

**숙제2 로봇 팔 움직임 구현**

**과목명 컴퓨터그래픽스**

**담당교수 김상철교수님**

**제출일 20210519**

**전공 컴퓨터전자시스템**

**학번 201904458**

**이름 이준용**

* **소스 코드**

/\* 제목 : 한국외대 컴퓨터 그래픽스 과제

프로그램 : 로봇팔 움직임 구현

학 과 : 컴퓨터 전자시스템 전공

학 번 : 201904458

이 름 : 이준용

\*/

#include <windows.h>

#include <GL/glut.h>

#include <GL/gl.h>

#include <GL/glu.h>

#include <stdlib.h>

int shoulderleft = 0, elbowleft = 0, body = 0, legleft = 0, footleft = 0, head = 0, neck = 0;

int shoulderright = 0, elbowright = 0, legright = 0, footright = 0;

int flagshoulderleft = 0, flagelbowleft = 0, flagshoulderright = 0, flagelbowright = 0;

int stop\_elbowleft = 0, stop\_elbowright = 0;

float x\_Translation = 0.0, y\_Translation = 0.0, z\_move = 0.0;

float x\_Rotation = 0.0, y\_Rotation = 0.0;

bool mouse\_Lbutton\_pressed = false;

bool mouse\_Rbutton\_pressed = false;

bool mouse\_Mbutton\_pressed = false;

float Rmouse\_downx = 0;

float Rmouse\_downy = 0;

float Lmouse\_downx = 0;

float Lmouse\_downy = 0;

float mouse\_Mdownx = 0;

float mouse\_Mdowny = 0;

/\*opengl 277p 오렌지 매달기 프로그램(행렬 스택 활용 부분) 참조함 \*/

void drawhead()

{

glPushMatrix();//현재 단위 행렬을 스택에 넣습니다. 즉, 현재 행렬을 저장합니다.

glTranslatef(-1.0, 0.0, 0.0);//좌표계이동 현재 행렬은 Tt1

glRotatef((GLfloat)head, 0.0, 0.0, 1.0);//좌표계 회전 Tr행렬 만들기

glTranslatef(1.0, 0.0, 0.0);//좌표계 이동 현재 행렬은 Tt2

glPushMatrix();//현재 행렬을 스택으로 다시 push

glScalef(1.2, 1.2, 1.2);//좌표계 크기 확대후 Ts 행렬 만들기

glutWireCube(1.0);//머리 그리기(정육면체 선분만 그리기)

glColor3f(1.0f, 0.0f, 0.0f);

glPopMatrix();//스케일 효과를 취소하기 위해 매트릭스가 나타납니다.

glPopMatrix();//처음의 push했던 단위행렬로 다시 위치시킵니다.

}

void drawbody()

{

glPushMatrix();//현재 단위 행렬을 스택에 넣습니다. 즉, 현재 행렬을 저장합니다.

glTranslatef(-1.0, 0.0, 0.0);//좌표계이동 현재 행렬은 Tt1

glRotatef((GLfloat)body, 0.0, 0.0, 1.0);//좌표계 회전 Tr행렬 만들기

glTranslatef(1.0, 0.0, 0.0);//좌표계 이동 현재 행렬은 Tt2

glPushMatrix();//현재 행렬을 스택으로 다시 푸시

glScalef(3.7, 1.2, 2);//좌표계 크기 확대후 Ts 행렬 만들기

glutWireCube(1.0);//몸체 그리기(정육면체 선분만 그리기)

glColor3f(1.0f, 0.0f, 0.0f);

glPopMatrix();//스케일 효과를 취소하기 위한 행렬을 나타납니다.

glPopMatrix();//처음의 push했던 단위행렬로 다시 위치시킵니다.

}

void drawneck()

{

glPushMatrix();//현재 단위 행렬을 스택에 넣습니다. 즉, 현재 행렬을 저장합니다.

glTranslatef(-1.0, 0.0, 0.0);//좌표계이동 현재 행렬은 Tt1

glRotatef((GLfloat)head, 0.0, 0.0, 1.0);//좌표계 회전 Tr행렬 만들기

glTranslatef(1.0, 0.0, 0.0);//좌표계 이동 현재 행렬은 Tt2

glPushMatrix();//현재 행렬을 스택으로 다시 push

glScalef(0.4, 0.7, 0.6);//좌표계 크기 확대후 Ts 행렬 만들기

glutWireCube(1.0);//목 그리기(정육면체 선분만 그리기)

glColor3f(1.0f, 0.0f, 0.0f);

glPopMatrix();//스케일 효과를 취소하기 위해 매트릭스가 나타납니다.

glPopMatrix();//처음의 push했던 단위행렬로 다시 위치시킵니다.

}

void drawlimbright()

{

glPushMatrix();//현재 단위 행렬을 스택에 넣습니다. 즉, 현재 행렬을 저장합니다.

glTranslatef(-1.0, 0.0, 0.0);//좌표계이동 현재 행렬은 Tt1

glRotatef((GLfloat)shoulderright, 0.0, 0.0, 1.0);//좌표계 회전 Tr행렬 만들기

glTranslatef(1.0, 0.0, 0.0);//좌표계 이동 현재 행렬은 Tt2

glPushMatrix();//현재 행렬을 스택으로 다시 푸시

glScalef(1.8, 0.4, 0.4);//좌표계 크기 확대후 Ts 행렬 만들기

glutWireCube(1.0);//위쪽 팔을 그립니다. (오른쪽)

glPopMatrix();// 위에서 glPushMatrix()에서 저장한 좌표계를 pop해서 scale이전의 좌표계를 다시 가져옴.

glTranslatef(0.75, 0.0, 0.0); //Tt1 이동

glRotatef((GLfloat)elbowright, 0.0, 0.0, 1.0); //Tr행렬 만들기

glTranslatef(1.0, 0.0, 0.0); //Tt2 이동

glPushMatrix();//현재 행렬을 스택으로 다시 푸시

glScalef(1.7, 0.4, 0.4);

glutWireCube(1.0);//팔의 팔뚝 그리기(정육면체 선분만 그리기)

glColor3f(1.0f, 0.0f, 0.0f);

glPopMatrix();//psuh했던 행렬 pop

glPopMatrix();//처음의 push했던 단위행렬로 다시 위치시킵니다.

}

void drawlimbleft()

{

glPushMatrix();//현재 단위 행렬을 스택에 넣습니다. 즉, 현재 행렬을 저장합니다.

glTranslatef(-1.0, 0.0, 0.0);//좌표계이동 현재 행렬은 Tt1

glRotatef((GLfloat)shoulderleft, 0.0, 0.0, 1.0);//좌표계 회전 Tr행렬 만들기

glTranslatef(1.0, 0.0, 0.0);//좌표계 이동 현재 행렬은 Tt2

glPushMatrix();//현재 행렬을 스택으로 다시 push

glScalef(1.8, 0.4, 0.4);//좌표계 크기 확대후 Ts 행렬 만들기

glutWireCube(1.0);//위쪽 팔을 그립니다. (왼쪽)

glPopMatrix();//위에서 glPushMatrix()에서 저장한 좌표계를 pop해서 scale이전의 좌표계를 다시 가져옴.

glTranslatef(0.75, 0.0, 0.0); //Tt1 이동

glRotatef((GLfloat)elbowleft, 0.0, 0.0, 1.0); //Tr행렬 만들기

glTranslatef(1.0, 0.0, 0.0); //Tt2 이동

glPushMatrix();//현재 행렬을 스택으로 다시 푸시

glScalef(1.7, 0.4, 0.4);

glutWireCube(1.0);//팔의 팔뚝 그리기(정육면체 선분만 그리기)

glColor3f(1.0f, 0.0f, 0.0f);

glPopMatrix();//psuh했던 행렬 pop

glPopMatrix();//처음의 push했던 단위행렬로 다시 위치시킵니다.

}

void drawlegleft()

{

glPushMatrix();//현재 단위 행렬을 스택에 넣습니다. 즉, 현재 행렬을 저장합니다.

glTranslatef(-0.7, 0.0, 0.0);//좌표계이동 현재 행렬은 Tt1

glRotatef((GLfloat)legleft, 0.0, 0.0, 1.0);//좌표계 회전 Tr행렬 만들기

glTranslatef(1.0, 0.0, 0.0);//좌표계 이동 현재 행렬은 Tt2

glPushMatrix();//현재 행렬을 스택으로 다시 push

glScalef(2.0, 0.6, 0.6);//좌표계 크기 확대후 Ts 행렬 만들기

glutWireCube(1.0);//왼쪽 다리 그리기(정육면체 선분만 그리기)

glPopMatrix();//위에서 glPushMatrix()에서 저장한 좌표계를 pop해서 scale이전의 좌표계를 다시 가져옴.

glTranslatef(1.0, 0.0, 0.0); //Tt1 이동

glRotatef((GLfloat)footleft, 0.0, 0.0, 1.0); //Tr행렬 만들기

glTranslatef(1.0, 0.0, 0.0); //Tt2 이동

glPushMatrix();//현재 행렬을 스택으로 다시 푸시

glScalef(2.0, 0.6, 0.6);

glutWireCube(1.0);//왼쪽 다리 그리기(정육면체 선분만 그리기)

glColor3f(1.0f, 0.0f, 0.0f);

glPopMatrix();//psuh했던 행렬 pop

glPopMatrix();//처음의 push했던 단위행렬로 다시 위치시킵니다.

}

void drawlegright()

{

glPushMatrix();//현재 단위 행렬을 스택에 넣습니다. 즉, 현재 행렬을 저장합니다.

glTranslatef(-0.7, 0.0, 0.0);//좌표계이동 현재 행렬은 Tt1

glRotatef((GLfloat)legright, 0.0, 0.0, 1.0);//좌표계 회전 Tr행렬 만들기

glTranslatef(1.0, 0.0, 0.0);//좌표계 이동 현재 행렬은 Tt2

glPushMatrix();//현재 행렬을 스택으로 다시 push

glScalef(2.0, 0.6, 0.6);//좌표계 크기 확대후 Ts 행렬 만들기

glutWireCube(1.0);//오른쪽 다리 그리기(정육면체 선분만 그리기)

glPopMatrix();//위에서 glPushMatrix()에서 저장한 좌표계를 pop해서 scale이전의 좌표계를 다시 가져옴.

glTranslatef(1.0, 0.0, 0.0);//Tt1 이동

glRotatef((GLfloat)footright, 0.0, 0.0, 1.0);//Tr행렬 만들기

glTranslatef(1.0, 0.0, 0.0);//Tt2 이동

glPushMatrix();//현재 행렬을 스택으로 다시 푸시

glScalef(2.0, 0.6, 0.6);

glutWireCube(1.0);//오른쪽 다리 그리기(정육면체 선분만 그리기)

glColor3f(1.0f, 0.0f, 0.0f);

glPopMatrix();//psuh했던 행렬 pop

glPopMatrix();//처음의 push했던 단위행렬로 다시 위치시킵니다.

}

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glLoadIdentity();

// 처음의 로봇을 볼때 시점을 설정

gluLookAt(10.0+x\_Translation, y\_Translation, z\_move, x\_Translation, y\_Translation, z\_move, 0.0, 1.0, 0.0);

glRotatef(x\_Rotation, 1.0, 0.0, 0.0);

glRotatef(y\_Rotation, 0.0, 1.0, 0.0);

//왼쪽 팔 그리기

glPushMatrix();//현재 행렬을 스택으로 푸시합니다. 즉, 현재 행렬을 저장합니다.

glTranslatef(-1.0, 0.5, -1.3);

glRotatef(-90.0, 0.0, 0.0, 1.0);

glColor3f(0.0f, 1.0f, 1.0f);

drawlimbleft();

glPopMatrix();//저장된 현재 좌표계 행렬 pop

//오른쪽 팔 그리기

glPushMatrix();//현재 행렬을 스택으로 푸시합니다. 즉, 현재 행렬을 저장합니다.

glTranslatef(-1.0, 0.5, 1.3);

glRotatef(-90.0, 0.0, 0.0, 1.0);

glColor3f(0.0f, 1.0f, 1.0f);

drawlimbright();

glPopMatrix();

//목 그리기

glPushMatrix();

glTranslatef(-1.0, 0.0, 0.0);

glRotatef(90.0, 0.0, 0.0, 1.0);

drawbody();

glPopMatrix();

//머리 그리기

glPushMatrix();

glTranslatef(-1.0, 2.8, 0.0);

glRotatef(90.0, 0.0, 0.0, 1.0);

drawhead();

glPopMatrix();

//목 그리기

glPushMatrix();

glTranslatef(-1.0, 2.0, 0.0);

glRotatef(90.0, 0.0, 0.0, 1.0);

drawneck();

glPopMatrix();

//왼쪽 다리 그리기

glPushMatrix();

glTranslatef(-1.0, -2.5, -0.5);

glRotatef(-90.0, 0.0, 0.0, 1.0);

drawlegleft();

glPopMatrix();

//오른쪽 다리 그리기

glPushMatrix();

glTranslatef(-1.0, -2.5, 0.5);

glRotatef(-90.0, 0.0, 0.0, 1.0);

drawlegright();

glPopMatrix();

glutSwapBuffers();

}

void reshape(int w, int h)

{ // gluLoadIdentity() gluPerspective()

/\*생성 된 matrix를 사용 하 여 glMultMatrix 를 호출한 것 처럼

gluPerspective 에 의해 생성 된 행렬에 현재 행렬이 곱해집니다.

대신 현재 matrix 스택에 큐브 뷰 매트릭스를 로드 하려면

glLoadIdentity에 대 한 호출을 사용 하 여 gluPerspective 에 대 한 호출 앞에 옵니다.\*/

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(65.0, (GLfloat)w / (GLfloat)h, 1.0, 200.0);

//1.y방향의 보기각도 65도

//2. X 방향의 뷰 필드를 결정 하는 가로 세로 비율입니다. 가로 세로 비율은 x (너비)와 y (높이)의 비율입니다

//3. 뷰어에서 가까운 클리핑 평면 까지의 거리 (항상 양수)입니다.

//4.뷰어에서 먼 클리핑 평면 까지의 거리 (항상 양수)입니다.

glMatrixMode(GL\_MODELVIEW);

}

void keyboard(unsigned char key, int x, int y)

{

switch (key) {

case 'q': // 양팔 들어 올리기 (360회전, z 양수축으로 증가하는것이 양수)

shoulderleft = (shoulderleft + 3) % 360;

shoulderright = (shoulderright + 3) % 360;

glutPostRedisplay();

break;

case 'a': // 양팔 전체 내리기 (360회전, z 음수축으로 증가하는것이 음수)

shoulderleft = (shoulderleft - 3) % 360;

shoulderright = (shoulderright - 3) % 360;

glutPostRedisplay();

break;

case 'p': // 양쪽 팔뚝부분 전체 올리기( 각도가 0부터 165까지만 올리고 165되면 팔올리기 멈춤)

if (stop\_elbowleft) {

elbowleft = (elbowleft + 3) % 360;

if (elbowleft >= 165 || elbowleft >= -165)

stop\_elbowleft = 0;

}

else{

if (elbowleft <= 165 || elbowleft <= -165)

stop\_elbowleft = 1;

}

if (stop\_elbowright) {

elbowright = (elbowright + 3) % 360;

if (elbowright >= 165 || elbowright >= -165)

stop\_elbowright = 0;

}

else {

if (elbowright <= 165 || elbowright <= -165)

stop\_elbowright = 1;

}

glutPostRedisplay();

break;

case 'l': // 양쪽 팔뚝 부분 전체 내리기 ( 각도가 0이 될경우는 팔을 일자로 펴졌을 경우임. 각도가 0이 되면 멈춤)

if (stop\_elbowleft) {

elbowleft = (elbowleft - 3) % 360;

if (elbowleft == 0)

stop\_elbowleft = 0;

}

else {

if (elbowleft > 0 && elbowleft <= 170)

stop\_elbowleft = 1;

}

if (stop\_elbowright) {

elbowright = (elbowright - 3) % 360;

if (elbowright == 0)

stop\_elbowright = 0;

}

else {

if (elbowright > 0 && elbowright <= 170)

stop\_elbowright = 1;

}

glutPostRedisplay();

break;

case'w': // 자연스럽게 걷기

//왼쪽 팔 앞뒤로 흔들기

if (flagshoulderleft)

{

shoulderleft = (shoulderleft - 3) % 360;

if (shoulderleft <= -40)flagshoulderleft = 0;

}

else

{

shoulderleft = (shoulderleft + 3) % 360;

if (shoulderleft >= 40)flagshoulderleft = 1;

}

if (flagelbowleft)

{

elbowleft = (elbowleft - 3) % 360;

if (elbowleft <= -5)flagelbowleft = 0;

}

else

{

elbowleft = (elbowleft + 3) % 360;

if (elbowleft >= 40)flagelbowleft = 1;

}

//오른쪽 팔 앞뒤로 흔들기

if (flagshoulderright)

{

shoulderright = (shoulderright + 3) % 360;

if (shoulderright >= 40)flagshoulderright = 0;

}

else

{

shoulderright = (shoulderright - 3) % 360;

if (shoulderright <= -40)flagshoulderright = 1;

}

if (flagelbowright)

{

elbowright = (elbowright + 3) % 360;

if (elbowright >= 40)flagelbowright = 0;

}

else

{

elbowright = (elbowright - 3) % 360;

if (elbowright <= -5)flagelbowright = 1;

}

glutPostRedisplay();

break;

case 27:

exit(0);

break;

default:

break;

}

}

void mouse(int button, int state, int x, int y)

{

if (button == GLUT\_LEFT\_BUTTON) {

if (state == GLUT\_DOWN)//마우스 왼쪽 버튼을 누르면 mouse\_Lbutton\_pressed가 설정되고 커서 위치가 기록됩니다.

{

mouse\_Lbutton\_pressed = true;

Lmouse\_downx = x;

Lmouse\_downy = y;

}

else {

mouse\_Lbutton\_pressed = false;

}

}

if (button == GLUT\_RIGHT\_BUTTON) {

if (state == GLUT\_DOWN)//마우스 오른쪽 버튼을 누르면 mouse\_Rbutton\_pressed가 설정되고 커서 위치가 기록됩니다.

{

mouse\_Rbutton\_pressed = true;

Rmouse\_downx = x;

Rmouse\_downy = y;

}

else {

mouse\_Rbutton\_pressed = false;

}

}

if (button == GLUT\_MIDDLE\_BUTTON) {

if (state == GLUT\_DOWN)//마우스 휠을 누르면 mouse\_Mbutton\_pressed가 설정되고 커서 위치가 기록됩니다.

{

mouse\_Mbutton\_pressed = true;

mouse\_Mdownx = x;

mouse\_Mdowny = y;

}

else {

mouse\_Mbutton\_pressed = false; // 마우스 나머지 버튼은 사용 x

}

}

}

void mouseMove(int x, int y)

{

if (mouse\_Lbutton\_pressed) { // 마우스 좌클릭 한채로

x\_Translation += (Lmouse\_downx - x) / 50.0f; // +x축으로 마우스 이동시 로봇 오른쪽으로 회전

y\_Translation -= (Lmouse\_downy - y) / 50.0f; // -x으로 마우스 이동시 로봇 왼쪽으로 회전

Lmouse\_downx = x; // -y축으로 마우스 이동시 로봇 반시계 방향 회전

Lmouse\_downy = y; // +y축으로 마수으 이동시 로봇 시계 방향 회전

glutPostRedisplay();

}

if (mouse\_Rbutton\_pressed) { // 마우스 우클릭 한채로

x\_Rotation -= (Rmouse\_downy - y) / 3.0f;

y\_Rotation -= (Rmouse\_downx - x) / 3.0f;

Rmouse\_downx = x;

Rmouse\_downy = y;

glutPostRedisplay();

}

}

void specialKeyFunc(int key, int x, int y)

{

switch (key)

{

case GLUT\_KEY\_DOWN: // 아래키 왼쪽으로 한칸 이동

z\_move--;

glutPostRedisplay();

break;

case GLUT\_KEY\_UP: // 위키 오른쪽으로 한칸 이동

z\_move++;

glutPostRedisplay();

break;

case GLUT\_KEY\_LEFT: // 왼쪽키 로봇 반시계 방향으로 회전

x\_Rotation++;

glutPostRedisplay();

break;

case GLUT\_KEY\_RIGHT: // 오른쪽키 로봇 반시계 방향으로 회전

x\_Rotation--;

glutPostRedisplay();

break;

default:

break;

}

}

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE); // 애니메이션을 사용 할때 더블 버퍼링을 사용하는게 좋다. 더블버퍼링 안하면 ( 깜빡이는현상 발생)

//전면 버퍼를 화면에 뿌리고, 애니메이션은

// 그위에 뿌리는 형식

glutInitWindowSize(800, 1000); // 윈도우의 사이즈를 800, 1000으로 설정

glutInitWindowPosition(400, 0); // 프로그램 실행 시킬 경우 나타나는 화면을 가운데로 위치시킴.

glutCreateWindow("OpenGl\_숙제2 drawArm 로봇팔 움직임구현\_201904458\_이준용"); //새로운 윈도우 상단에 title 바에 나타남

glClearColor(1.0, 1.0, 1.0, 1.0); //float형 색상 값이 들어감 인자는 순서대로 RGB 마지막 인자는 알파(투명도) (1.0,1.0,1.0,1.0)으로 바꾸면 배경화면이 흰색으로 바뀜

// glClearColor 함수가 배경화면 설정하는 함수임.

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glutDisplayFunc(display); // 화면에 출력

glutReshapeFunc(reshape);

glutKeyboardFunc(keyboard); // 키보드 입력

glutMouseFunc(mouse); // 마우스 입력

glutMotionFunc(mouseMove); // 마우스로 화면 회전 이동

glutSpecialFunc(specialKeyFunc); // 위아래좌우키로 화면 이동

glutMainLoop();

return 0;

}

* **화면 dump**

**★ p => 팔 전체 올리기**

**★ a => 팔 전체 내리기**

**★ p => 팔 아래부분 올리기 (팔 접히면 멈춤)**

**★ l => 팔 아래부분 내리기 (팔이 일자로 펴질 경우 멈춤)**

**★ w => 자연스럽게 걷기 모션**

**★ 마우스 ‘우’ 클릭하면서 상하좌우 => 상하: 부드럽게 원판 돌리기**

**좌우: 물체 커지기와 작아지기**

**★ 마우스 ‘좌’ 클릭하면서 상하좌우 => 상하: 카메라 위아래 움직임**

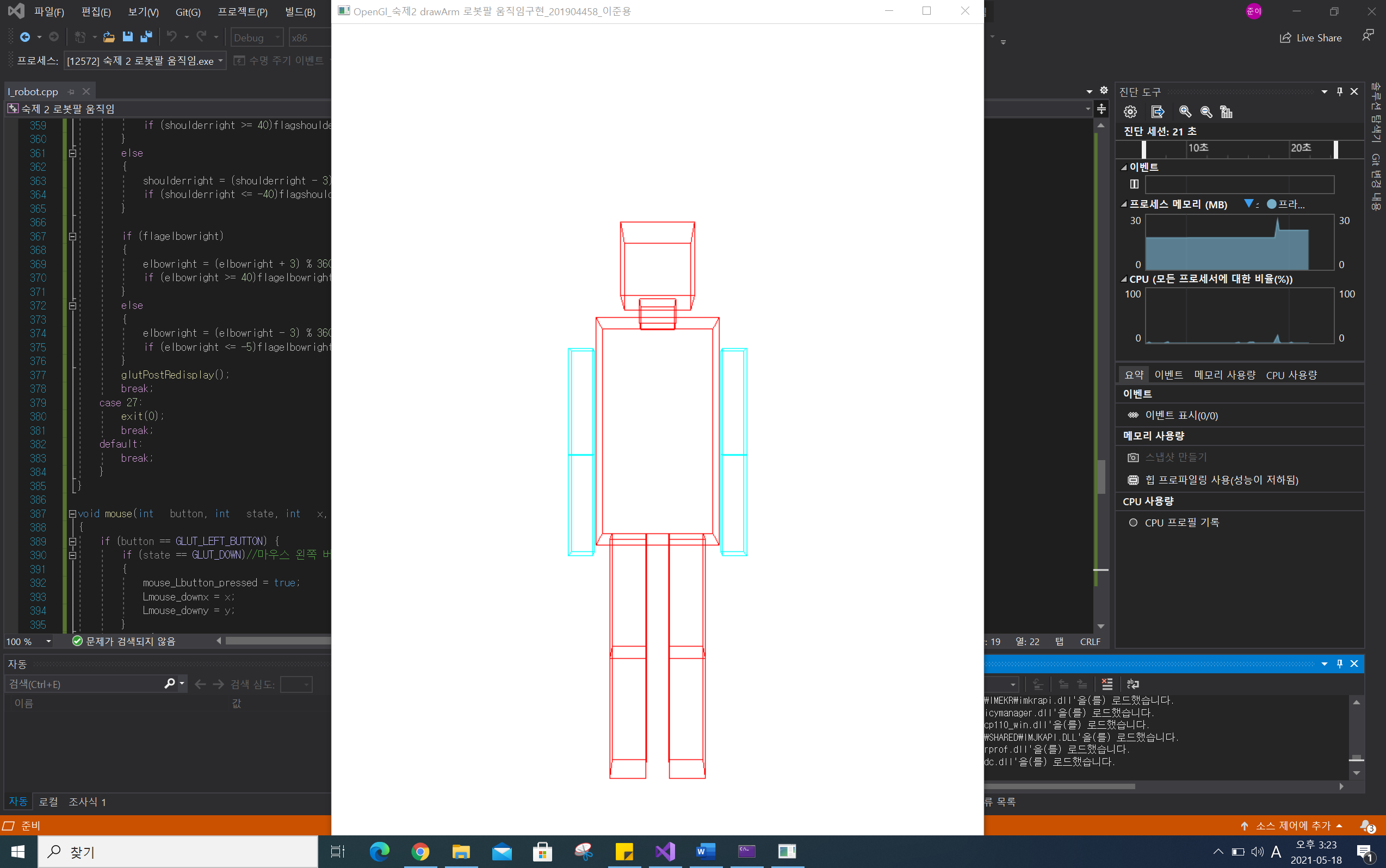
**좌우: 카메라 멀어짐과 가까워짐**

**★ 키보드 상하좌우 화살표키 => 상하: 작은 각도로 원판 돌리기**

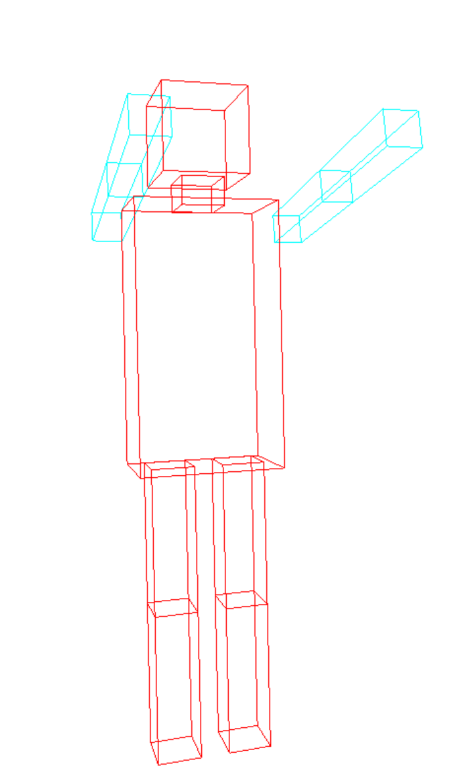
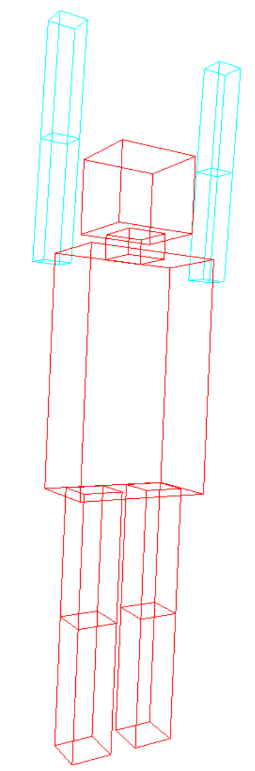
**좌우: 물체 좌우로 한칸씩 움직임**

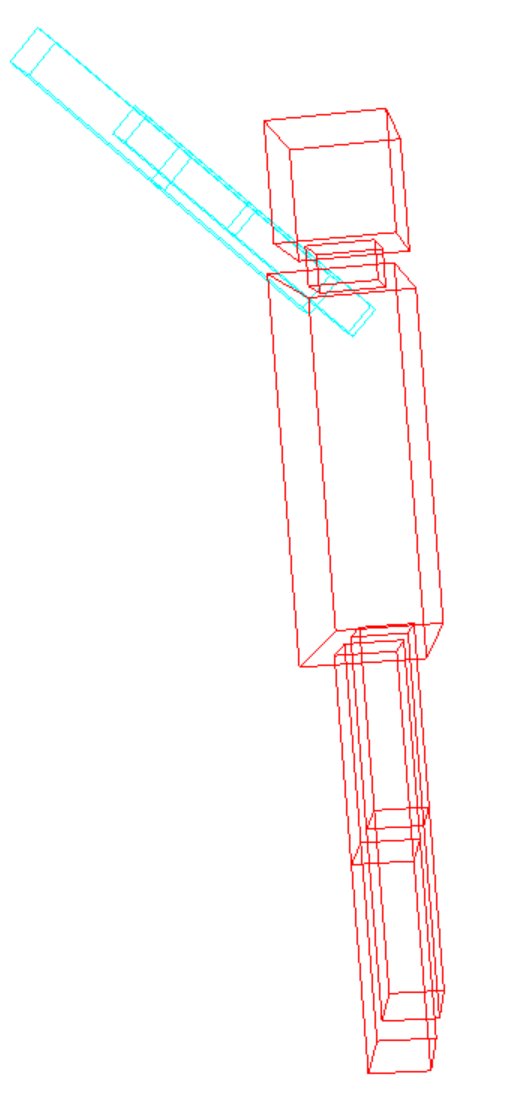
**★종료(esc)**

<프로그램 시작 화면>

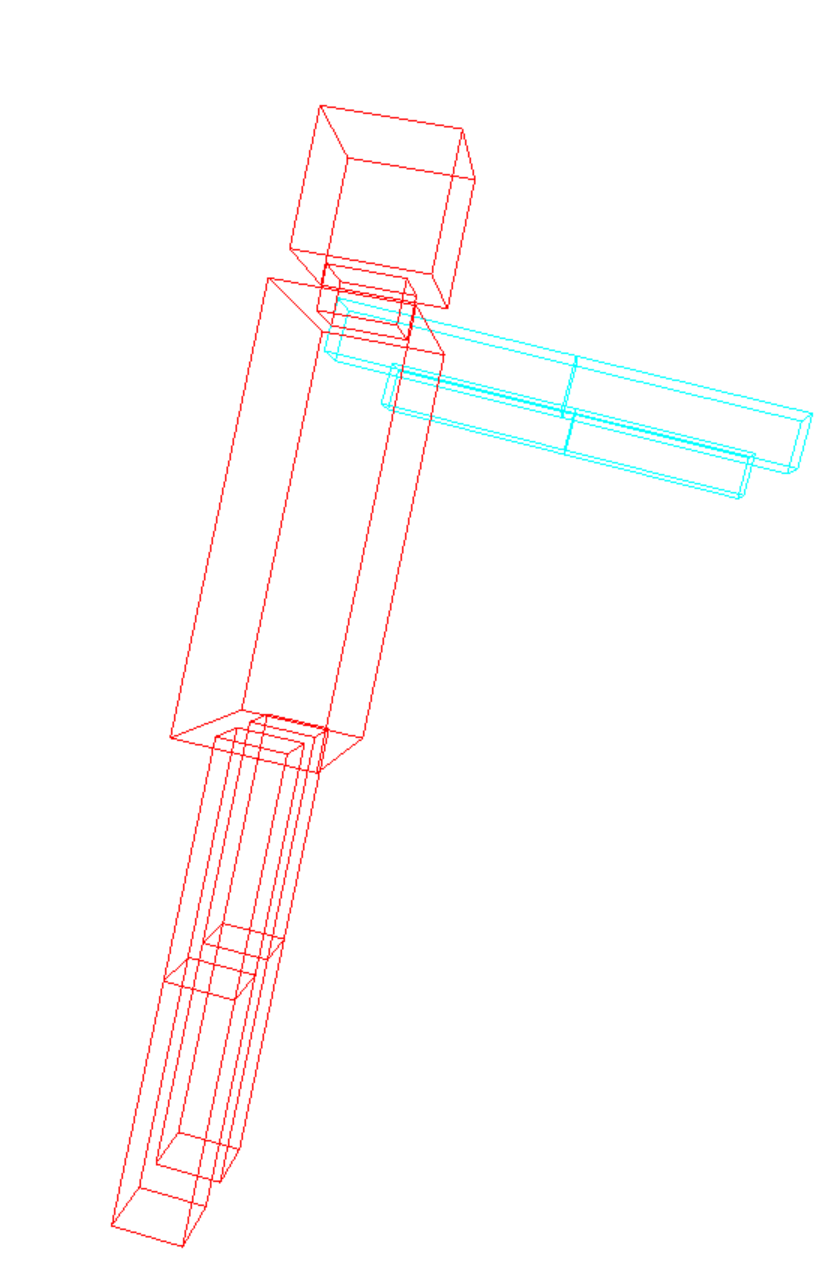
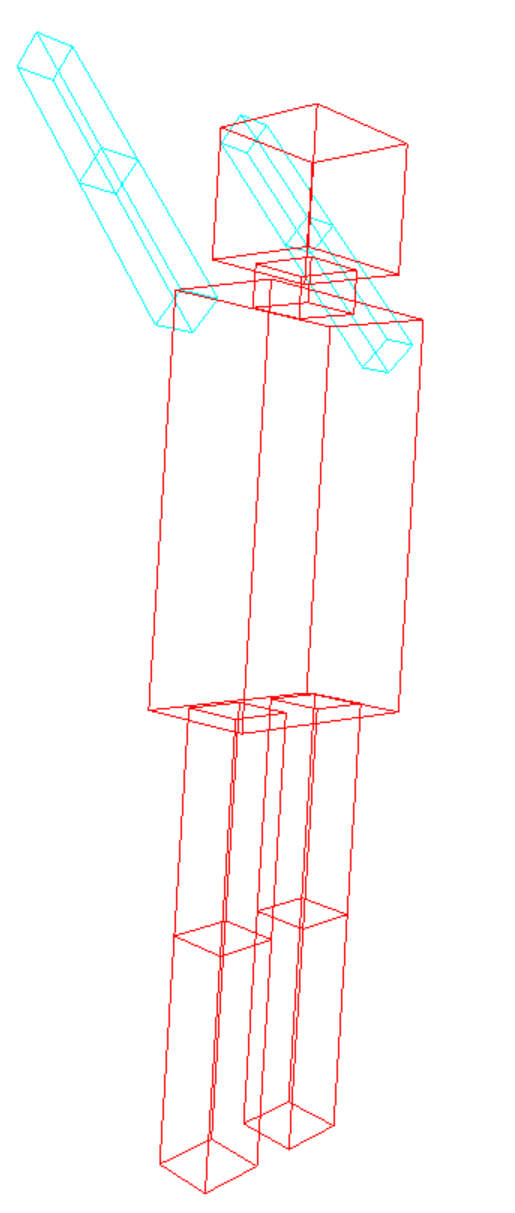
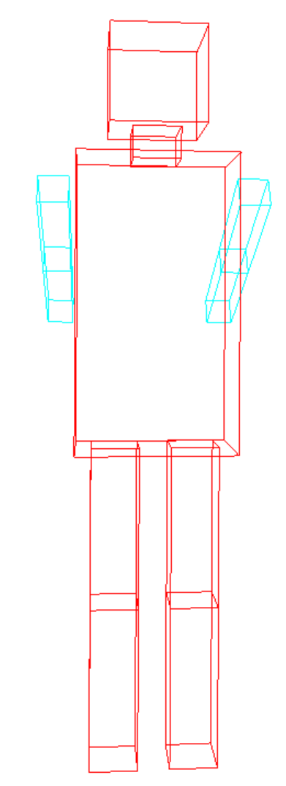


<프로그램 시작 후 키보드로 ‘q’키 눌렀을 경우>

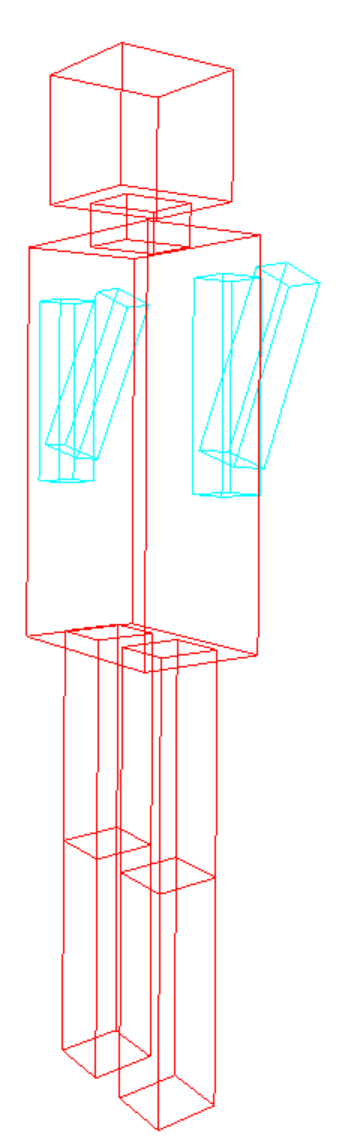
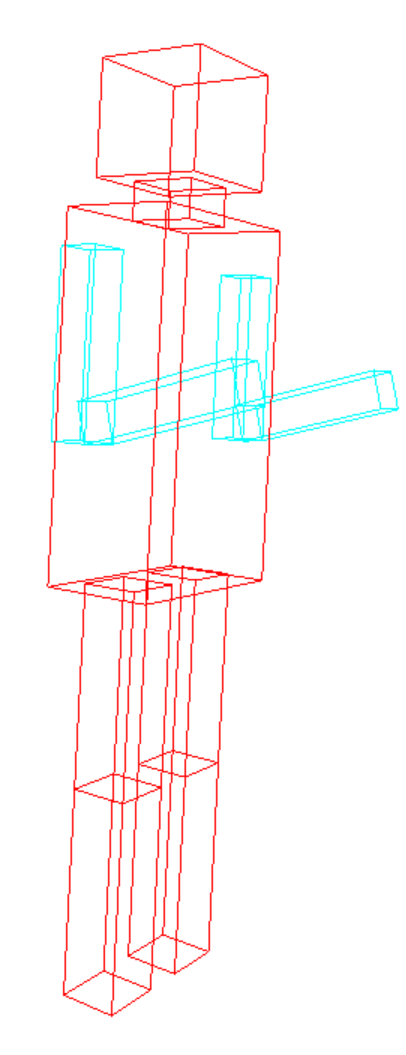
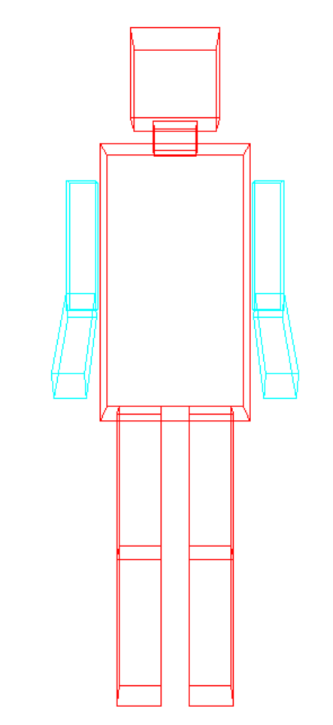




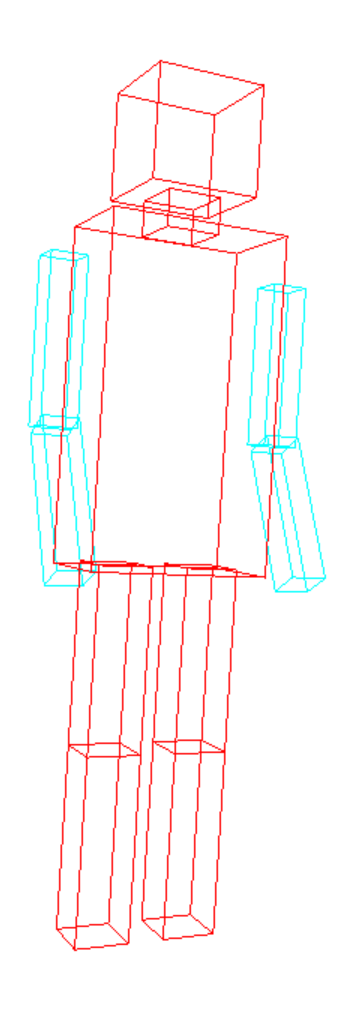
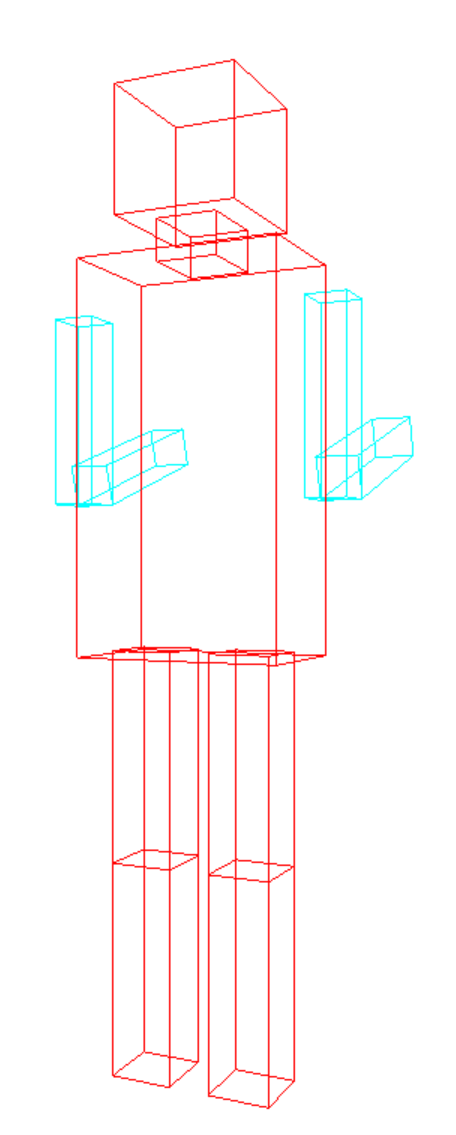
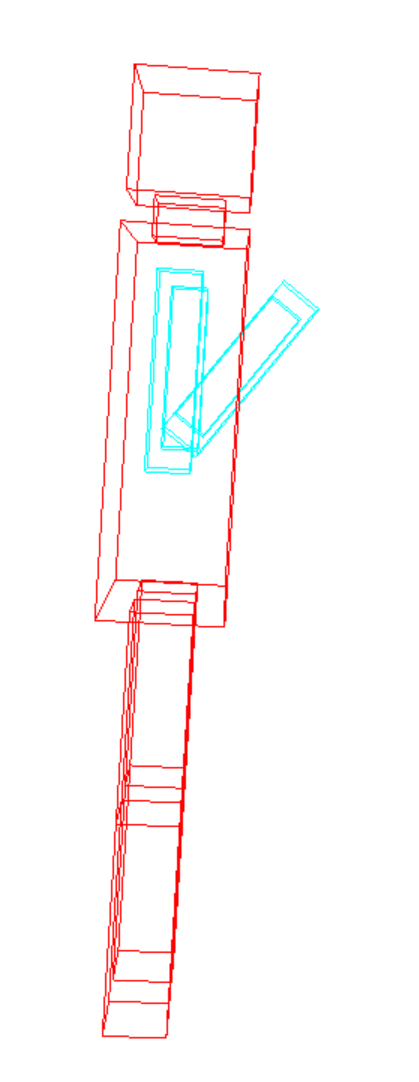
<프로그램 시작 후 키보드로 ‘a’키 눌렀을 경우 >

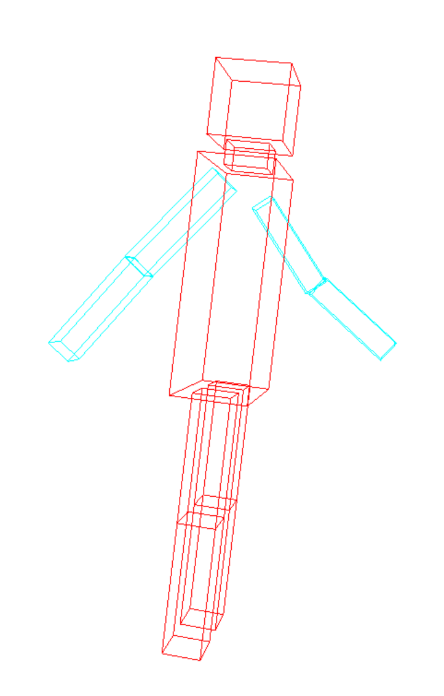
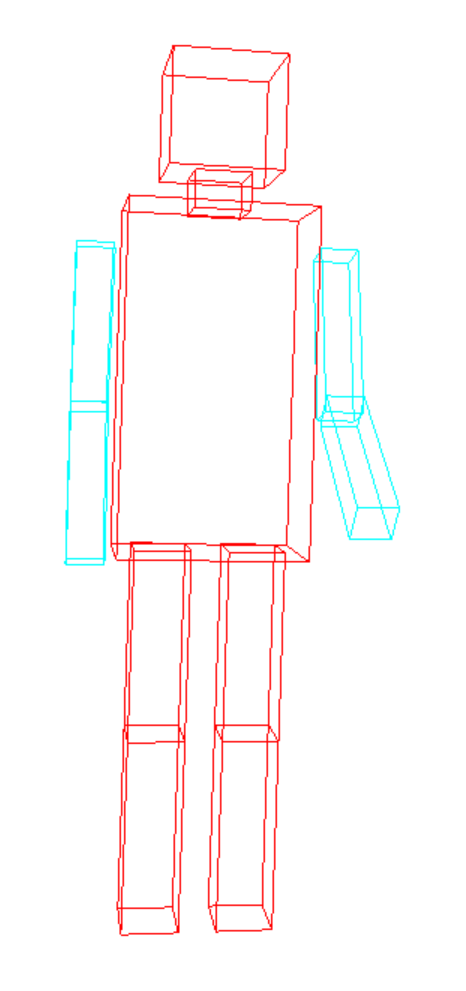
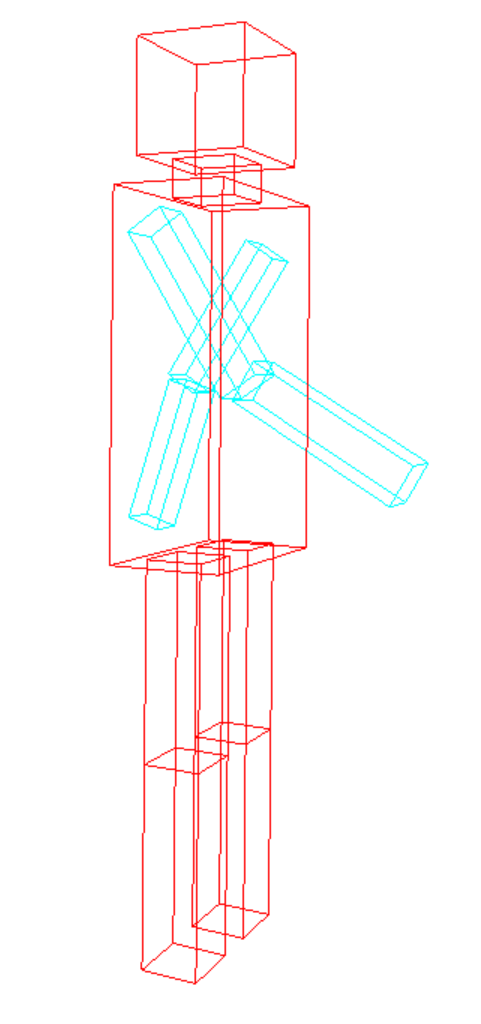


<프로그램 시작 후 키보드로 ‘p’키 눌렀을 경우>



<프로그램 시작 후 키보드로 ‘l’키 눌렀을 경우>

  
<프로그램 시작 후 키보드로 ‘w’키 눌렀을 경우>

* **소감**

중간고사 시험문제에도 나왔던 오렌지를 키워서 돌리는 예제에서 변형을 통해 로봇을 구현을 해보려고 하는 부분에서 좌표계를 설정해서 좌표를 미리 계산해 설정하는 것이 조금 어려웠던 부분이었습니다. 그렇지만 많은 시행착오가 있었지만 오른쪽다리와 오른팔을 몸통에 잘 설정하고 나서는 왼쪽 부분은 -부호만 해주면 됐기 때문에 로봇을 구현을 할 수 있게 되었습니다.

그리고 저는 2D가 아닌 3D로 구현 했기 때문에 카메라가 정면만 보고 있으면 로봇의 팔 움직임을 보여주는 부분에 있어서 힘듦이 있기 때문에 카메라를 자유자재로 돌리거나 로봇을 크기를 조절하고 위치를 다르게 하는 등 마우스 콜백과 키보드 콜백을 사용하여 로봇을 팔 움직임을 보다 정확하게 판단할 수 있게 만들어 주었습니다.

조교님께서 실습시간에 보여주신 팔 올리는 장면은 팔을 좌우로 벌려서 흔드는 모습이었습니다. 하지만 저는 로봇의 팔부분을 직육면체로 설정하였기에 좌우가 아닌 앞뒤로 흔드는 것으로 설정했습니다. 왜냐하면 좌우로 흔들게 될 경우 0도부터 시작해서 180도에서 멈추게 설정하면 만세 장면이 보여지게 되지만 로봇의 관절부부은 설정하지 않았으며 관절 부분들이 모두 직각이다 보니 앞뒤로 흔드는 움직임이 좀더 로봇다운 모습을 보여주는 것이라고 생각했기에 좌우보다는 앞뒤로 흔들게 구현했습니다.

컴퓨터 그래픽스에서 로봇 팔 움직임을 실제로 제가 구현을 함으로써 좌표계 설정과 좌표 값 계산, 콜백기능, opengl의 많은 명령어들을 배우고 활용할 수 있게 되었습니다. 시간이 없어 다리는 구현하지는 못했지만 걷는 모션에서 다리 부분도 구현해서github에 저장하고 싶습니다.