R E P O R T

컴퓨터그래픽스및실습

실습과제 06

과목명	컴퓨터그래픽스및실습
분반	01
교수	최 영 규
학번	2020136129
이름	최 수 연
제출일	2022년 10월 24일 월요일

문제 분석 및 해결 방법

[문제 분석]

- 강의자료 5장의 로봇 조립 프로그램 구현하기
- 모델 변환을 이용해 로봇을 조립하자.
- 동영상 참고: 그래픽스-05장-6 동영상
- 동영상에서와 같은 기본 기능 구현
- 자신이 만들거나 수정한 로봇을 이용하는 것을 권장함.
  - 기본 로봇 모델(수업에서 제공) 사용 → 10점(기본)
  - 기본 로봇을 간략히 수정(3DMax이용)하여 사용한 경우 → +2점
  - 완전히 새로운 멋진 모델을 만들어 사용한 경우 → 최대 +2점
  - 로봇을 부품으로 나누는 형태는 기본 모델과 동일하게 처리할 것

[해결 방법]

- 먼저 모델의 정보를 ASE 파일에서 가져오기 위해 파일을 열고 읽는 Mesh 클래스를 만든다.
- 또 모델을 그리는 draw 함수를 생성하여 Robot 클래스에서 로봇을 그릴 때, 해당 Mesh 클래스의 draw 함수를 사용해 그림을 그린다.
- MakeRobot.cpp 파일을 만들어 화면의 로봇을 그리고, 키보드, 마우스, 조명 등을 설정하는 콜백 함수를 등록한다.
- 추가로 좌표계 색상과 선, x, y, z 글자 입력 등의 좌표계를 설정하기 위한 함수를 만든다.

주요 설명 코드

<Mesh.h>

```
void clearAse() { // vertex, face, normal 리스트의 동적 할당 해제하는 함수
		if (nVtx != 0)
			delete[] vertex;
		if (nFace != 0) {
			delete[] face;
			delete[] normal;
		}
		nVtx = nFace = 0;
	}
```

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

	if ((fp = fopen(fileName, "r")) == NULL) { // NULL이면 파일을 찾을 수 없음
		cout << "File is Not Found" << endl;
		return;
	}
	while (fgets(line, 256, fp) != NULL) { // NULL일 때까지 반복
		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

			fgets(line, 256, fp); // *MESH_VERTEX_LIST
			sscanf(line, "%s", str);
			if (strcmp(str, "*MESH_VERTEX_LIST") == 0) { // *MESH_VERTEX_LIST 이게 맞으면
				for (int i = 0; i < nVtx; i++) { // vertex의 x, y, z 데이터 출력
					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_VERTEX_LIST
                        sscanf(line, "%s", str);
                        if (strcmp(str, "*MESH_FACE_LIST") == 0) {
                                for (int i = 0; i < nFace; i++) { // face의 vertex 인덱스 데이터 출력
                                        fgets(line, 256, fp); // Read the Line *MESH_VERTEX_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 '}'
                        fgets(line, 256, fp); // Read the Line *MESH_NUMCVERTEX
                        sscanf(line, "%s", str);
                        if (strcmp(str, "*MESH_NUMCVERTEX") == 0) { // 내용이 *MESH_NUMCVERTEX 가 맞으면
                                fgets(line, 256, fp); // 다시 *MESH_NORMALS를 위한 라인을 읽기
                                sscanf(line, "%s", str);
                        }
                        if (strcmp(str, "*MESH_NORMALS") == 0) { // normal 의 법선벡터 데이터 출력
                                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);
}
```

```
void draw(float scale = 1.0f, bool bCoord = false) { // 세 정점 주소를 가지고 삼각형을 그림
// 1.0f 크기에 맞게, bCoord가 false이므로 좌표계를 그리지 않고 삼각형 그림
                glBegin(GL_TRIANGLES);
                for (int i = 0; i < nFace; i++) {
                        Vertex* v1 = &vertex[face[i].vi[0]]; // 정점의 주소
```

```
                        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); // 비율에 맞게 그림
                        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) // bCoord가 true면 좌표계를 그리고, false면 좌표계를 그리지 않음
                        glkDrawCoord(1.0); // glk.cpp 파일의 glkDrawCoord 함수 호출
        }
```

<Robot.h>

```
void resize(bool flag = true) { // 로봇 크기를 다시 설정하는 함수
        if (flag) scale *= 1.05f; // Z를 누르면 원래 크기의 1.05배가 됨
        else scale *= 0.95; // z를 누르면 원래 크기의 0.95배가 됨
}
```

```
void draw() { // 로봇 모델을 행렬 스택을 사용해 그리는 함수
        Body.draw(0.5, 0.8, 0.8, scale, true); // 몸통 그리기

        glPushMatrix(); // 몸통 좌표계 저장
        glTranslated(0.0, -0.08, 0.0);
        glScalef(1.1f, 1.1f, 1.1f);
        Head.draw(0.8, 0.7, 0.7, scale); // 머리 그리기

        glPushMatrix(); // 머리 좌표계 저장
        glTranslated(0.0, -0.1, 0.0);
        glScalef(1.1f, 1.1f, 1.1f);
        Hat.draw(0.8, 0.7, 0.7, scale); // 모자 그리기
        glPopMatrix(); // 머리 좌표계로 돌아가기
        glPopMatrix(); // 몸통 좌표계로 돌아가기

        glPushMatrix(); // 몸통 좌표계 저장
        RightArm.draw(0.4, 0.4, 0.8, scale); // 오른쪽어깨 그리기

        glPushMatrix(); // 오른쪽어깨 좌표계 저장
        RightHand.draw(0.8, 0.8, 0.8, scale); // 오른손 그리기

        glPushMatrix(); // 오른손 좌표계 저장
        HammerHandle.draw(0.8, 0.8, 0.8, scale); // 해머 손잡이 그리기

        glPushMatrix(); // 해머 손잡이 좌표계 저장
```

```
        HammerHead.draw(0.8, 0.8, 0.8, scale); // 해머머리 그리기

        glPopMatrix(); // 해머 손잡이 좌표계로 돌아가기
        glPopMatrix(); // 오른손 좌표계로 돌아가기
        glPopMatrix(); // 오른쪽어깨 좌표계로 돌아가기
        glPopMatrix(); // 몸통 좌표계로 돌아가기

        glPushMatrix(); // 몸통 좌표계 저장
        LeftArm.draw(0.4, 0.4, 0.8, scale); // 왼쪽어깨 그리기

        glPushMatrix(); // 왼쪽어깨 좌표계 저장
        LeftHand.draw(0.8, 0.8, 0.8, scale); // 왼손 그리기
        glPopMatrix(); // 왼쪽어깨 좌표계로 돌아가기
        glPopMatrix(); // 몸통 좌표계로 돌아가기

        glPushMatrix(); // 몸통 좌표계 저장
        RightLeg.draw(0.4, 0.4, 0.8, scale); // 오른쪽다리 그리기

        glPushMatrix(); // 오른쪽다리 좌표계 저장
        RightFoot.draw(0.8, 0.8, 0.8, scale); // 오른발 그리기
        glPopMatrix(); // 오른쪽다리 좌표계로 돌아가기
        glPopMatrix(); // 몸통 좌표계로 돌아가기

        glPushMatrix(); // 몸통 좌표계 저장
        LeftLeg.draw(0.4, 0.4, 0.8, scale); // 왼쪽다리 그리기

        glPushMatrix(); // 왼쪽다리 좌표계 저장
        LeftFoot.draw(0.8, 0.8, 0.8, scale); // 왼발 그리기
        glPopMatrix(); // 왼쪽다리 좌표계로 돌아가기
        glPopMatrix(); // 몸통 좌표계로 돌아가기
}
```

<glkWin.cpp>

```
# include <windows.h>
char* glkFileDlg(const char* filter) // 파일 다이얼로그를 띄워 파일 가져오는 함수
{
        const int MaxLen = 1024; // 파일 전체 길이
        TCHAR fileName[MaxLen] = L""; // 파일을 읽어올 배열

        OPENFILENAME open_file; // OPENFILENAME 구조체 선언
        memset(&open_file, 0, sizeof(OPENFILENAME)); // 메모리 설정하는 함수
        open_file.lStructSize = sizeof(OPENFILENAME); // 구조체 크기
        open_file.hwndOwner = NULL; // 대화상자 소유자 없음
        open_file.lpstrFilter = L"ASE 3D data (*.ase)\0*.ase\0All(*.*)\0*.*\0"; // 파일 필터 설정
```

```
        open_file.nFilterIndex = 1; // 윗줄의 파일 형식 콤보박스에서 사용할 필터의 인덱스 지정
        open_file.lpstrFile = fileName; // 사용자가 선택한 파일의 경로
        open_file.nMaxFile = MaxLen; // lpstrFile의 길이
        open_file.lpstrTitle = L"Select a file"; // 대화 상자의 캡션을 지정
        open_file.lpstrDefExt = L"ASE"; // 디폴트 확장자 지정
        open_file.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST | OFN_HIDEREADONLY;
       // 사용자 지정 경로 및 파일 이름이 존재하는지 확인, "읽기 전용으로 열기" 체크박스를 숨김
        bool ret = GetOpenFileName(&open_file); // 파일 열기 대화상자 호출 값 bool에 저장

        return (ret) ? (char*)fileName : NULL; // ret이 true면, 파일 경로 반환
}
```

<glk.h>

```
// glk.cpp 파일의 함수를 사용하기 위해 extern 변수 선언
extern void glkString(const char* s);
extern void glkSetColor(float r, float g, float b, float a);
extern void glkDrawCoord(double len);

// ASE, BVH 파일 형식 정의
#define FILTER_ASE "ASE 3D data (*.ase)\0*.ase\0A11 (*.*)\0*.*\0"
#define FILTER_BVH "BVH Motion Data (*.bvh)\0*.bvh\0A11 (*.*)\0*.*\0"
#ifdef WIN32
extern char* glkFileDlg(const char*); // glkWin.cpp 파일의 함수를 사용하기 위해 extern 변수 선언
#endif
```

<glk.cpp>

```
// 어떤 문자열을 화면에 그리는 함수, glkDrawCoord 함수에서 좌표계 x, y, z 문자를 화면에 출력할 때 사용
void glkString(const char* s) {
        unsigned int i;
        for (i = 0; i < strlen(s); i++)
                glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, s[i]);
}
```

```
// 재질, 색상, 빛 설정하는 함수
void glkSetColor(float r, float g, float b, float a) {
        float color[4] = { r, g, b, a };
        glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, color); // 겉면 속면 모두 표현, 발산광 조명 사용
        glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, color); // 겉면 속면 모두 표현, 주변광 조명 사용
}
```

```
// 좌표계 그리기. 원점에서 +x(red), +y(blue), +z(gray)로 길이 len인 선분
void glkDrawCoord(double len) {
        glDisable(GL_LIGHTING); // 조명 사용 설정
        glColor3f(1.0f, 0, 0);
        glkLine(0, 0, 0, len, 0, 0); // RED ==> +x axis
        glRasterPos3f(len, 0, 0); // 문자열 x가 그려질 위치 좌표 지정
        glkString("x"); // 화면에 x 그림
```

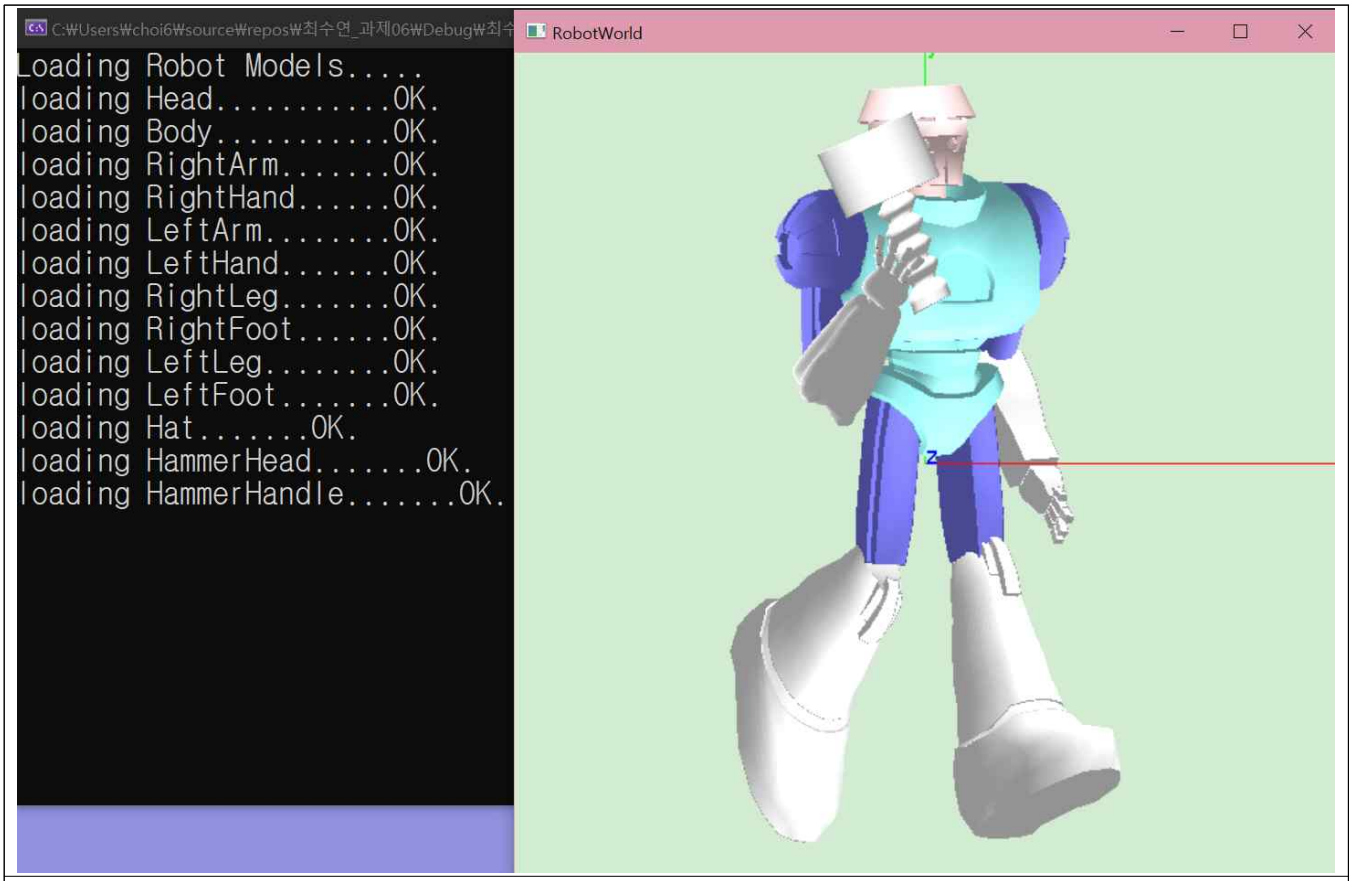
```
        glColor3f(0, 1.0f, 0);
        glkLine(0, 0, 0, 0, len, 0);  // GREEN ==> +y axis
        glRasterPos3f(0, len, 0); // 문자열 y가 그려질 위치 좌표 지정
        glkString("y"); // 화면에 y 그림

        glColor3f(0, 0, 1.0f);
        glkLine(0, 0, 0, 0, 0, -len);  // BLUE ==> +z axis
        glRasterPos3f(0, 0, -len); // 문자열 z가 그려질 위치 좌표 지정
        glkString("z"); // 화면에 z 그림
        glEnable(GL_LIGHTING); // 조명 사용 설정
}
```

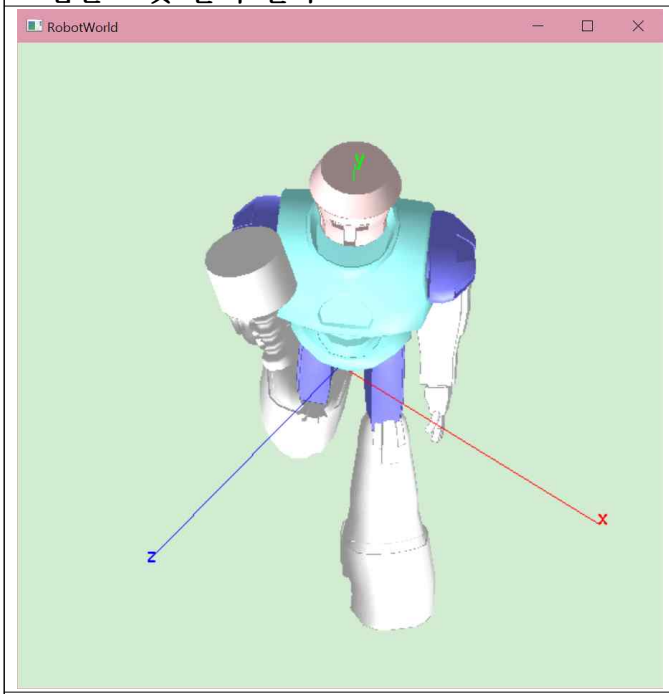
<MakeRobot.cpp>

```
void keyboard(unsigned char key, int x, int y) { // 키보드 버튼을 통해 특정 기능들을 사용하는 함수
        /*if (key == 'l') { // 파일 다이얼로그 열기
                char* filename = glkFileDlg(FILTER_ASE);
                if (filename != NULL)
                        obj3D.readAse(filename);
        }
        else*/ if (key == 'i') { // 단위행렬로 초기화
                glMatrixMode(GL_MODELVIEW);
                glLoadIdentity();
        }
        else if (key == 'w') { // 모델 외부 표면을 선으로 채우고, 내부 표면을 점으로 채움
                glPolygonMode(GL_FRONT, GL_LINE); // 폴리곤모드: 도형이 그려지는 모드
                glPolygonMode(GL_BACK, GL_POINT);
        }
        else if (key == 's') { // 모델 외부 표면을 모두 채우고, 내부를 선으로 채움
                glPolygonMode(GL_FRONT, GL_FILL);
                glPolygonMode(GL_BACK, GL_LINE);
        }
        else if (key == 'z' || key == 'Z') { // z를 누르거나 Z를 누르면 크기 변환
                robot.resize(key == 'z'); // Robot 클래스의 resize 함수 사용
        }
        else if (key == 'q') // q를 누르면 실행 종료
                exit(0);
        glutPostRedisplay(); // 화면에 모델을 다시 그리도록 요청
}
```

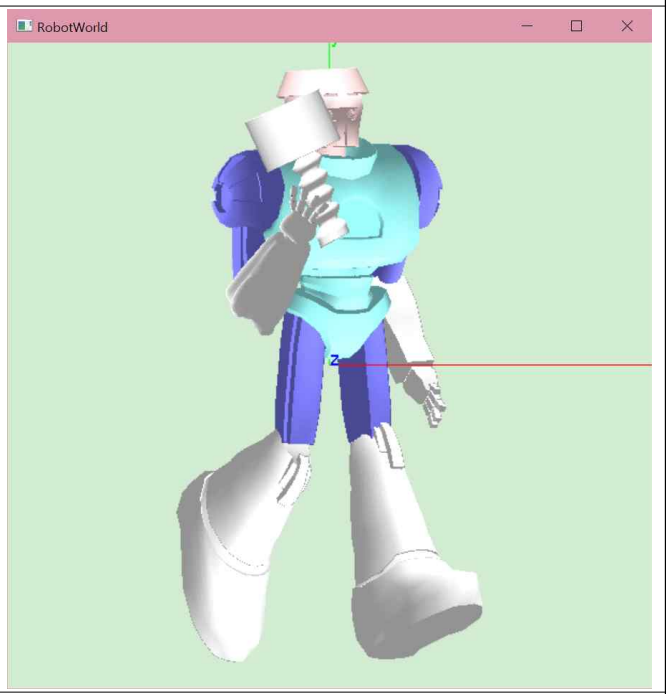
테스트 결과



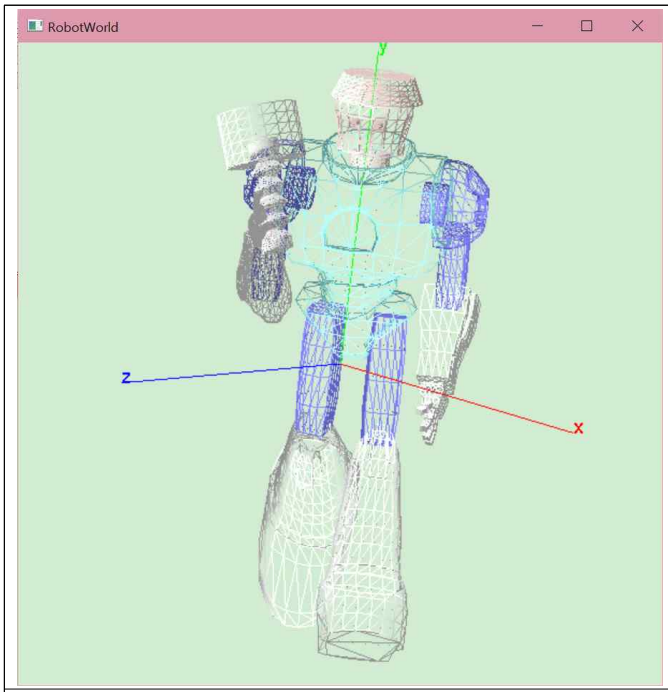
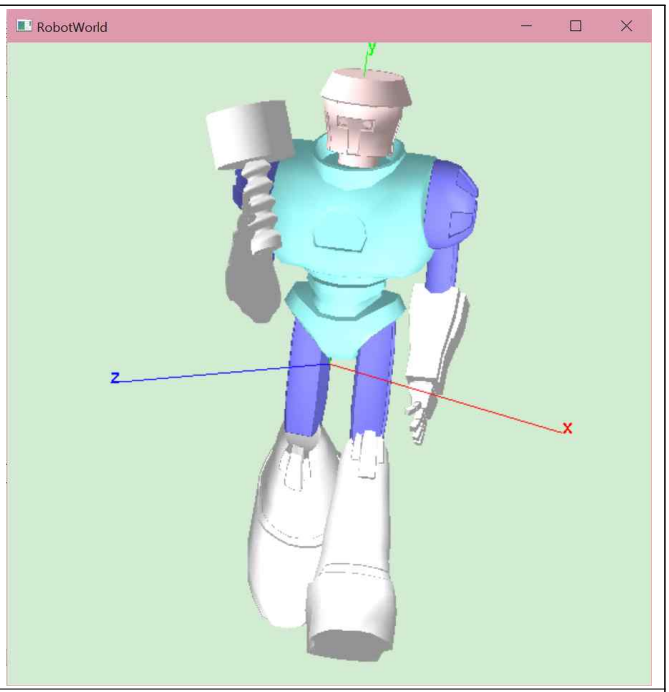
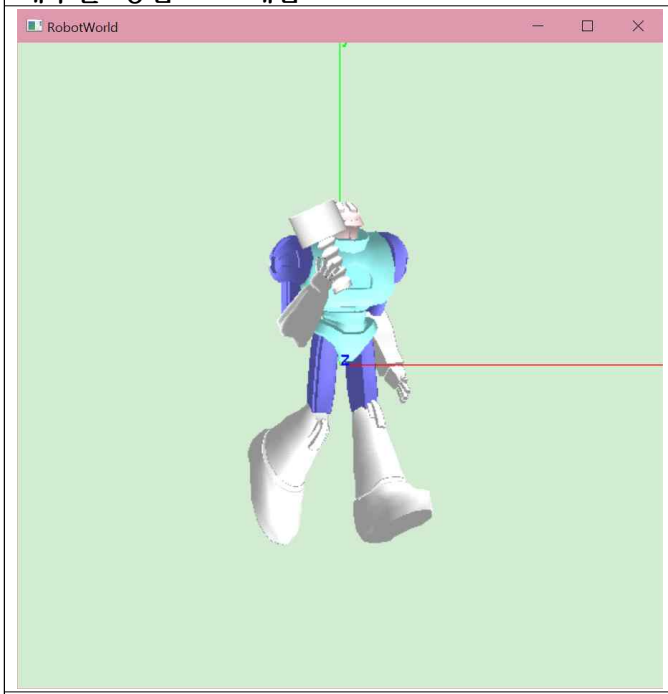
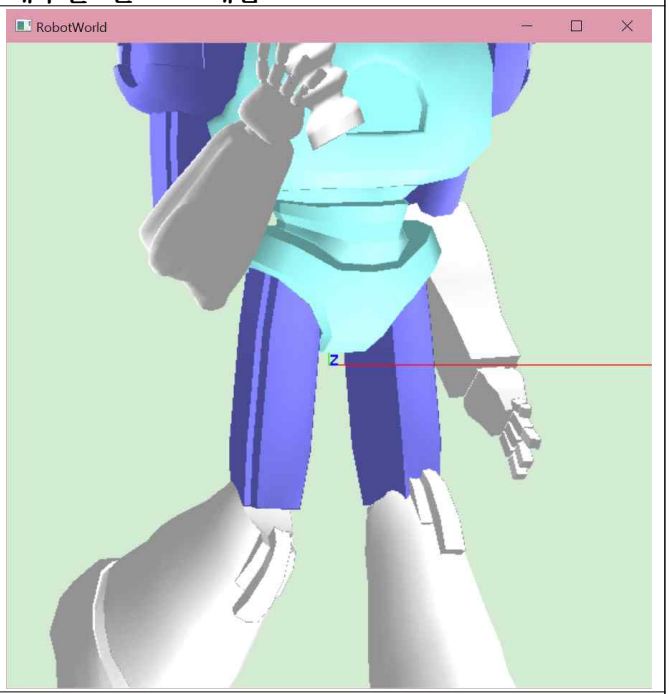
조립된 로봇 출력 결과



마우스로 화면 움직임



i키를 눌러 원래 화면 각도로 리셋

	
w키를 눌러 모델 외부 표면을 선으로 채우고 내부를 정점으로 채움	s키를 눌러 모델 외부 표면을 채우고 내부를 선으로 채움
	
z키를 눌러 모델 축소	Z키를 눌러 모델 확대

고찰 및 느낀점

이번에 로봇을 조립하는 과제가 지금까지 했던 과제 중에 제일 흥미롭게 했던 것 같다. 사실 처음에는 glkWin.cpp 파일에서 형식 지정자 부분이 계속 오류 나서 시간을 많이 잡아먹었지만, 본 과제에서는 부품별로 확인하는 것이 아니라 로봇 각 부품을 조립해서 하나의 전역 좌표계로 보여주는 것이기 때문에, 과제 수행하는 데 큰 어려움은 없었다. 그러나 위 오류뿐만 아니라 다른 시행착오가 있었는데, 3ds Max에서 ASE 파일을 export 했는데, 처음에 3ds Max에서 그렸던 모델이 3ds Max에 있는 좌표계 중심에 있지 않아서, export 했을 때 그것이 반영되어 OpenGL에 제대로 모델이 나타나지 않고 구석에 치우쳐 있었다. 그래서 3ds Max에서 다시 내 로봇을 그룹화하여 몸통을 중심으로 좌표 (0,0,0)으로 위치를 수정한 다음에 다시 export 했더니 OpenGL에서 모델이 좌표계에 맞게 잘 출력되었다. 또, 한 가지 아쉬운 점은 모델 색을 원래 3ds Max에서 수정했던 색으로 OpenGL에 반영하지 못했다는 것이다. 이 부분에 대해서는 제대로 찾아보지 못해서 시험 기간 끝나고 한 번 다시 시도해봐야 할 것 같다.