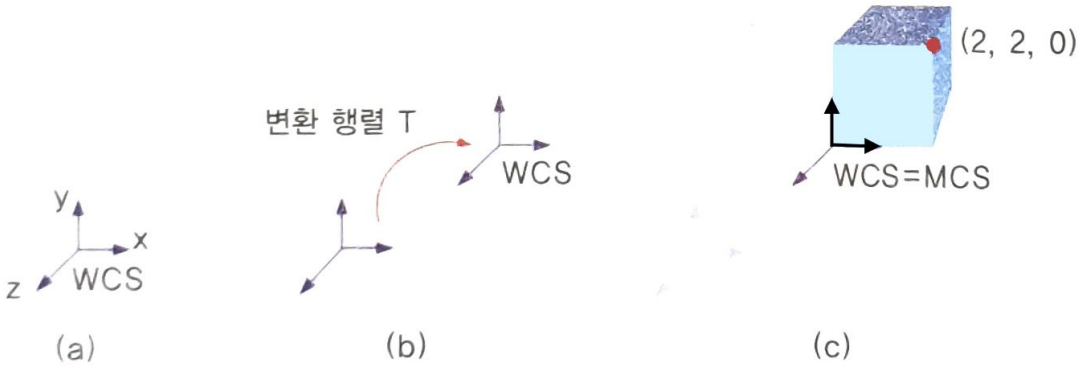


$$P' = TP = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 2 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 0 \\ 1 \end{bmatrix}$$

- GL 프로그램이 처음 실행될 때-(a)와 같이 모델좌표계와 전역좌표계가 일치. 2x2x2 크기의 정육면체 모델의 오른쪽 모서리(적색 구)의 정점 좌표는 p=(2, 2, 0)로 설정.
- 정육면체 모델에 이동변환(translation)이 ΔT=(3, 2, 0)만큼 가해질 경우 전역좌표계는 모델좌표계와 분리. 정육면체 오른쪽 모서리의 모델좌표계는 변화가 없으나 전역좌표계에서는 ΔT+p=(5, 4, 0)으로 변화.
- 이를 homogeneous transformation으로 표현하면 좌측과 같음

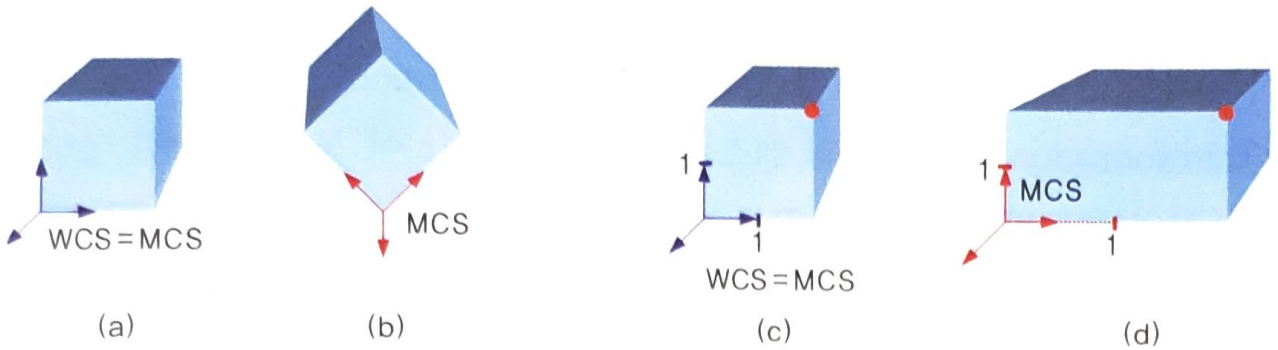
모델 좌표계에서 전역 좌표계로 변환

$$P' = TP = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 2 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 0 \\ 1 \end{bmatrix}$$



- GL의 입장에서는 물체의 이동보다 '좌표계의 이동'으로 바라봄. 따라서 전역좌표계를 (3, 2, 0)만큼 이동시 GL은 "전역 좌표계를 모델 좌표계로 일치시키기 위한 것이 변환 행렬임 " 으로 인식.
- 그림의 예에서 GL은 (a)의 전역좌표계에 변환 행렬 T를 가하여 (b)로 모델 좌표계로 일치시킴. 모델 좌표계에서 정점을 그려내면 변환된 위치에 육면체가 그리쳔다는 것.
- 따라서 GL 입장에서 볼 때 물체에 변환을 가한다는 것은 결국 전역 좌표계를 모델 좌표계와 일치시키기 위해서 변환을 가한다는 것.

모델 좌표계에서 전역 좌표계로 변환

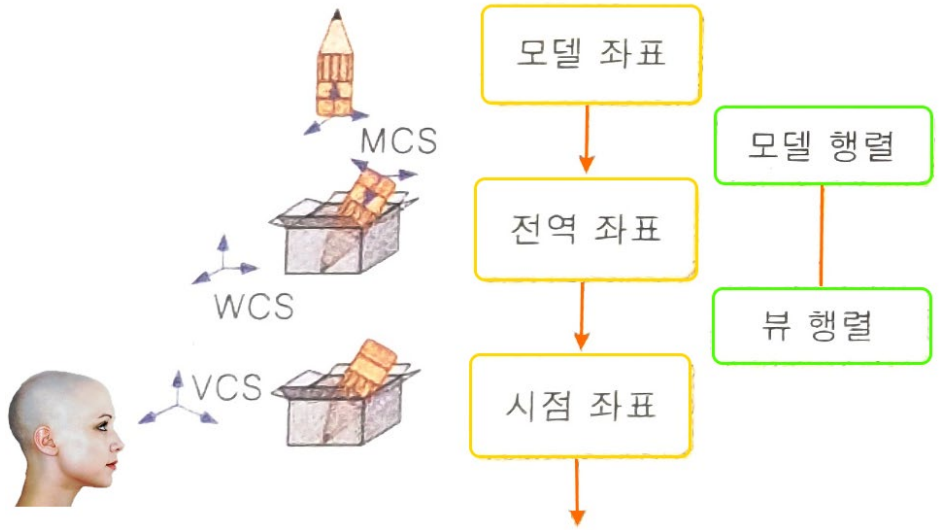


- (b): (a)의 물체를 원점 중심으로 45도 회전. 모델 좌표계 기준 물체 좌표계는 변함이 없음
- (c)의 우상단 점의 좌표 ᄀᆡ는 (2, 2, 0). WCS=MCS 상태이기 때문에 모델좌표계와 전역 좌표가 서로 일치
- (d)처럼 변형이 일어날 경우 x축의 길이에서 WCS와 MCS에서 차이가 생김. MCS에서는 (2, 2, 0)인 좌상단 점이 WCS에서는 (4, 2, 0)이 됨.
- GL 입장에서 보았을 때 물체에 가해지는 모든 변환을 전역 좌표계를 모델좌표계와 일치시키기 위한 변환. 물체 이동은 원점의 이동, 물체 회전은 축 방향의 회전, 쿨체 크기 조절은 축 눈금 길이의 조절. 각각의 변환을 위한 변환행렬은 전역 좌표계를 변화된 모델 좌표계로 일치시키기 위한 행렬로 사용.

GL 파이프라인

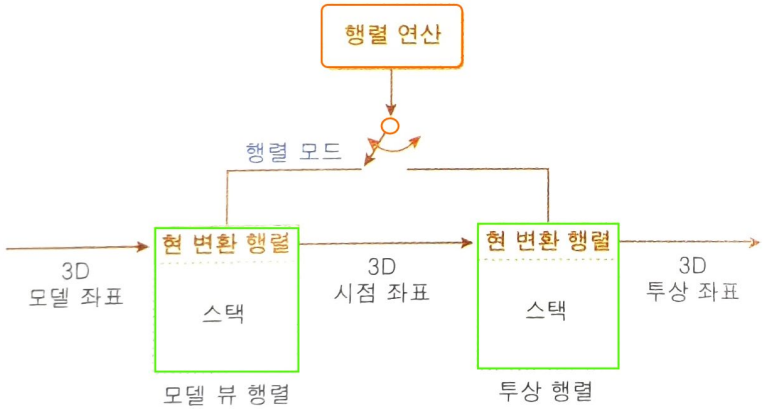
- 모델 변환 (Modelling Transformation): 설계된 물체에 대해 이동, 회전, 크기 조절 등의 기하 변환
- 시점 변환 (Viewpoint Transformation): 변환된 물체를 관찰하기 위해 카메라를 이동하고 회전함으로써 카메라 위치와 방향을 설정하는 작업. 시점 변환을 뷰 변환 (View Transformation)이라고도 함.
- 투상 변환 (Projection Transformation): 카메라의 렌즈를 선택하고 촬영하여 물체의 2차원 영상을 필름에 맺히게 하는 작업.
- 뷰포트 변환 (Viewport Transformation): 찍힌 사진의 크기를 줄이거나 늘리는 작업.
- 모델 뷰 변환 (Model-View Transformation): 모델변환 + 뷰 변환

GL 파이프라인



- GL의 모든 변환은 변환 행렬 (Transformation matrix)로 대변됨. 모델 변환은 모델 행렬, 시점 변환은 뷰 행렬, 투상 변환은 투상 행렬로 대변됨. 즉, 물체에 변환을 가한다는 것은 물체 좌표에 해당 행렬을 곱하는 것.
- GL에서는 모델 변환과 뷰 변환을 합하여 모델 뷰 변환으로 정의했기 때문에 행렬 역시 모델 뷰 행렬 (ModelView matrix) 하나로 취급. 원래의 모델 좌표로 표시된 물체 좌표에 모델 뷰 행렬 곱하면 그 결과는 시점 기준의 좌표가 됨.

모델 변환



- GL의 행렬 모드(Matrix Mode)설정
  - 변환을 위해서는 먼저 변환의 종류(모델 뷰 변환, 투상 변환, 텍스쳐 변환 등)를 먼저 명시함. 변환의 종류별로 별도의 행렬이 존재하므로, 모델 뷰 행렬, 투상 행렬, 텍스쳐 행렬 중 하나를 선택하는 것.
- 그림에서 보듯이 행렬 모드를 설정하는 것은 일종의 스위칭(switching)임. 이 스위치가 모델 뷰 행렬에 붙기도 하고 투상 행렬에 붙기도 함.

모델 변환

- void glMatrixMode(GLenum mode);
  - 행렬모드를 설정하는 함수 프로토타입
  - mode에 들어가는 상수는 GL_MODELVIEW, GL_PROJECTION, GL_TEXTURE 등임
- GL에서는 상태 변수를 사용하기 때문에 현재의 상태 변수 값이 중요함. 모든 파이프라인 프로세스가 현 상태 변수를 기준으로 하기 때문. 현 변환 행렬 (CTM: Current Transformation Matrix)이 중요
- void glLoadIdentity();
  - 현 변환행렬을 항등 행렬(identity matrix)로 초기화 함.
  - $CTM = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

모델 변환

- 변환 후의 정점 P’와 변환 전의 정점 P의 관계
  - $P' = CTM \cdot P$
  - 프로그램이 시작할 때는 CTM=I이므로 기본적으로 모델 좌표계=전역 좌표계= 시점좌표계
- void glLoadMatrixf(const GLfloat *M);
  - 배열 M에 있는 행렬 요소 값을 현 변환 행렬로 올림.
  - GL은 모든 행렬이 열 우선 순위(Column Major Order)로 저장됨에 주의
  - $M = \begin{bmatrix} a & e & i & m \\ b & f & j & n \\ c & g & k & o \\ d & h & l & p \end{bmatrix}$
  - 즉, 위 행렬을 배열에 저장할 때는 GLfloat M[16]={a, b, c,..., p}의 순서로 저장해야 함.

모델 변환

- void glMultMatrixf(const GLfloat *M);
  - glMultMatrixf(M)은 현 변환 행렬에 배열 M에 있는 행렬을 곱함. 이 곱셈은 후위 곱셈(post multiplication)으로 식의 오른쪽에 M을 곱하는 것에 유의함.
  - CTM=CTM·M
- 행렬대신 기하 변환 작업 자체를 명시할 수 있음
- void glTranslatef(GL float dx, GLfloat dy, GLfloat dz);
  - 모델 좌표계를 전역 좌표계로부터 x, y, z 축 방향으로 dx, dy, dz만큼 이동시팀
- void glScalef(GLfloat sx, GLfloat sy, GLfloat sz);
  - 모델좌표계 x, y, z 축의 눈금이 전역 좌표계 눈금의 sx, sy, sz배가 되도록 크기조절
- void glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z);
  - 모델좌표계를 전역 좌표계로부터 변수 angle에 제시된 각도만큼 벡터(x y, z)를 축으로 반시계 방향으로 회전. 각도는 degree임에 유의.

모델 변환

- 모든 변환은 전역 좌표계를 기준으로 하는 모델 좌표계의 변환
- 현 변환 행렬의 오른쪽에 해당 변환 행렬이 곱해짐.
  - Ex) 현 변환 행렬이 identity matrix로 초기화된 직후 glTranslatef(1, 2, 0)이 호출
  - $\mathrm{CMT} = \mathrm{CMT} \cdot \mathrm{T} = \mathrm{I} \cdot \mathrm{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
  - 기준 물체의 정점좌표 P를 위의 식 우변에 곱하면 이동 변환의 결과 전역 좌표 P'가 나오게 됨
  - $\mathrm{P}' = \mathrm{I} \cdot \mathrm{T} \cdot \mathrm{P} = \mathrm{T} \cdot \mathrm{P}$

복합 변환에 의한 모델링

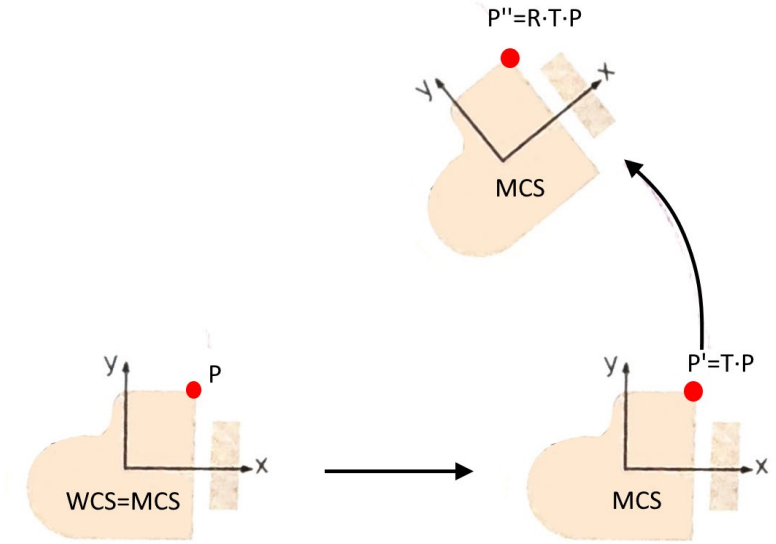
- example)

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glScalef(sx, sy, sz);
glRotatef(theta, vx, vy, vz);
glBegin(GL_POINTS);
  glVertexef(px, py, pz);
glEnd();
```

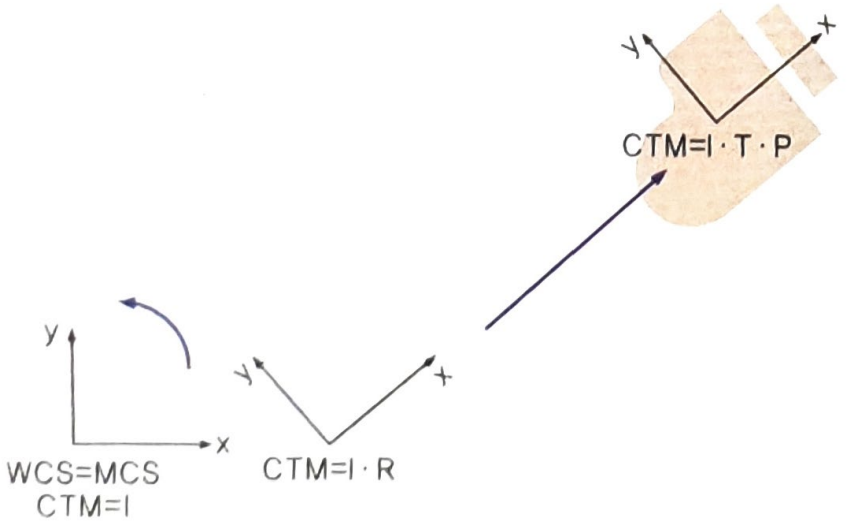
함수 종류	GL 함수	matrix computation
matrix initialization	glLoadIdenty();	CTM = I
scale	glScalef();	CTM = CTM·S=I·S
rotation	glRotatef();	CTM=CTM·R=I·S·R
point declaration	glVertex3f();	P'=CTM·P=I·S·R·P

- GL의 정점좌표는 열행렬(column)으로 표시됨. 전위 곱셈(pre-multiplication) 규칙으로 계산됨에 유의
- 연산순서와 코드 나열 순서와는 반대인 것에 유의.

복합 변환에 의한 모델링

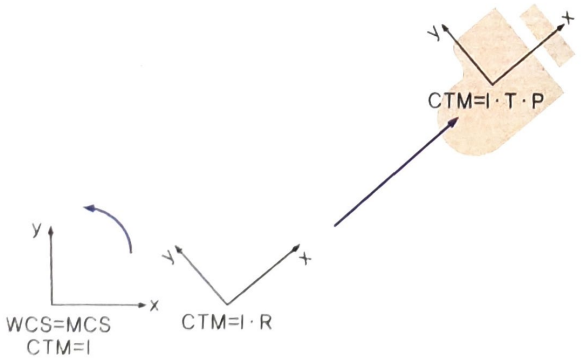


- 전역좌표계 기준으로 물체 이동
- Pre-multiplication
- 물체 자체를 이동시킨 후 회전
  - P’=T·P
  - P’’=R·P’=R·T·P



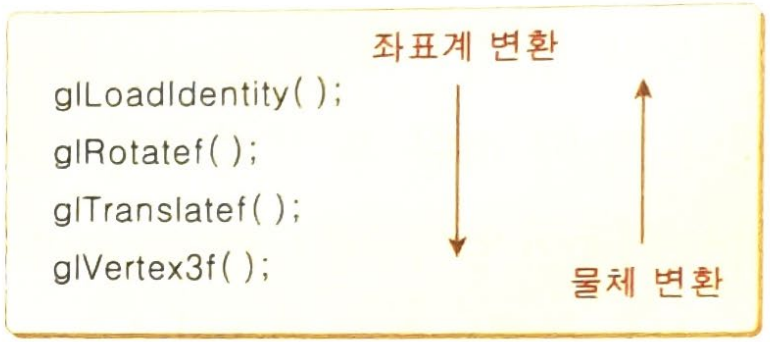
- 모델 좌표계 자체를 이동
- Post-mulitiplication
  - CTM=I·R
  - CTM= I· R·T
  - P’’=I· R·T·P= R·T·P

복합 변환에 의한 모델링



- 변환행렬로 표현
  - glMatrixMode(GL_MODELVIEW);
  - glLoadIdentity();
  - glRotatef(45, 0.0, 0.0, 1.0);
  - glTranslatef(10.0, 0.0, 0.0);
  - glVertex3f(Px, Py, Pz);

복합 변화에 의한 모델링



- 물체변환 (Object Transformation)
  - 고정된 전역 좌표계를 기준으로 물체를 연속적으로 움직이는 방법. 물체가 전역 좌표계에 원점에 이미 그려져 있다고 가정하고 전역 좌표계 기준으로 물체를 이동, 회전, 크기조절하여 원하는 위치와 방향으로 이동
- 좌표계 변환 (Coordinate Trnaformation)
  - 모델 좌표계를 연속적으로 움직이는 방법. 물체가 아직 그려지지 않았다고 가정하고 대신 모델 좌표계를 연속적으로 움직여 원하는 위치와 방향으로 가져다 놓은 다음, 최종적인 모델 좌표계를 기준으로 물체를 그림. GL의 명령어 순서는 이 방법을 기준으로 함.
  - 일반적으로는 물체에 대한 변환으로 파악하는 것이 수월함.

예제

```
void display();
void myInit();
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_RGB);
   glutInitWindowSize(300, 300);
   glutInitWindowPosition(300, 300);
   glutCreateWindow("OpenGL sample drawing");
   glutDisplayFunc(display);
   glutKeyboardFunc(keyboard);
   myInit();
   glutMainLoop();
   return 0;
}
void myInit()
{
   glClearColor(1.0, 1.0, 1.0, 1.0);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);
}
```

```
void display()
{
   glClear(GL_COLOR_BUFFER_BIT);
   glViewport(0, 0, 300, 300);
   glColor3f(1.0, 0.0, 0.0);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   glRotatef(45.0, 0.0, 0.0, 1.0);
   glTranslatef(0.6, 0.0, 0.0);
   glutSolidCube(0.3);
   glFlush();
}
```