05 CHAPTER

# 모델변환

## 5장. 학습 내용

- 3D 그래픽 파이프라인과 좌표계
- 모델 좌표계: 로봇 부품들을 만들자.
- 모델 변환: 로봇의 합체
- OpenGL의 모델 변환
- 행렬 스택
- 모델 변환 응용: 로봇을 조립하자.

2

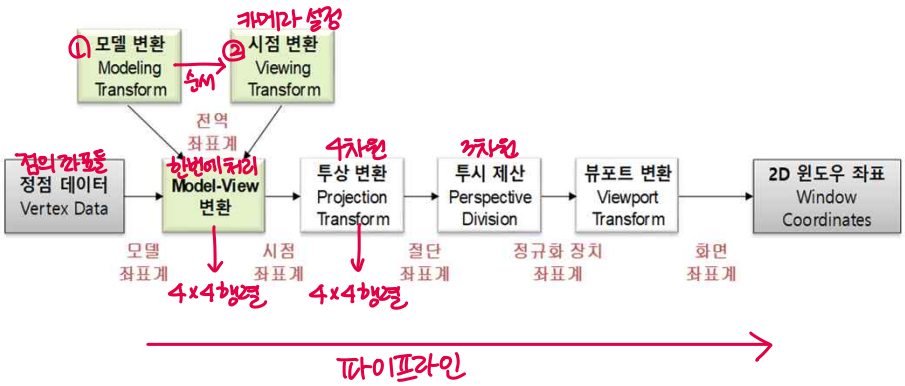
# 5.1 3D 그래픽 파이프라인과 좌표계

- 3D 그래픽 파이프라인의 이해
- 주요 변환들

3

# 3D 그래픽 파이프라인의 이해

- 3차원 그래픽 파이프라인

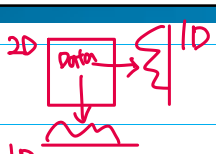



4

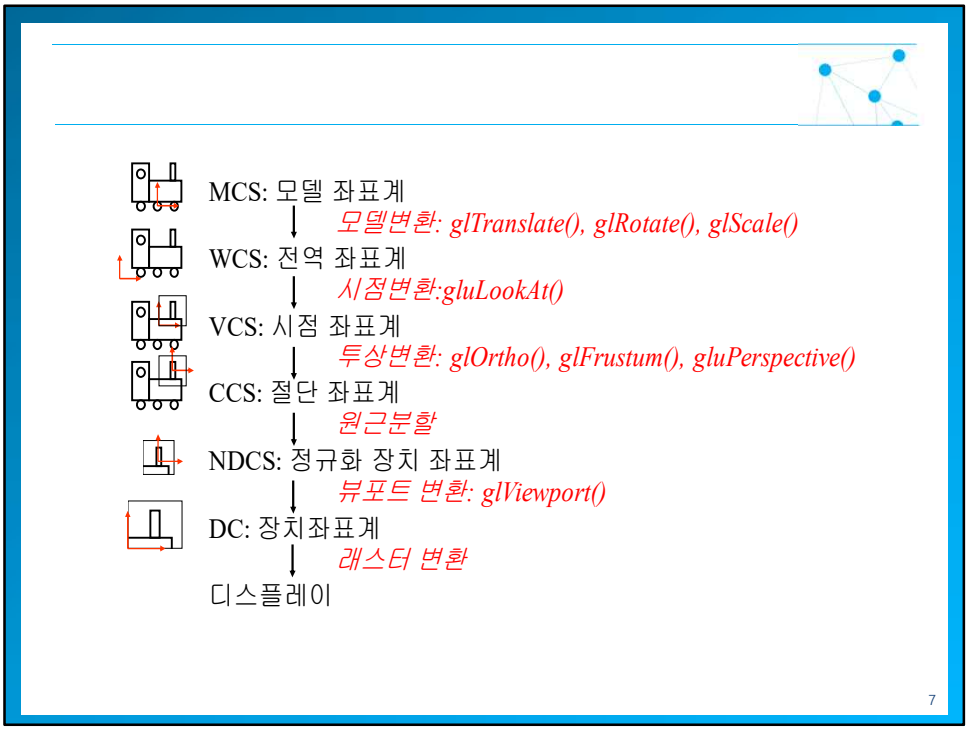
## 주요 변환들

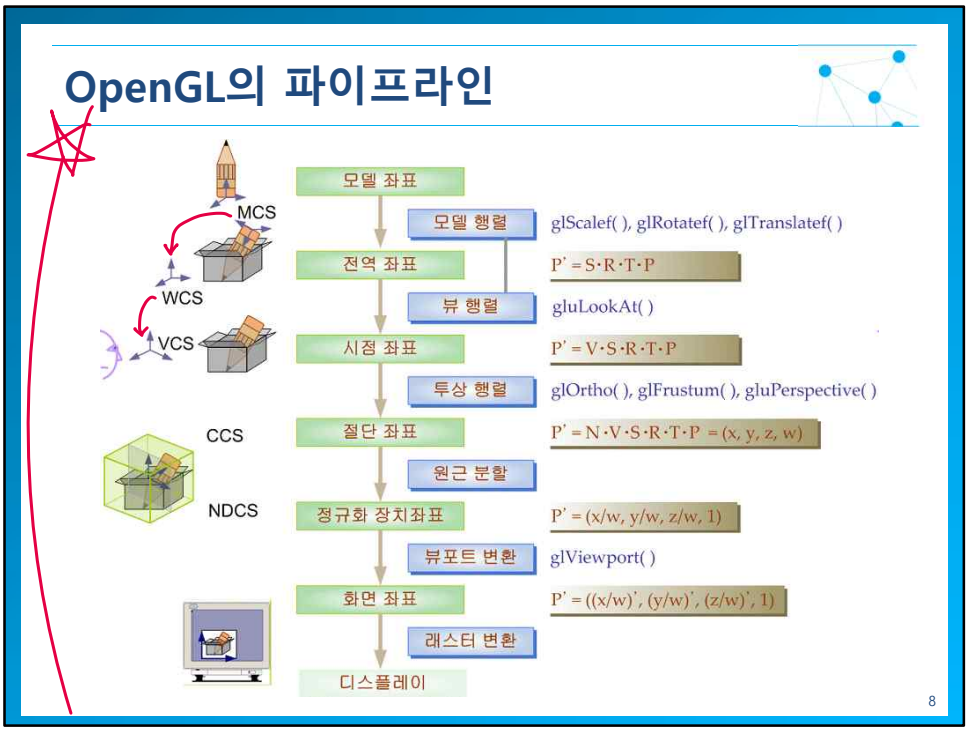
- 모델 변환(Modeling Transformation)
  - 처음에는 각 모델의 모든 정점들은 각자의 모델 좌표계로 정의
  - 이들을 한 자리에 모아 하나의 장면(scene)을 만드는 작업
  - 이를 위해 하나의 통합된 좌표계, 즉 전역 좌표계(WCS)가 필요
  - 변환 후 장면을 구성하는 모든 부품의 정점들은 전역 좌표계를 기준으로 변환됨.
- 시점 변환(Viewing Transformation)
  - 다음으로 카메라를 설정
  - 이를 통해 모든 정점들은 카메라 중심의 시점 좌표계(VCS)로 변환된다.






- 투상 변환(Projection Transformation)과 투시 제산
  - 시점 좌표계에서 가시 부피(View Volume)를 지정하고 이를 정규화 가시부피로 변환하는 과정
  - 모든 정점들을 정규화 장치좌표계(NDS, Normalized Device Coordinate)를 기준으로 변환.
- 뷰포트 변환(Viewport Transformation)
  - 드디어 정점들을 화면상의 화소 위치로 변환
  - 이후 렌더링 과정을 거쳐 화면으로 출력.


MCS: 모델 좌표계
모델변환: glTranslate(), glRotate(), glScale()
WCS: 전역 좌표계
시점변환:gluLookAt()
VCS: 시점 좌표계
투상변환: glOrtho(), glFrustum(), gluPerspective()
CCS: 절단 좌표계
원근분할
NDCS: 정규화 장치 좌표계
뷰포트 변환: glViewport()
DC: 장치좌표계
래스터 변환
디스플레이
7


OpenGL의 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
8

## 5.2 모델 좌표계: 로봇 부품들을 만들자.

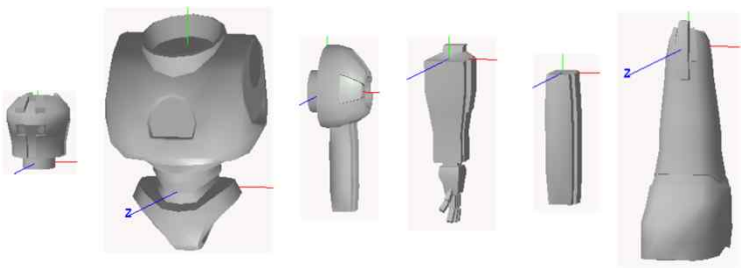
- 최종 목표: 로봇 부품을 선택해 화면에 출력하는 프로그램 작성

- 모델 좌표계
- 3차원 모델 저장 파일 형식
- Lab: 3DS Max에서 내보내기(ASE)
- 3차원 모델을 위한 자료구조
- Lab: ASE 출력 프로그램

9

## 모델 좌표계

- 로봇 부품들
  - 3DS Max등의 툴 사용해서 만듦
  - 파일로 저장(export)
  - 각각 자신의 좌표계로 표시됨 → 모델 좌표계



10

# 3D 모델 저장 파일 형식

- 3DS Max와 같은 툴에서 편집한 후 다양한 형식으로 Export 가능, 서로 변환도 가능.
  - obj, ase, vrml, …
- OBJ 파일
  - 비교적 간단한 형식
  - 정점, 면 및 법선의 개수를 나타내는 부분이 리스트 뒤쪽에 저장
  - 저장 공간을 미리 할당하기 위해서는 불편

```
# 3ds Max Wavefront OBJ Exporter v0.97b - (c)2007 guruware
# File Created: 14.10.2010 10:39:26
# object Box01
v -10414.8359 -10117.6094 0.0000
....
v -10414.8359 -6921.1289 2028.5356
# 8 vertices
vn 0.0000 0.0000 -1.0000
...
vn -1.0000 0.0000 0.0000
# 6 vertex normals
g Box01
f 1//1 2//1 3//1
...
f 5//6 8//6 2//6
# 12 faces
```

12개 면 (삼각형 두개가 한 면)



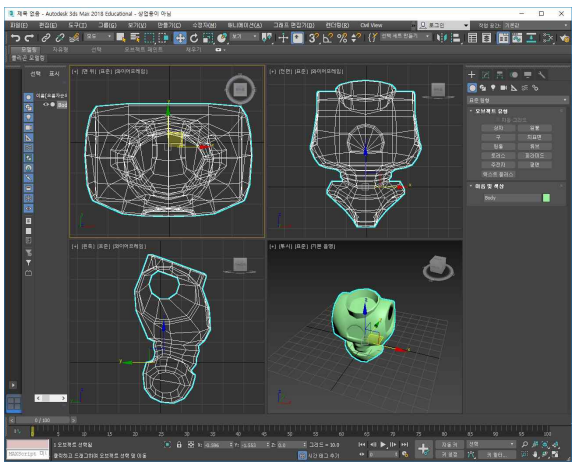
- ASE 파일: ASE(ASCII EXPORT) 파일은 OBJ 보다는 약간 복잡하게 출력
  - *SCENE부분에서는 애니메이션 정보
  - *MATERIAL_LIST 부분에서는 재질정보
  - *GEOMOBJECT 부분에서는 메쉬의 정점, 면, 법선, 조명, 색상 등의 물체 정보
  - 약간은 복잡하지만 정점이나 면의 개수가 앞쪽에 있어 이를 이용해 미리 정점, 면, 법선 리스트를 동적으로 할당할 수 있다.
- 3DS Max에서 내보내기(export)한 ASE파일의 예

```
*3DSMAX_ASCIIEXPORT 200
*COMMENT "AsciiExport 버전 2.00 - Thu Nov 29 16:07:49 2018"
*SCENE { ... }
*GEOMOBJECT {
*NODE_NAME "Body"
...
*MESH {
    *TIMEVALUE 0
    *MESH_NUMVERTEX 343
    *MESH_NUMFACES 666
    *MESH_VERTEX_LIST {
     *MESH_VERTEX 0 -0.3881 2.8345 10.3895
     ...
     *MESH_VERTEX 342 -24.0591 4.3258 33.2297
    }
    *MESH_FACE_LIST {
     *MESH_FACE 0: A: 79 B: 78 C: 2 AB: 1 BC: 1 CA: 0
     ...
     *MESH_FACE 665: A: 188 B: 341 C: 190 AB: 1 BC: 1 CA: 1
    }
    *MESH_NORMALS {
     *MESH_FACENORMAL 0 –0.4167 0.8495 0.3236
     *MESH_VERTEXNORMAL 79 –0.3703 0.9098 0.1873
     *MESH_VERTEXNORMAL 78 –0.5542 0.5404 0.6331
     *MESH_VERTEXNORMAL 2 -0.4979 0.7379 0.4556
     ...
    }
}
*PROP_MOTIONBLUR 0
*PROP_CASTSHADOW 1
*PROP_RECVSHADOW 1
}
```

표면정보
꼭짓점수
표면 수
면의 방향 (렌더링 잘 나오기위해 중요)

13

# Lab: 3DS Max에서 내보내기(ASE)



14

- ASE 파일로 내보내기(Export)
  - 표면의 **법선(normal)** 정보가 저장되도록 해야 함.
  - '메시 옵션->메시 법선'을 체크

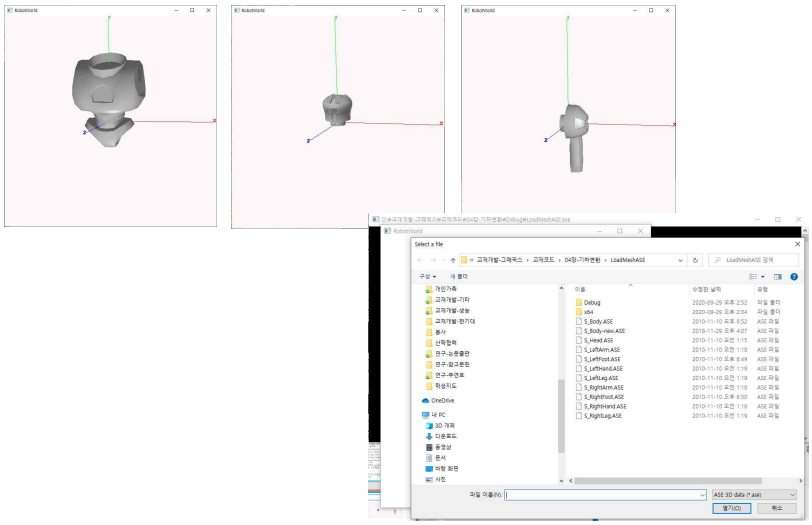



# 모델 저장을 위한 자료구조

- 저장된 모델 파일을 읽어 들여야 함 → 자료구조 필요
- 필요한 기본 데이터
  - Vertex : 3차원 좌표상의 한 점을 표시한다.
  - Normal : 면이나 정점의 법선 벡터를 나타낸다.
  - Face: 하나의 면을 나타낸다. 면은 삼각형으로 한정한다.
  - Mesh: 하나의 표면 모델을 나타낸다.
    - 다수의 Vertex, Normal 및 Face 정보를 포함
    - 클래스로 구현
    - 화면 출력 함수 draw()와 3D 모델 파일 입력 함수 포함.
- Mesh.h → 각 구조체와 클래스 구현

- 실습: ASE 형식의 모델을 읽어 화면에 출력하는 프로그램을 구현해 보자.

# Lab: ASE 출력 프로그램



17

# Mesh.h : Vertex, Face, Normal

```
#pragma once
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <gl/glut.h>
#include "glk.h"
using namespace std;

struct Vertex {          // Vertex: x, y, z 의 위치값
    float x, y, z;
};
struct Face {            // Face: vertex 3개의 인덱스
    int vi[3];
};
struct Normal {          // Normal: face와 세 정점의 법선벡터
    float norFace[3];
    float norV1[3], norV2[3], norV3[3];
};
```

18

## Mesh 클래스

```
class Mesh {                    // Mesh 클래스
    int      nVtx=0;            // Vertex 개수
    Vertex* vertex = NULL;      // Vertex 리스트
    int      nFace=0;           // Face 개수
    Face*    face = NULL;       // Face 리스트
    Normal* normal = NULL;      // Normal 리스트

public:
    Mesh() { }
    ~Mesh() { clearAse(); }
    void clearAse() {// -> vertex, face, normal 리스트의 동적할당 해제하는
        if (nVtx != 0)
            delete[] vertex;
        if (nFace != 0) {
            delete[] face;
            delete[] normal;
        }
        nVtx = nFace = 0;
    }
    void readAse(const char* fileName) { ... }
    void draw(float scale = 1.0f, bool bCoord = false) { ... }
};
```



## Mesh::draw()

```
void draw(float scale = 1.0f, bool bCoord = false) {
    glBegin(GL_TRIANGLES);
    for (int i = 0; i < nFace; i++) {
        Vertex* v1 = &vertex[face[i].vi[0]];
        Vertex* v2 = &vertex[face[i].vi[1]];
        Vertex* v3 = &vertex[face[i].vi[2]];
        glNormal3fv(normal[i].norV1);
        glVertex3f(v1->x/scale, v1->y/scale, v1->z/scale);
        glNormal3fv(normal[i].norV2);
        glVertex3f(v2->x/scale, v2->y/scale, v2->z/scale);
        glNormal3fv(normal[i].norV3);
        glVertex3f(v3->x/scale, v3->y/scale, v3->z/scale);
    }
    glEnd();
    if (bCoord)
        glkDrawCoord(1.0);
}
```

## Mesh::readAse()

```
void readAse(const char* fileName) { // ASE file load
    FILE* fp;
    char line[256], str[40];    // 한 라인과 문자열 버퍼
    float _x, _y, _z;
    int num = 0;

    if ((fp = fopen(fileName, "r")) == NULL) {
        cout << "File is Not Found" << endl;
        return;
    }
    while (fgets(line, 256, fp) != NULL) {
        sscanf(line, "%s", str);
        if (strcmp(str, "*MESH") == 0) {    //"*MESH"라는 문자가 나올 때까지 계속
            fgets(line, 256, fp);           // *TIMEVALUE
            fgets(line, 256, fp);           // *MESH_NUMVERTEX
            sscanf(line, "%s%d", str, &nVtx);
            fgets(line, 256, fp);           // *MESH_NUMFACES
            sscanf(line, "%s %d", str, &nFace);

            vertex = new Vertex[nVtx];      // vertex 리스트 동적할당
            face = new Face[nFace];         // face 리스트 동적할당
            normal = new Normal[nFace];     // normal 리스트 동적할당
```

21

```
            fgets(line, 256, fp);           // *MESH_VERTEX_LIST
            sscanf(line, "%s", str);
            if (strcmp(str, "*MESH_VERTEX_LIST") == 0) {
                for (int i = 0; i < nVtx; i++) {        // vertex 의 x, y, z 더
                    fgets(line, 256, fp);  // 각 정점 정보 읽기
                    sscanf(line, "%s%d%f%f%f", str, &num, &_x, &_y, &_z);
                    vertex[i].x = _x;
                    vertex[i].y = _z;
                    vertex[i].z = _y;
                }
            }
            fgets(line, 256, fp);           // Read the Line '}'
            fgets(line, 256, fp);           // Read the Line  *MESH_FACE_LIST
            sscanf(line, "%s", str);
            if (strcmp(str, "*MESH_FACE_LIST") == 0) {
                for (int i = 0; i < nFace; i++) {   // face 의 vertex 인덱스 데이
                    fgets(line, 256, fp);   // Read the Line  *MESH_FACE_LIST
                    sscanf(line, "%s%s%s%d%s%d%s%d%s%d%s%d%s%d%s",
                        str, str, str,
                        &(face[i].vi[0]), str,
                        &(face[i].vi[1]), str,
                        &(face[i].vi[2]), str,
                        &num, str, &num, str, &num, str);
                }
            }
            fgets(line, 256, fp);  // Read the Line '}'
```

22

```
                fgets(line, 256, fp);  // Read the Line  *MESH_NUMCVERTEX ---> 없애야
                sscanf(line, "%s", str);
                if (strcmp(str, "*MESH_NUMCVERTEX") == 0) {     // 내용이 *MESH_NUMCVE
                    fgets(line, 256, fp);       // 다시  *MESH_NORMALS 를 위한 라인을 읽
                    sscanf(line, "%s", str);
                }
                if (strcmp(str, "*MESH_NORMALS") == 0) {    // normal 의 법선벡터 데이터
                    for (int i = 0; i < nFace; i++) {
                        float* nF = normal[i].norFace;
                        float* nV1 = normal[i].norV1;
                        float* nV2 = normal[i].norV2;
                        float* nV3 = normal[i].norV3;

                        fgets(line, 256, fp);  // Read the Line  *MESH_FACENORMAL
                        sscanf(line, "%s%d%f%f%f", str, &num, nF, nF + 1, nF + 2);
                        fgets(line, 256, fp);  // Read the Line  *MESH_VERTEXNORMAL 1
                        sscanf(line, "%s%d%f%f%f", str, &num, nV1, nV1+1, nV1+2);
                        fgets(line, 256, fp);  // Read the Line  *MESH_VERTEXNORMAL 2
                        sscanf(line, "%s%d%f%f%f", str, &num, nV2, nV2+1, nV2+2);
                        fgets(line, 256, fp);  // Read the Line  *MESH_VERTEXNORMAL 3
                        sscanf(line, "%s%d%f%f%f", str, &num, nV3, nV3 + 1, nV3 + 2);
                    }
                }
                break;
            }
        }
        fclose(fp);
    }
```

23

# 전체 프로그램(LoadMeshASE.cpp)

```
#include "Mesh.h"

static Mesh obj3D;

void display() {
    glClearColor(0.99, 0.97, 0.97, 1.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    obj3D.draw(80.0f, true);//Scale , 카메라
    glutSwapBuffers();더블 버퍼링
    glFlush();
}
static int PrevX, PrevY;
void mouseClick(int button, int state, int x, int y) { ... }
void mouseMotion(GLint x, GLint y) { ... }
```

24

```
#define FILTER_ASE  "ASE 3D data (*.ase)\0*.ase\0All (*.*)\0*.*\0"
```

정의

```
void keyboard(unsigned char key, int x, int y) {
    if (key == 'l') {
        char *filename = glkFileDlg(FILTER_ASE);
        if (filename != NULL)
            obj3D.readAse(filename);
//          obj3D.readAse("s_body.ase");
    }
    else if (key == 'i') {
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
    }
    else if (key == 'w') {
        glPolygonMode(GL_FRONT, GL_LINE);
        glPolygonMode(GL_BACK, GL_POINT);
    }
    else if (key == 's') {
        glPolygonMode(GL_FRONT, GL_FILL);
        glPolygonMode(GL_BACK, GL_LINE);
    }
    else if (key == 'q')
        exit(0);
    glutPostRedisplay();
}
```

파일 열기

```
#include <windows.h>
char* glkFileDlg(const char* filter)
{
    const int MaxLen = 1024;
    static char fileName[MaxLen] = "";

    OPENFILENAME    open_file;
    memset(&open_file, 0, sizeof(OPENFILENAME));
    open_file.lStructSize = sizeof(OPENFILENAME);
    open_file.hwndOwner = NULL;
    open_file.lpstrFilter = filter;
    open_file.nFilterIndex = 1;
    open_file.lpstrFile = fileName;
    open_file.nMaxFile = MaxLen;
    open_file.lpstrTitle = "Select a file";
    //open_file.lpstrDefExt = "bvh";
    open_file.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST | OFN_HIDEREADONLY

    bool ret = GetOpenFileName(&open_file);

    return (ret) ? fileName : NULL;
}
```

25

```
void initRendering()
{
    // 조명 처리
    GLfloat light_specular[] = { 0.8, 0.5, 0.8, 1.0 };
    GLfloat light_diffuse[] = { 0.8, 0.8, 0.8, 1.0 };
    GLfloat light_ambient[] = { 0.5, 0.5, 0.5, 1.0 };
    GLfloat light_position[] = { 0.f, -2.5f, 2.5f, 0.0f };
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
    glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);

    // Mesh의 재질
    GLfloat mat_specular[] = { 0.5, 1.0, 0.5, 1.0 };
    GLfloat mat_shininess[] = { 70.0 };
    GLfloat mat_color[] = { 0.5, 0.5, 0.5, 1.0 };
    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, mat_color);
    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, mat_color);
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, mat_specular);
    glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS, mat_shininess);

    glEnable(GL_LIGHTING);      // 조명 사용 설정
    glEnable(GL_LIGHT0);
    //  glEnable(GL_NORMALIZE);     // 정규화
    glEnable(GL_DEPTH_TEST);    // depth 검사(Z-버퍼 사용)
    glShadeModel(GL_SMOOTH);
}
```

10장 내용

26

```
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
    glutInitWindowSize(600, 600);
    glutCreateWindow("RobotWorld");

    glutDisplayFunc(display);       // 디스플레이 콜백 등록
    glutKeyboardFunc(keyboard);     // 키보드 콜백 등록
    glutMouseFunc(mouseClick);      // 마우스 클릭 콜백 등록
    glutMotionFunc(mouseMotion);    // 마우스 모션 콜백 등록

    initRendering();                // 조명값 설정
    keyboard('1', 0, 0);

    glutMainLoop();
    return 0;
}
```

10장

추가 코드는 동영상 참조

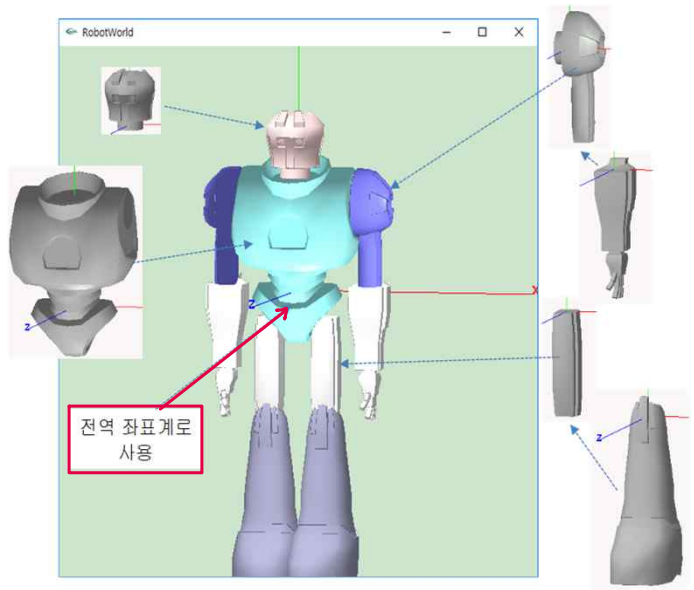
27

# 5.3 모델 변환 : 로봇의 합체

- 각 모델 좌표들의 통합 → 전역 좌표계
- 기준을 무엇으로 잡을까? 예를 들어, 몸통 좌표계
  - 머리 모델의 좌표들은?
  - 팔, 다리 등 모델들의 좌표들은 어떻게 해야하나?
- 머리를 더 크게 하려면? 다리를 더 길게 하려면?
- 로봇의 합체 과정 → 모델 변환
- OpenGL에서 모델 변환을 지원함 (함수들 제공)

행렬 모델 변환

28



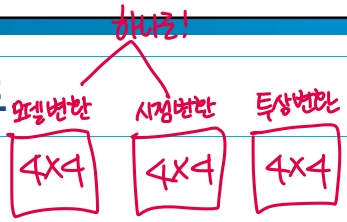

29

## 5.4 OpenGL의 모델 변환

- 행렬 모드의 선택
- 기준을 무엇으로 잡을까? 예를 들어, 몸통 좌표계
  - 머리 모델의 좌표들은?
  - 팔, 다리 등 모델들의 좌표들은 어떻게 해야하나?
- 머리를 더 크게 하려면? 다리를 더 길게 하려면?
- 로봇의 합체 과정 → 모델 변환
- OpenGL에서 모델 변환을 지원함 (함수들 제공)

30

## OpenGL 행렬 모드

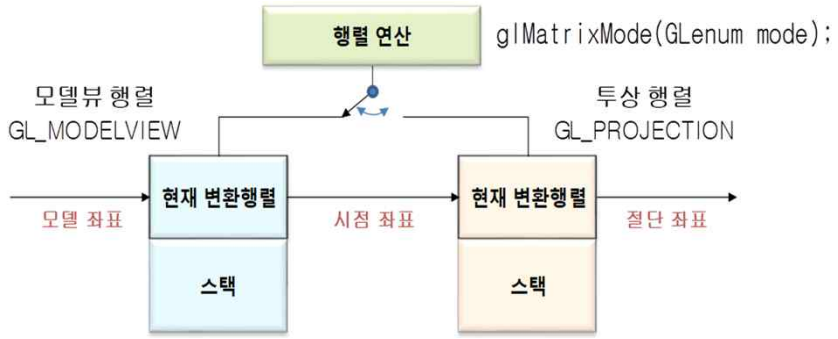



- 세 가지 행렬 모드
  - 모델뷰 행렬, 투영 행렬 및 텍스쳐 행렬
- 모델 행렬과 뷰 행렬을 따로 두지 않고 모델뷰 행렬 하나로 처리하는 것에 유의
  - 왜? 카메라를 돌려 물체의 측면을 촬영하는 것이나 카메라를 고정시키고 물체를 돌리는 것이나 같은 결과

- **현재 변환 행렬(Current Transformation Matrix, CTM)**

31

## 행렬 모드의 선택




```
void glMatrixMode(GLenum mode);
    GL_MODELVIEW : 모델뷰 행렬에 대한 변환
    GL_PROJECTION : 투상 행렬에 대한 변환
    GL_TEXTURE : 텍스쳐 행렬에 대한 변환
```

32

# OpenGL에서 제공하는 행렬 연산 함수

```
void glLoadIdentity( );
```

- 현재 변환행렬 CTM을 항등 행렬로 초기화 한다. 만약 행렬 모드가 GL_MODELVIEW라면 모델뷰 행렬이 항등행렬이 되고, GL_PROJECTION이라면 투상행렬이 항등행렬이 된다. 이것은 모든 행렬 연산 함수들에 동일하게 적용된다.

```
void glLoadMatrixf( const GLfloat *M )
```

- 행렬 M으로 CTM을 대치한다. 이때, 변환 행렬 M이 M[16] = { a, b, c, ... , p };의 순서를 갖는다면 행렬은 다음과 같이 정의된다.

$$M = \begin{bmatrix} a & e & i & m \\ b & f & j & n \\ c & g & k & o \\ d & h & l & p \end{bmatrix}$$

33

```
void glMultMatrixf( const GLfloat *M )
```

- CTM에 행렬 M을 곱한다. 변환 행렬이 후위에 곱해지는 것에 유의하라. 이제 CTM은 다음과 같다.

$$CTM = CTM \cdot M$$

34

```
void glTranslatef (GLfloat dx, GLfloat dy, GLfloat dz)
```

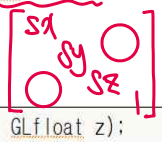
- (dx, dy, dz, 1)만큼의 이동 변환 행렬을 만들어 CTM에 곱한다. 이 행렬을 T라고 한다면 CTM은 다음과 같이 변한다.

$$CTM = CTM \cdot T$$

```
void glScalef (GLfloat sx, GLfloat sy, GLfloat sz);
```

- x, y, z축 방향으로 sx, sy, sz만큼 신축변환하는 행렬 S를 만들어 CTM에 곱한다.

$$CTM = CTM \cdot S$$




```
void glRotatef (GLfloat angle, GLfloat x, GLfloat y, GLfloat z);
```

- 벡터 (x, y, z, 0)를 축으로 angle 만큼 회전하는 변환 행렬을 만들어 CTM에 곱한다.

$$CTM = CTM \cdot R$$



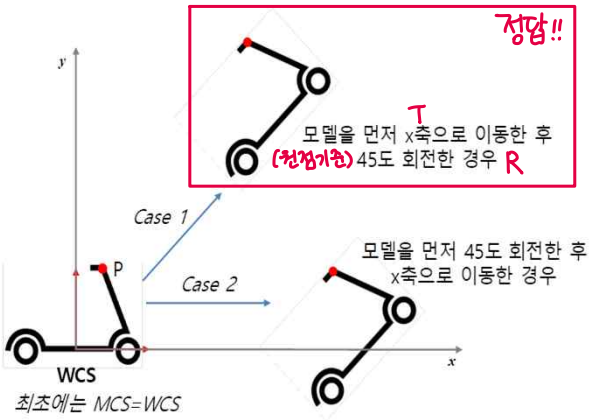
# OpenGL에서의 복합 변환

- 복합 변환시 유의할 점
  - 행렬곱 교환 법칙 성립 안함
    - 전역 좌표계에 어떤 물체를 이동한 후 회전시키는 것과, 회전을 먼저 시키고 그 결과를 이동시키는 것은 완전히 다른 결과
  - 변환의 적용 순서가 매우 중요

**가장 최근에 지정한 변환이 정점에 가장 먼저 적용된다.**

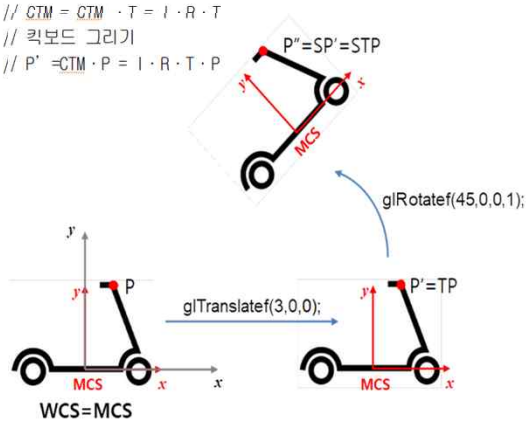
```
glMatrixMode(GL_MODELVIEW);          // CTM을 MODELVIEW 행렬로 설정
glLoadIdentity( );                   // CTM = I (항등행렬)
glRotatef(45, 0, 0, 1);              // CTM = CTM ·R = I ·R =R
glTranslatef(3, 0, 0);               // CTM = CTM ·T = I ·R ·T
glBegin(GL_POINTS);                  // 키보드 그리기
      glVertex3f(px, py, pz);        // P' =CTM ·P = I ·R ·T ·P
glEnd( );                                        변환이 적용되는 순서
```

- 한 번씩의 이동과 회전 변환을 이용한 복합 변환 ??






```
glMatrixMode(GL_MODELVIEW);          // CTM을 MODELVIEW 행렬로 설정
glLoadIdentity( );                   // CTM = I (항등행렬)
glRotatef(45, 0, 0, 1);              // CTM = CTM ·R = I·R
glTranslatef(3, 0, 0);               // CTM = CTM ·T = I·R·T
glBegin(GL_POINTS);                  // 키보드 그리기
      glVertex3f(px, py, pz);        // P' =CTM·P = I·R·T·P
glEnd( );
```

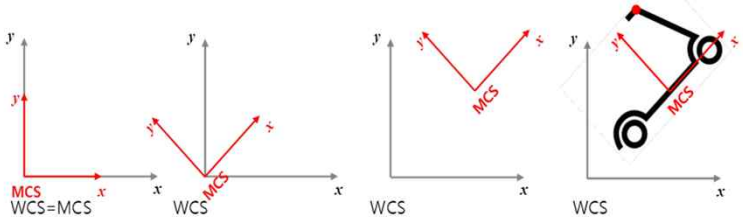


## 함수의 호출 순서는 좌표계의 변환 순서와 동일

```
glMatrixMode(GL_MODELVIEW);          // CTM을 MODELVIEW 행렬로 설정
glLoadIdentity( );                   // CTM = I (항등행렬)
glRotatef(45, 0, 0, 1);              // CTM = CTM ·R = I·R
glTranslatef(3, 0, 0);               // CTM = CTM ·T = I·R·T
glBegin(GL_POINTS);                  // 키보드 그리기
      glVertex3f(px, py, pz);        // P' =CTM·P = I·R·T·P
glEnd( );
```

그림이 어떻게 나오는지 시험!

glLoadIdentity(); glRotatef(45,0,0,1); glTranslatef(3,0,0); glVertex(...);




39

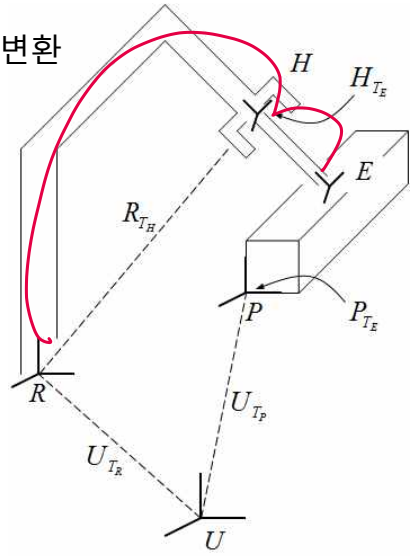
## 5.5 행렬 스택

- CTM만으로는 어떤 변환이 어떤 순서로 적용되었는지 알 수는 없음
- 만약 변환이 적용되기 전의 상태로 정확히 되돌리기 위해서는? 예) 로봇에서 왼팔 장착 후 오른팔 장착
- OpenGL에서는 이를 위해 행렬 스택 사용

```
glPushMatrix( );      // 현재 변환 행렬 CTM을 행렬 스택에 저장
glPopMatrix( );       // 스택의 최상위 행렬을 꺼내 CTM으로 설정함
```

40

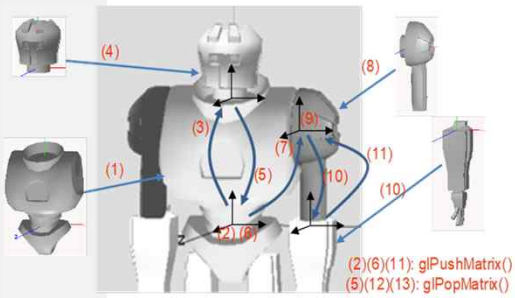
- 행렬 스택을 이용한 좌표계 변환




41

# 로봇의 머리와 왼쪽 팔을 조립하는 과정

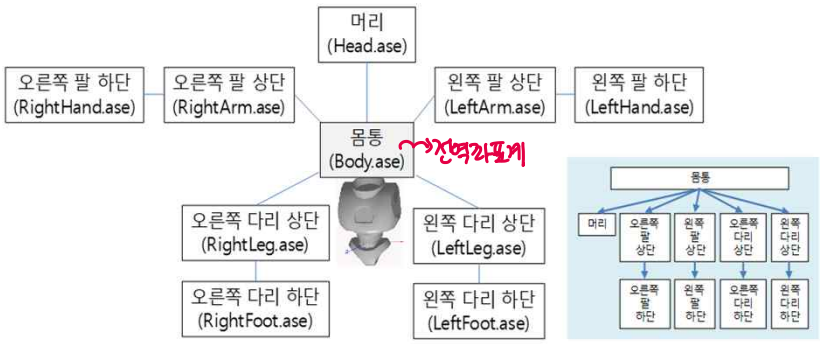
```
(1)  draw_body();
(2)  glPushMatrix();
(3)       glTranslatef(); …
(4)       draw_head();
(5)  glPopMatrix();
(6)  glPushMatrix();
(7)       glTranslatef(); …
(8)       draw_upper_left_arm();
(9)       glPushMatrix();
(10)        glTranslatef(); …
(11)        draw_lower_left_arm();
(12)      glPopMatrix();
(13) glPopMatrix();
(14)      ...
```




42

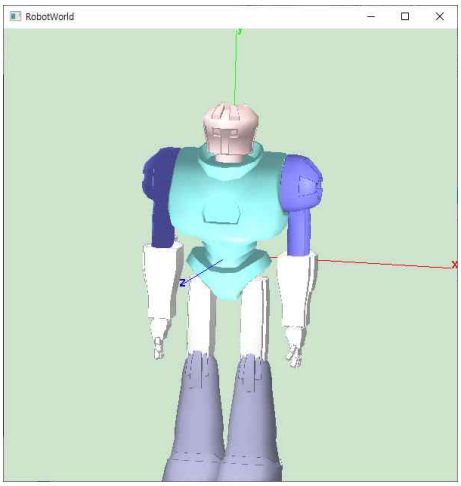
# 5.6 모델 변환 응용: 로봇을 조립하자.

- 로봇의 전체 부품과 연결 관계: 계층적인 트리 구조






# Lab: 로봇 조립 프로그램




## Robot 클래스 추가

```
#pragma once
#include "Mesh.h"

class Robot {
public:
    Mesh Head, Body, RightArm, RightHand, LeftArm, LeftHand;
    Mesh RightLeg, RightFoot, LeftLeg, LeftFoot;
    float scale;        // 모델의 스케일

    Robot() {
        scale = 100;    // 좌표값이 -100~100 범위

        printf("Loading Robot Models....\n");
        Head.readAse("S_Head.ASE");             printf("  loading Head..... OK.\n");
        Body.readAse("S_Body.ASE");             printf("  loading Body.....OK.\n");
        RightArm.readAse("S_RightArm.ASE");     printf("  loading RightArm....OK.\n");
        RightHand.readAse("S_RightHand.ASE");   printf("  loading Righthand...OK.\n");
        LeftArm.readAse("S_LeftArm.ASE");       printf("  loading LeftArm.....OK.\n");
        LeftHand.readAse("S_LeftHand.ASE");     printf("  loading LeftHand....OK.\n");
        RightLeg.readAse("S_RightLeg.ASE");     printf("  loading RightLeg....OK.\n");
        RightFoot.readAse("S_RightFoot.ASE");   printf("  loading RightFoot...OK.\n");
        LeftLeg.readAse("S_LeftLeg.ASE");       printf("  loading LeftLeg.....OK.\n");
        LeftFoot.readAse("S_LeftFoot.ASE");     printf("  loading LeftFoot....OK.\n");
    }
```



## Robot::draw() : 로봇을 조립해서 그리기

```
void draw() {
    Body.draw(0.5, 0.8, 0.8, scale, true);
    glPushMatrix();             // 몸통 좌표계 저장
        glTranslated(0.0, 0.45, -0.07);
        glScalef(1.1f, 1.1f, 1.1f);
        Head.draw(0.8, 0.7, 0.7, scale);
    glPopMatrix();              // 몸통 좌표계로 돌아가기

    glPushMatrix();             // 몸통 좌표계 저장
        glTranslated(-0.25, 0.32, 0.0);
        glScalef(1.0f, 1.0f, 1.0f);
        RightArm.draw(0.4, 0.4, 0.8, scale);

        glPushMatrix();         // 오른어깨 좌표계 저장
            glTranslated(-0.05, -0.3, 0.0);
            glScalef(1.0f, 1.0f, 1.0f);
            RightHand.draw(0.8, 0.8, 0.8, scale);
        glPopMatrix();          // 오른어깨 좌표계 돌아가기
    glPopMatrix();              // 몸통 좌표계 돌아가기
```

- Mesh의 draw()함수 오버로딩
  - 각 부품의 색을 다르게 하기 위해

```
1. class Mesh {
2.     ...
3.     void setColor(float r, float g, float b, float a) {
4.             float color[4] = { r,g,b,a };
5.     꼬면 glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, color);
6.     특성 glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, color);
7.     }
8.     void draw(float r,float g,float b,float scale=1.0f, bool bCoord=false){
9.             setColor(r, g, b, 1.0);       // 재질의 색상을 설정한 후
10.            draw(scale, bCoord);          // draw()함수 호출
11.    }
12. };
```



## MakeRobot.cpp (LoadMeshASE.cpp 수정)

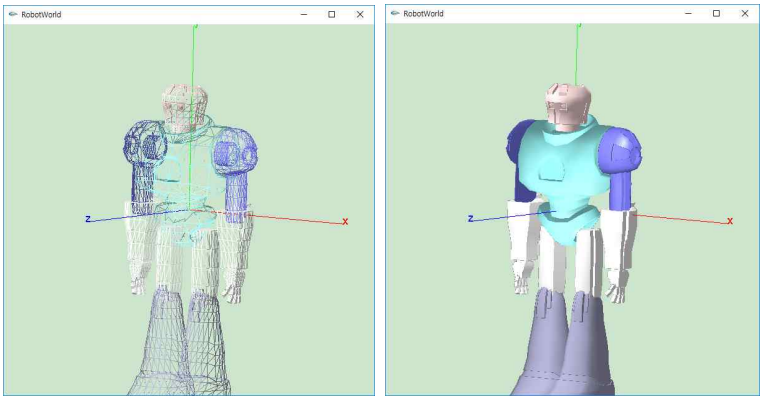
```
#include "Robot.h"
#include <gl/glut.h>
#include "glk.h"

//static Mesh obj3D;
static Robot robot;
static bool bRobotRun = false;

void display() {
    glClearColor(0.8, 0.9, 0.8, 1.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    robot.draw();
    //obj3D.draw(80.0f, true);
    glutSwapBuffers(); 더블버퍼링
    glFlush();
}
```

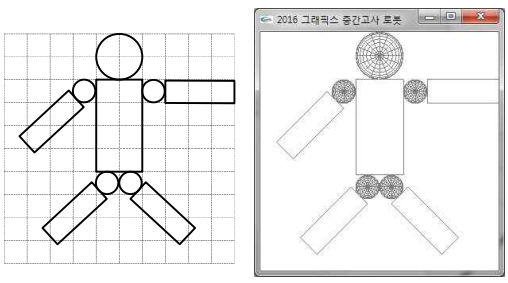
## 실행 결과



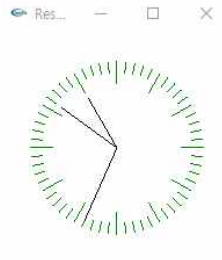



## 5장 연습문제

- [Lab 5-1] 모델 변환을 이용하여 다음과 같은 로봇 그리기 프로그램을 구현하라. 이때, 모든 부품은 glut의 Sphere와 Cube를 사용하면 되며, 팔과 다리는 동일한 형태이다.



- [Lab 5-2] 1초마다 현재 시각을 화면에 출력하는 시계 프로그램을 완성하라. 시계판은 모두 선분(GL_LINES)을 사용하면 되고, 시계침은 선분이나 삼각형 등을 이용해 그릴 수 있다. 1초에 한번씩 갱신되기 위해서는 타이머 콜백 함수를 사용해야 할 것이다.



51

- [Lab 5-3] 이 장에서 공부한 로봇 예제를 이용해 자신만의 로봇을 만들어 보라. 부품의 수는 최소한 프로그램 13의 로봇과 같거나 이상이 되도록 하라. 그림 5.12와 같은 계층구조트리를 그려보라. 부품들은 3DS Max등을 이용해 직접 설계해도 되지만 인터넷에서 찾아서 활용해도 좋다. 물론 이 경우 3DS Max등의 편집기에서 모델 편집 기능을 사용할 수 있어야 할 것이다. 또한 모델의 기준 좌표축과 크기 등에 대해 정확히 파악되어야 조립이 가능할 것이다.

52

감사합니다!

53