Final Term Project 레포트

작성일	2022.06.15	전공	컴퓨터소프트웨어공학
작성자	김주원	학번	20184009
강의명	컴퓨터 그래픽스	강의자	홍 민 교수님

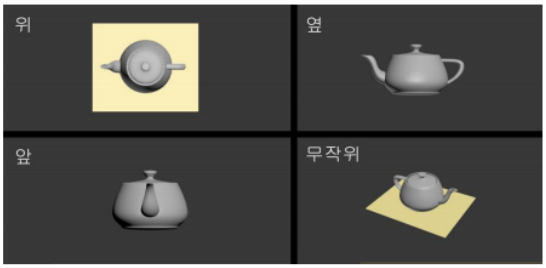
| 목차 |

1. 03 과제

기말 프로젝트

본인이 제작한 3D 물체를 obj 파일을 생성하여 읽어온 후 아래와 같은 기능을 메뉴를 이용하여 제공하는 프로그램을 작성: 6월 15일(수) 제출, 소스코드 및 레포트 제출

- 파일 읽기: 3D 물체 obj
- 카메라 이동: keyboard 화살표 키 활용(줌-인, 줌-아웃, 좌, 우)
- Smooth, Flat shading
- 조명 설치
- 3개 이상의 다른 재질 반영
- Texture 적용



1-1 문제분석

obj 파일에 오브젝트 데이터의 저장 방식으로 데이터를 읽어와 기존 오브젝트의 형태로 출력되게 만들고, 입혀 놓았던 PNG 형태로 저장된 텍스쳐 파일을 읽어와 오브젝트에 입혀야 한다. 여러 방향으로 4 개의 뷰포트에 출력해야 한다.

렌더링 된 모델을 관찰하는 카메라를 왼쪽, 오른쪽 방향키를 이용하여 회전하게 만들어 오브젝트를 관찰할 수 있어야 한다. 위, 아래 방향키를 입력하면 zoom in, zoom out 이 작동해야 한다.

우클릭을 누르면 팝업 메뉴가 생기고, 생성된 팝업 메뉴에는 3 가지의 재질을 바꿀 수 있는 버튼, 조명 on/off 버튼, shande 모드 선택, 텍스쳐 on/off 기능이 있어야 한다.

1-2 소스 코드 분석

```
1   /*
2           작성자 : 20184009 김주원
3           작성일 : 22.05.23 ~ 22.06.15
4           프로그램 명 : Model Viewer
5           프로그램 설명 : obj파일의 모델을 읽어와 4개의 시점으로
6                           출력하고, 회전, 줌, 재질 변경, 조명의
7                           기능을 이용하여 모델을 볼 수 있는 프로그램이다.
8   */
9   #include <GL/glut.h>
10  #include <iostream>
11  #include <math.h>
12  #include <vector>       //텍스쳐 파일의 이름을 저장하기 위한 배열의 헤더파일
13  #include "MeshFactory.h"    //obj파일을 불러오는 헤더파일
14  #include "ObjMesh.h"        //오브젝트의 정보를 담는 클래스
15  #include "lodepng.h"        //PNG파일인 텍스쳐 파일을 읽는 함수
16
17   using namespace std;
```

메인 파일인 temp.cpp 파일이다. 프로그램에 관련된 설명을 주석으로 작성하였다. 텍스쳐 파일의 데이터를 저장하기 위한 vector 헤더 파일을 추가하였고, obj 파일의 데이터를 저장하기 위한 클래스가 저장된 헤더파일 ObjMesh.h, Obj 파일에서 데이터를 읽어 저장하는 MeshFactory.h, 텍스쳐가 저장된 PNG 파일을 읽어 저장하기 위한 헤더파일 lodepng.h 헤더 파일을 추가해주었다.

```
19      void Display();     //display함수의 콜백 함수
20      void Reshape(int w, int h);     //reshape함수의 콜백 함수
21      void Keyboard(int key, int x, int y);   //keyboard함수의 콜백 함수
22      void Grid();    //발판을 그리는 함수
23      void PopupMenu(int index);  //팝업메뉴 함수의 콜백 함수
24      void InitL();   //조명을 초기화하는 함수
25      void loadTexture(GLuint * texture, const char *path);//PNG파일에서 텍스쳐를 불러오는 함수
26      void spin_Display(int key); //카메라를 돌리는 함수
27      void MyKeyboard(unsigned char KeyPressed, int X, int Y);    //special함수의 콜백 함수
```

display 함수의 콜백함수 Display()와 reshape 함수의 콜백함수 Reshape(), special keyboard 함수의 콜백함수 keyboard(), 발판을 그리는 Grid()함수, 팝업 메뉴의 콜백함수인 PopipMenu(), 조명을 초기화하는 함수 InitL(), 텍스쳐 파일인 png 파일을 읽어 텍스쳐의 데이터를 저장하는 함수 loadTexture(), 카메라를 돌리는 함수 spin_Display() 그리고 keyboard 함수의 콜백 함수인 MyKeyboard()를 선언하였다.

```
29      float angle = 0.0;
30
31      bool light = true;      //조명 활성화 토글 변수
32      bool texture = true;    //텍스쳐 활성화 토글 변수
33      bool line = true;
34
35      GLint menu;         //팝업 메뉴 변수
36      ObjMesh objMesh;    //오브젝트 클래스의 객체
37      GLint w, h;         //원근 투영의 종횡비 값
```

카메라의 각도를 나타내는 angle 변수와 조명의 on/off 를 결정하는 light 변수, 텍스쳐의 활성, 비활성을 결정하는 texture 변수, 오브젝트의 테두리 라인의 출력을 결정하는 line 변수가 있다.

팝업 메뉴가 될 변수 menu, 오브젝트 데이터가 저장되는 클래스 Object 의 객체인 object 와 원근 투영의 인자 값인 종횡비 값이 될 w, h 를 선언하였다.

```
39      //camera
40      float fov = 60;      //원근 투영의 시야각
41      GLfloat light_position[] = { 10.0, 10.0, 10.0, 0.0 };   //조명위치
42
43
44      char f[] = "dsa.obj";   //obj파일의 이름
45      std::vector<unsigned char> image2;  //텍스쳐의 데이터를 저장하는 벡터
46      GLuint textureID[1];    //텍스쳐 파일의 이름
```

원근 투영의 시야각인 fov 이다. Zoom in, Zoom out 에 사용되는 변수이다.
image2 변수는 텍스쳐 파일의 데이터가 저장되는 vector 자료형 변수이다. textureID 변수는 텍스쳐 파일의 이름을 저장할 변수이다.

```
48      void Init(){
49          CMeshFacotry::LoadObjModel(f, &objMesh);
50
51          glDepthFunc(GL_LESS); // Depth 버퍼 설정
52          glEnable(GL_DEPTH_TEST); // Depth Testing 활성화
53
54          glClearColor(0.0, 0.0, 0.0, 1.0); //윈도우 배경 설정
55
56          InitL();    //조명 초기화
57          loadTexture(&textureID[0], "Texture.png");  //텍스쳐 파일을 불러옴
58      }
```

여러 기초적인 작업을 통해 프로그램을 초기화하는 함수이다. obj 파일의 이름이 저장된 변수 f로 파일을 찾아 오브젝트의 객체변수인 objMesh 변수에 오브젝트의 데이터를 저장한다.

Depth 버퍼의 비교 연산을 GL_LESS 로 설정한다. 그리고 조명을 초기화 한 후에, loadTexture 의 두번째 인자 값으로 Textrue.png 를 넣어 텍스쳐의 데이터를 읽어 image2 변수에 저장한다.

```
60      void InitL()    //조명 초기화
61      {
62          GLfloat light_ambient[] = { 1.0, 1.0, 1.0, 1.0 };   // 주변광
63          GLfloat light_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };   // 확산광
64          GLfloat light_specular[] = { 1.0, 1.0, 1.0, 1.0 };  // 경면광
65
66          // 조명 설정
67          glLightfv(GL_LIGHT0, GL_POSITION, light_position);
68          glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
69          glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
70          glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
71
72          glEnable(GL_LIGHTING);  //조명을 활성화 시킴
73          glEnable(GL_LIGHT0);    //사용할 조명의 번호로 설정
74      }
```

조명을 초기화하는 함수이다. 주변광, 확산광, 경면강의 값을 배열에 저장하고 저장한 배열로 각각의 조명의 성질을 설정하였다. 그리고 조명 0 번을 사용하도록 하였다.

```
76  void loadTexture(GLuint * texture, const char *path) {  //PNG파일에서 텍스쳐를 불러오는 함수
77      std::vector<unsigned char> image;
78      unsigned width, height;
79      unsigned error = lodepng::decode(image, width, height, path);   //텍스쳐 파일 decoding
80      if (!error)     //문제가 있을 경우 경고
81          std::cout << "error" << error << ": " << lodepng_error_text(error) << std::endl;
82      size_t u2 = 1; while (u2 < width) u2 *= 2;
83      size_t v2 = 1; while (v2 <height) v2 *= 2;      //파일 크기 파악
84      image2 = std::vector<unsigned char>(u2*v2 * 4); //파일의 사이즈에 4를 곱해 크기 할당
```

Path 로 읽어온 문자열과 일치하는 이름의 텍스쳐 파일을 불러와서 저장하는 함수이다. vector 자료형 image 에 텍스쳐 파일의 데이터를 저장하고, 높이와 넓이를 height, width 에 저장한다. 그리고 일반적인 파일의 크기인 2 의 제곱 형태로 u2, v2 에 높이와 넓이를 구하고 4 를 곱해 전역 변수인 image2 에 크기를 할당한다.

```
85      for (size_t y = 0; y < height; y++)
86          for (size_t x = 0; x < width; x++)
87              for (size_t c = 0; c < 4; c++) {    //파일의 데이터를 복사함
88                  image2[4 * u2 * y + 4 * x + c] = image[4 * width * y + 4 * x + c];
89              }
90      glGenTextures(1, texture);      //텍스쳐 이름 생성
91      glBindTexture(GL_TEXTURE_2D, *texture);     //명명된 텍스쳐를 2D로 생성
92      glTexImage2D(GL_TEXTURE_2D, 0, 4, u2, v2, 0, GL_RGBA, GL_UNSIGNED_BYTE, &image2[0]);    //텍스쳐 객체에 이미지 설정
93      glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
94      glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);      //2D인 텍스쳐를 3D에 맞춰 매핑시킴
95  }
```

텍스쳐의 데이터를 처음 가져왔던 image1 에서 프로그램에서 실제로 쓰일 전역변수 image2 에 값을 복사해준다. 그리고 텍스쳐의 이름을 생성해주고 2D 로 생성해준다. 마지막으로 glTexParameter 함수로, 생성된 2D 텍스쳐를 3D 형태로 매핑해주었다.

```
97  int main(int argc, char **argv){
98      glutInit(&argc, argv);      //GLUT초기화
99      glutInitDisplayMode(GLUT_DOUBLE | GLUT_DEPTH);  //두개 버퍼 사용
100     glutInitWindowSize(810, 810);
101     glutCreateWindow("OBJ file Loader");
102
103     Init();
```

메인 함수이다. GLUT 을 초기화 해주었고 버퍼를 두개로 설정해주고 3 차원 형태의 출력을 위해 깊이 버퍼를 사용한다.

```
105     //콜백 함수 적용
106     glutSpecialFunc(Keyboard);      //ESC키 입력 함수
107     glutKeyboardFunc(MyKeyboard);   //화살표 키 입력 함수
108     glutDisplayFunc(Display);
109     glutReshapeFunc(Reshape);
```

콜백 함수를 적용하였다.

```
111         menu = glutCreateMenu(PopupMenu);   //팝업 메뉴 설정
112         glutAddMenuEntry("Light", 0);       //빛 on/off
113         glutAddMenuEntry("Texture", 1);     //텍스쳐 on/off
114         glutAddMenuEntry("material1", 2);   //Meterial 설정
115         glutAddMenuEntry("material2", 3);
116         glutAddMenuEntry("material3", 4);
117         glutAddMenuEntry("Smooth", 5);      //Shading 설정
118         glutAddMenuEntry("Flat", 6);
119         glutAddMenuEntry("Line_draw", 7);   //선 on/off
120         glutAttachMenu(GLUT_RIGHT_BUTTON);
121
122         glutMainLoop();
123
124         return 0;
125     }
```

팝업 메뉴를 설정해준다. Menu 변수에 8 개의 팝업이 설정된다.

```
127     int Mat_N = 0;
128     int shadmodel = 0;
129
130     GLfloat mat1[4] = { 0.0215, 0.1745, 0.0215, 1.0 };  //Material값
131     GLfloat mat2[4] = { 0.135, 0.2225, 0.1575, 1.0 };
132     GLfloat mat3[4] = { 0.05375, 0.05, 0.06625, 1.0 };
```

Material 의 번호 변수와 shademodel 설정 변수를 정의하였고, Meterial 의 값이 될 변수 3 개를 정의하였다.

```
134     void Display(void)      //display함수의 콜백 함수
135     {
136
137         glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); //color, depth 버퍼 초기화
138
139         glMatrixMode(GL_MODELVIEW);
140         glLoadIdentity();
```

버퍼를 초기화 시켜주었다. 모델뷰 행렬을 가져와 주었다.

```
142         if (texture) glEnable(GL_TEXTURE_2D);   //텍스쳐 활성화/비활성화
143         else glDisable(GL_TEXTURE_2D);
144
145         if (light) glEnable(GL_LIGHTING);       //조명 활성화/비활성화
146         else glDisable(GL_LIGHTING);
```

Texture 와 light 토글 변수가 현재 설정된 값에 따라 텍스쳐와 조명을 활성화 할지 정하게 된다.

```
148         switch (shadmodel) {    //Shading 설정
149         case 0:
150             glShadeModel(GL_SMOOTH);
151             break;
152         case 1:
153             glShadeModel(GL_FLAT);
154             break;
155         }
```

shadmodel 변수의 현재 값으로 shading 의 모델이 설정된다.

```
157     switch (Mat_N) {    //Material 설정
158     case 1:
159         glMaterialfv(GL_FRONT, GL_AMBIENT, mat1);
160         break;
161     case 2:
162         glMaterialfv(GL_FRONT, GL_AMBIENT, mat2);
163         break;
164     case 3:
165         glMaterialfv(GL_FRONT, GL_AMBIENT, mat3);
166         break;
167     }
```

팝업으로 선택된 Mat_N 의 값에 따라 Material 이 바뀐다.

```
169     int i = 1;
170     while (i < 5) {
171         switch (i) {
172         case 1: //1사분면
173             glPushMatrix();
174
175             glMatrixMode(GL_PROJECTION);
176
177             glLoadIdentity();
178             gluPerspective(fov, (GLfloat)w / (GLfloat)h, 0.1, 100000.0);
179
180             gluLookAt(35, 15, 0, 0, 6, 0, 0.0, 1.0, 0.0);   //카메라 위치 설정
181             glViewport(410, 410, 400, 400);     //오른쪽 위 뷰포트 생성
```

i 값이 증가하며 각각 다른 viewport 를 4 개 생성하여 서로 다른 출력이 나타나게 된다. i 가 1 일 때는

오른쪽 위에 뷰포트가 생성된다. 그리고 카메라가 오브젝트의 우측을 비추게 위치시켰다.

```
183             glPushMatrix();
184             glRotatef(angle, 0, 1, 0);  //카메라 회전
185             glTranslatef(0, 0, 143);
186             objMesh.Render();   //오브젝트 출력
187
188             glPopMatrix();
189
190             glPopMatrix();
191
192             break;
```

위에 작업했던 것들을 push 하여 저장하고 현재 각도에 따라 오브젝트를 생성하고 오브젝트가 카메라쪽으로 위치하도록 이동시켰다. 그리고 렌더링 함수를 통해 출력시켰다. 마지막으로 저장된 스택을 pop 하여 정상적으로 그려지도록 만들었다.

```
194             case 2: //2사분면
195                 glPushMatrix();
196 
197                 glMatrixMode(GL_PROJECTION);
198 
199                 glLoadIdentity();
200                 gluPerspective(fov, (GLfloat)w / (GLfloat)h, 0.1, 100000.0);
201 
202                 gluLookAt(0, 120, 5, 0, 6, 0, 0.0, 1.0, 0.0);
203                 glViewport(0, 410, 400, 400);   //왼쪽 위 뷰포트 생성
204 
205                 glPushMatrix();
```

i 갑이 증가하여 다음 뷰포트를 생성한다. 뷰포트의 위치는 오른쪽 아래이다. 그리고 전의 뷰포트와는 다른 카메라의 위치를 설정하여 오브젝트의 위에서 비추도록 만들었다.

```
207                 glRotatef(angle, 0, 1, 0);
208 
209                 Grid();     //발판 생성
210 
211                 glScalef(2.7f, 2.7f, 2.7f);     //크기 설정
212                 glTranslatef(0, 0, 143);
213                 objMesh.Render();
214 
215                 glPopMatrix();
216 
217                 glPopMatrix();
218 
219                 break;
```

그리고 발판을 Grid()함수로 그려주었고, 카메라가 멀리 이동하여 작게 보이는 오브젝트의 크기를 확대시켰다.

```
221             case 3: //3사분면
222                 glPushMatrix();
223 
224                 glMatrixMode(GL_PROJECTION);
225 
226                 glLoadIdentity();
227                 gluPerspective(fov, (GLfloat)w / (GLfloat)h, 0.1, 100000.0);
228 
229                 glViewport(0, 0, 400, 400);     //왼쪽 아래 뷰포트 생성
230                 gluLookAt(0.0f, 25.75f, 80.0f, 0, 25.75f, 0, 0.0, 1.0, 0.0);
231 
232                 glPushMatrix();
```

세번째 뷰포트이다. 왼쪽 아래에 생성된다. 오브젝트의 정면을 비추도록 만들었다.

```
234                 glScalef(2.7f, 2.7f, 2.7f);
235                 glRotatef(angle, 0, 1, 0);
236                 glTranslatef(0, 0, 143);
237                 objMesh.Render();
238 
239                 glPopMatrix();
240 
241                 glPopMatrix();
242 
243                 break;
```

카메라가 위로 이동하여 작게 보이는 오브젝트의 크기를 확대시켜주었다.

```
245             case 4: //4사분면
246                 glPushMatrix();
247 
248                 glMatrixMode(GL_PROJECTION);
249 
250                 glLoadIdentity();
251                 gluPerspective(fov, (GLfloat)w / (GLfloat)h, 0.1, 100000.0);
252 
253                 gluLookAt(-70, 50, 80, 0, 6, 0, 0.0, 1.0, 0.0);
254                 glViewport(410, 0, 400, 400);       //오른쪽 아래 뷰포트 생성
255 
256                 glPushMatrix();
257 
258                 glRotatef(angle, 0, 1, 0);
259                 Grid();
```

4 번째 뷰포트이다. 오른쪽 아래에 뷰포트가 생성된다. 여기서 Perspective 함수에 fov 라는 변수가 들어가는데 fov 는 원근투영의 시야각이다. 이 시야각을 증가시키거나 감소시키면 촬영하는 화면의 크기가 증가하거나 감소하기 때문에, 이를 이용하여 오브젝트를 확대하거나 감소하는 효과를 줄 수 있다. 따라서 변수로 인자를 넣어 zoom in, zoom out 의 동작을 구현하였다.

그리고 rotatef 의 인자로 angle 값을 넣어 angle 값에 따라 오브젝트가 다른 각도로 돌아가게 출력하였다. 이는 카메라가 움직여 오브젝트를 촬영하는 것과 같은 동작이기 때문에 이것으로 카메라의 회전을 구현하였다.

```
261                 glScalef(2.7f, 2.7f, 2.7f);
262                 glTranslatef(0, 0, 143);
263                 objMesh.Render();
264 
265                 glPopMatrix();
266 
267                 glPopMatrix();
268 
269                 break;
270             }
271             i++;
272         }
273         glutSwapBuffers(); //버퍼 swap
274     }
```

카메라가 이동하여 크기가 작아진 오브젝트의 크기를 조정하였고, 마지막으로 버퍼를 swap 하여 변한 상태를 출력한다.

```
276  void Reshape(int width, int height) //reshape함수의 콜백 함수
277  {
278      w = width;      //종횡비 값을 재설정
279      h = height;
280  }
```

윈도우의 변하는 크기를 w, h 전역 변수에 가져온다. 이 변수는 원근 투영 함수의 종횡비 값이다. 따라서 변한 윈도우 크기에 종횡비를 맞출 수 있게 한다.

```
282  void Keyboard(int key, int a, int s)    //special함수의 콜백 함수
283  {
284      switch (key) {  //화살표 키 입력 처리
285      case GLUT_KEY_LEFT:     //카메라 좌로 회전
286          spin_Display(key);
287          break;
288
289      case GLUT_KEY_RIGHT:    //카메라 우로 회전
290          spin_Display(key);
291          break;
```

키보드의 방향키 입력을 받아 처리하는 함수이다. 왼쪽 화살표키의 입력을 받게 되면 함수를 실행시켜 카메라가 왼쪽으로 회전하게 된다. 오른쪽 화살표키라면 오른쪽으로 회전한다.

```
293      case GLUT_KEY_DOWN:     //줌 아웃
294          fov += 0.5f;
295          glutPostRedisplay();
296          break;
297
298      case GLUT_KEY_UP:       //줌 인
299          fov -= 0.5f;
300          glutPostRedisplay();
301          break;
302      }
303  }
```

아래쪽 화살표키는 원근투영의 시야각 변수인 fov 가 증가하여 zoom in 이 된다. 반대로 위쪽 화살표키는 fov 가 감소하여 zoom out 이 된다.

```
305  void MyKeyboard(unsigned char KeyPressed, int X, int Y) {
306      switch (KeyPressed) {   //keyboard콜백 함수
307      case 27:        //esc 입력시 종료
308          exit(0);
309          break;
310      }
311  }
```

키보드의 입력을 받는 또 다른 함수이다. Esc 를 입력 받는다면 프로세스가 종료되게 된다. 일반적으로 윈도우의 x 버튼을 눌러 종료시키면 오브젝트의 소멸자가 작동하지 않는데, 이를 해결한다.

```
313     void Grid()     //발판을 그리는 함수
314     {
315         glPushMatrix();
316         glColor3f(1, 1, 1);
317         for (int i = -50; i < 50; i++){
318             glBegin(GL_LINES);
319             glNormal3f(0, 1, 0);
320             glVertex3f(i, 0, -50);
321             glVertex3f(i, 0, 50);
322             glEnd();
323         }
324 
325         for (int i = -50; i < 50; i++){
326             glBegin(GL_LINES);
327             glNormal3f(0, 1, 0);
328             glVertex3f(-50, 0, i);
329             glVertex3f(50, 0, i);
330             glEnd();
331         }
332 
333         glPopMatrix();
334     }
```

발판을 그리는 함수이다. 정점을 조금씩 이동시켜 50 개의 선을 가로, 세로로 그려 발판을 그리게 된다.

```
336     void PopupMenu(int index)   //팝업 처리 함수
337     {
338         switch (index) {
339         case 0:     //조명 on/off
340             light = !light;
341             break;
342         case 1:     //텍스쳐 on/off
343             texture = !texture;
344             break;
```

팝업 메뉴의 0 번을 누르면 light 변수가 토글 돼서 불이 켜지거나 꺼진다. 1 번 메뉴인 텍스쳐도 마찬가지이다.

```
345         case 2:     //meterial 번호 설정
346             Mat_N = 1;
347             break;
348         case 3:
349             Mat_N = 2;
350             break;
351         case 4:
352             Mat_N = 3;
353             break;
```

2, 3, 4 번은 재질을 3 개중 한 개로 설정하게 된다.

```
354         case 5:     //Smooth shading
355             shadmodel = 0;
356             break;
357         case 6:     //Flat shading
358             shadmodel = 1;
359             break;
360         case 7:     //오브젝트 라인 on/off
361             objMesh.line = !objMesh.line;
362             break;
363         }
364         glutPostRedisplay();
365     }
```

5 번과 6 번은 값을 변경하여 shading 모델을 선택하게 된다. 7 번은 오브젝트 테두리 선을 그릴지 정하는 토글 변수를 토글시킨다.

```
367     void spin_Display(int key) {        //각도 변환 함수
368         if(key == GLUT_KEY_RIGHT)
369             angle -= 10;        //0.05씩 각도 변환
370         else if(key == GLUT_KEY_LEFT)
371             angle += 10;        //0.05씩 각도 변환
372         if (angle > 360)            //한바퀴를 돌았다면
373             angle = angle - 360;    //다시 0도부터 회전
374         else if(angle < 0)
375             angle = angle + 360;
376         glutPostRedisplay();
377     }
```

rotatef 함수에서 각도의 인자인 angle 변수의 값을 바꾸는 함수이다. 매개변수로 가져온 key 가 오른쪽 키라면 각도를 감소시켜 왼쪽으로 돌게 되어 카메라가 오른쪽으로 이동하는 효과를 준다.

ObjMesh.h

```
13  class ObjMesh
14  {
15  public:
16      ObjMesh();
17      virtual ~ObjMesh();
18      void Render();
19      void SetColor(float r, float g, float b);
```

오브젝트의 클래스가 정의돼 있는 헤더파일이다. 생성자, 소멸자가 선언돼 있다. 그리고 오브젝트를 그리는 Render(), RGB 값을 설정하는 SetColor() 멤버 함수가 선언돼 있다.

```
21      float* m_vertices;      //정점 좌표 배열
22      float* m_texCoords;     //텍스쳐 좌표 배열
23      float* m_normals;       //normal 좌표 배열
24
25      int* m_Faces;           //면의 세 정점 좌표 번호 배열
26      int* m_TextureFace;     //면의 텍스쳐 좌표 번호 배열
27      int* m_norface;         //면의 noraml 좌표 번호 배열
```

멤버 변수들이다.

```
29      float m_Color[3];       //rgb값 배열
30
31      int m_numVertices;      //정점 개수
32      int m_numTexCoords;     //텍스쳐 개수
33      int m_numNormals;       //normal 개수
34      int m_numFaces;         //면 개수
35
36      bool line;      //오브젝트 선 토글 변수
37  };
```

기존 멤버변수에 line 이라는 변수를 추가했다. 오브젝트의 테두리 선을 그릴지 정하는 토글 변수이다.

meshFactory.cpp

```
1   #pragma once
2   #include <GL/glut.h>
3   #include "MeshFactory.h"
4   #include <windows.h>
5   #include "ObjMesh.h"
6
7   void CMeshFacotry::LoadObjModel(char* fileDir, ObjMesh* mesh)
8   {
9       FILE* fp = fopen(fileDir, "r");     //obj파일 read
10      if (fp == NULL) {
11          printf("%s file can not open", fileDir);
12          exit(1);
13      }
```

파일에서 오브젝트의 데이터를 읽어 저장하는 함수가 저장된 헤더파일이다. 매개변수로 가져온 파일이름을 찾아 읽기모드로 연다.

```
14      int numVertex = 0;      //정점 개수 변수
15      int numFaces = 0;       //면 개수
16      int numNormals = 0;     //정점 normal 좌표 개수
17      int numTexcoords = 0;   //텍스쳐 좌표 개수
18
19      float* vertices = NULL;
20      float* normal = NULL;
21      int* faces = NULL;
22      int* textureFace = NULL;
23      float* texCoords = NULL;
24      int* norface = NULL;
```

오브젝트 클래스의 값을 변경해주기 위한 변수를 선언했다. 함수가 동작하며 값이 변하게 된다.

```
26      char line[256];
27
28      while (!feof(fp)) {
29          fgets(line, 256, fp);
30          if (line[0] == 'v') {
31              if (line[1] == 't') {       //texture 개수 파악
32                  numTexcoords++;
33              }
34              else if (line[1] == 'n') {  //normal 개수 파악
35                  numNormals++;
36              }
37              else if (line[1] == ' ')    //정점 개수 파악
38              numVertex++;
39          }
40          else if (line[0] == 'f') {      //면 개수 파악
41              numFaces++;
42          }
43      }
44      rewind(fp);
```

파일에서 각 줄의 맨 앞에 저장된 문자 v, vt, vn, f 가 있는 문장의 개수를 파악한다.

```
51      vertices = (float*)malloc(sizeof(float) * 3 * numVertex);       //정점 좌표 배열
52      texCoords = (float*)malloc(sizeof(float) * 3 * numTexcoords);   //텍스쳐 좌표 배열
53      normal = (float*)malloc(sizeof(float) * 3 * numNormals);        //normal 좌표 배열
54      faces = (int*)malloc(sizeof(int) * 3 * numFaces);               //면의 정점 좌표 번호 배열
55      textureFace = (int*)malloc(sizeof(int) * 3 * numFaces);         //면의 텍스쳐 좌표 번호 배열
56      norface = (int*)malloc(sizeof(int) * 3 * numFaces);             //면의 normal 좌표 번호 배열
```

파악한 개수에 해당하는 데이터 배열을 동적할당 한다.

```
65      while (!feof(fp)) {
66          fgets(line, 256, fp);
67          if (line[0] == 'v') {
68              if (line[1] == 't') {       //texture 좌표 read
69                  fseek(fp, -(strlen(line) + 1), SEEK_CUR);   //읽은 문장의 앞으로 이동
70
71                  fscanf(fp, "%s %f %f %f", line, &x, &y, &z);
72                  texCoords[j++] = x; texCoords[j++] = y; texCoords[j++] = z; //좌표값 저장
73              }
```

맨 앞에 저장된 문자가 vt 인 경우, 텍스쳐 좌표를 읽어 저장한다. texCoords 는 텍스쳐의 좌표가 저장된 배열이 된다.

```
74              else if (line[1] == 'n') {  //normal 좌표 read
75                  fseek(fp, -(strlen(line) + 1), SEEK_CUR);   //읽은 문장의 앞으로 이동
76
77                  fscanf(fp, "%s %f %f %f", line, &x, &y, &z);
78                  normal[k++] = x;    normal[k++] = y;    normal[k++] = z;    //좌표값 저장
79              }
```

vn 인 경우, 이어서 나오는 normal 좌표를 읽어 normal 배열에 저장한다.

```
80              else if (line[1] == ' '){   //vertex 좌표 read
81                  fseek(fp, -(strlen(line) + 1), SEEK_CUR);   //읽은 문장의 앞으로 이동
82
83                  fscanf(fp, "%s %f %f %f", line, &x, &y, &z);
84                  vertices[l++] = x;  vertices[l++] = y;  vertices[l++] = z;  //좌표값 저장
85              }
86          }
```

V 인 경우, 이어 나오는 정점좌표를 읽어 vertices 배열에 저장한다. 정점 좌표의 배열이다.

```
87          else if (line[0] == 'f') {      //면 정보 read
88              fseek(fp, -(strlen(line) + 1), SEEK_CUR);       //한줄 앞으로
89
90              for (int a = 0; a < 3; a++) {   //삼각형의 정점 3개 좌표 read
91                  fscanf(fp, "%c %f%c%f%c%f", &temp, &x, &temp, &y, &temp, &z);
92                  faces[IdxFace++] = x-1; textureFace[IdxTexCoord++] = y-1;  norface[Idxnormal++] = z-1;
93              }
94          }
95      }
```

f 인 경우, 이어 나오는 면의 정보를 읽는다. 삼각형이기 때문에 3 개의 정점이 한문장에 나오므로 3 번 읽어 저장한다. 각 정점의 처음 값은 정점 배열의 번호, 두번째는 텍스쳐 배열의 번호, 세번째는 normal 배열의 번호이다.

```
97      //객체에 값 전달
98      //정점 개수, 좌표
99      mesh->m_numVertices = numVertex;
100     mesh->m_vertices = vertices;
101
102     //텍스쳐 개수, 좌표
103     mesh->m_numTexCoords = numTexcoords;
104     mesh->m_texCoords = texCoords;
105
106     //노말 개수, 좌표
107     mesh->m_numNormals = numNormals;
108     mesh->m_normals = normal;
109
110     //면의 정보(세 정점 번호, 텍스쳐 번호, normal 번호)
111     mesh->m_numFaces = numFaces;
112     mesh->m_Faces = faces;
113     mesh->m_TextureFace = textureFace;
114     mesh->m_norface = norface;
```

알아낸 오브젝트의 정보를 오브젝트 객체에 값으로 넘겨준다. 이렇게 오브젝트 객체에 obj 파일의 데이터를 읽는 것이다.

ObjMesh.cpp

```
14  ObjMesh::ObjMesh()  //생성자
15  {
16      m_vertices = NULL;
17      m_texCoords = NULL;
18      m_normals = NULL;
19
20      m_Faces = NULL;
21      m_TextureFace = NULL;
22
23      m_numTexCoords = m_numVertices = m_numFaces = m_numNormals = 0;
24
25      m_Color[0] = 1;
26      m_Color[1] = 1;
27      m_Color[2] = 1;
28
29      line = true;
30  }
```

최종적으로 오브젝트의 메쉬를 그리는 함수이다. 생성자가 선언돼 있다. 각 멤버 변수들의 값을 초기화 시켜준다. RGB 의 기본 값은 1, 1, 1 로 흰색으로 초기화가 된다.

```
32  ObjMesh::~ObjMesh() //소멸자
33  {
34          if(m_vertices != NULL)
35              free(m_vertices);
36          if(m_normals != NULL)
37              free(m_normals);
38          if(m_Faces != NULL)
39              free(m_Faces);
40          if(m_texCoords != NULL)
41              free(m_texCoords);
42          if(m_TextureFace != NULL)
43              free(m_TextureFace);
44  }
```

소멸자이다.

```
46  void ObjMesh::Render()  //오브젝트 랜더링 함수
47  {
48      glPushMatrix();
49
50      glColor3fv(m_Color);
51
52      //면 그리기
53      int j = 0;
54      for (int i = 0; i < m_numFaces*3; i += 3)
55      {
56          //텍스쳐 번호 배열(m_TextureFace)로 해당 면의 텍스쳐 좌표값 가져옴
57          float p1[2] = { m_texCoords[m_TextureFace[i + 0] * 3 + 0], m_texCoords[m_TextureFace[i + 0] * 3 + 1] };
58          float p2[2] = { m_texCoords[m_TextureFace[i + 1] * 3 + 0], m_texCoords[m_TextureFace[i + 1] * 3 + 1] };
59          float p3[2] = { m_texCoords[m_TextureFace[i + 2] * 3 + 0], m_texCoords[m_TextureFace[i + 2] * 3 + 1] };
```

가장 중요한 오브젝트를 렌더링하는 함수이다. 처음 초기화한 RGB 값으로 오브젝트를 그린다.

텍스쳐의 번호가 저장된 m_TextureFace 를 텍스쳐의 좌표값인 m_texCoords 의 인덱스로 넣어 해당 번호의 좌표 값을 구한다. 삼각형이므로 총 3 개를 꺼내온다.

```
61          //normal 번호 배열(m_norface)로 해당 면의 normal 좌표값 가져옴, 3으로 나누어 중심의 법선벡터 계산
62          float x = (m_normals[m_norface[i + 0] * 3 + 0] + m_normals[m_norface[i + 1] * 3 + 0] + m_normals[m_norface[i + 2] * 3 + 0]) / 3;
63          float y = (m_normals[m_norface[i + 0] * 3 + 1] + m_normals[m_norface[i + 1] * 3 + 1] + m_normals[m_norface[i + 2] * 3 + 1]) / 3;
64          float z = (m_normals[m_norface[i + 0] * 3 + 2] + m_normals[m_norface[i + 1] * 3 + 2] + m_normals[m_norface[i + 2] * 3 + 2]) / 3;
```

한 면에 해당하는 각 정점의 번호를 이용하여 각 정점의 법선 벡터를 구한 후, 3 으로 나누어 면에 해당하는 법선 벡터를 구해준다.

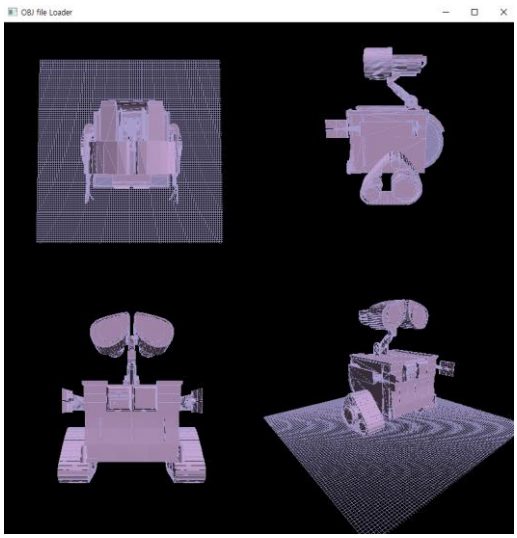
```
66          //텍스쳐, normal, 세 점으로 구성된 면을 매핑시켜 그림
67
68          glBegin(GL_TRIANGLES);
69          glNormal3f(x, y, z);
70          glTexCoord2fv(p1);
71          glVertex3f(m_vertices[m_Faces[i + 0] * 3 + 0], m_vertices[m_Faces[i + 0] * 3 + 1], m_vertices[m_Faces[i + 0] * 3 + 2]);
72
73          glTexCoord2fv(p2);
74          glVertex3f(m_vertices[m_Faces[i + 1] * 3 + 0], m_vertices[m_Faces[i + 1] * 3 + 1], m_vertices[m_Faces[i + 1] * 3 + 2]);
75
76          glTexCoord2fv(p3);
77          glVertex3f(m_vertices[m_Faces[i + 2] * 3 + 0], m_vertices[m_Faces[i + 2] * 3 + 1], m_vertices[m_Faces[i + 2] * 3 + 2]);
78          glEnd();
79      }
```

각 번호로 매핑된 정점과 텍스쳐, 법선 벡터로 한 면을 그린다.

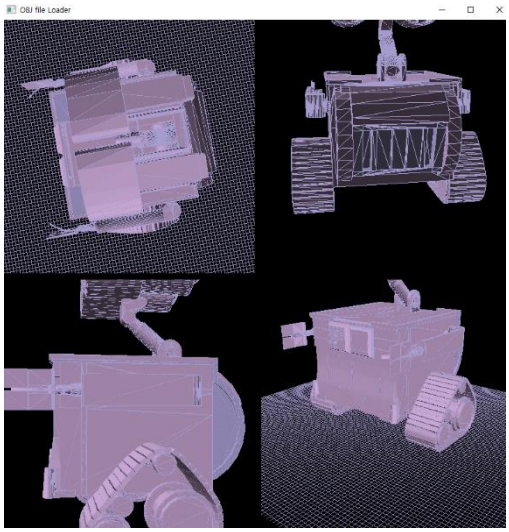
```
81      //외곽선 그리기
82      if (line) {
83          glScalef(1.0001, 1.0001, 1.0001);
84          glColor3d(0, 0, 0);
85          for (int i = 0; i < m_numFaces * 3; i += 3)
86          {
87              glBegin(GL_LINES);
88              glVertex3f(m_vertices[m_Faces[i + 0] * 3 + 0], m_vertices[m_Faces[i + 0] * 3 + 1], m_vertices[m_Faces[i + 0] * 3 + 2]);
89              glVertex3f(m_vertices[m_Faces[i + 1] * 3 + 0], m_vertices[m_Faces[i + 1] * 3 + 1], m_vertices[m_Faces[i + 1] * 3 + 2]);
90
91              glVertex3f(m_vertices[m_Faces[i + 1] * 3 + 0], m_vertices[m_Faces[i + 1] * 3 + 1], m_vertices[m_Faces[i + 1] * 3 + 2]);
92              glVertex3f(m_vertices[m_Faces[i + 2] * 3 + 0], m_vertices[m_Faces[i + 2] * 3 + 1], m_vertices[m_Faces[i + 2] * 3 + 2]);
93
94              glVertex3f(m_vertices[m_Faces[i + 2] * 3 + 0], m_vertices[m_Faces[i + 2] * 3 + 1], m_vertices[m_Faces[i + 2] * 3 + 2]);
95              glVertex3f(m_vertices[m_Faces[i + 0] * 3 + 0], m_vertices[m_Faces[i + 0] * 3 + 1], m_vertices[m_Faces[i + 0] * 3 + 2]);
96              glEnd();
97          }
98      }
99
100     glPopMatrix();
101 }
```

외각선 토글 변수를 확인한 후 외각선을 그리는 부분이다. 각 정점의 번호로 연결되는 선 3 개로 삼각형의 선을 그린다.

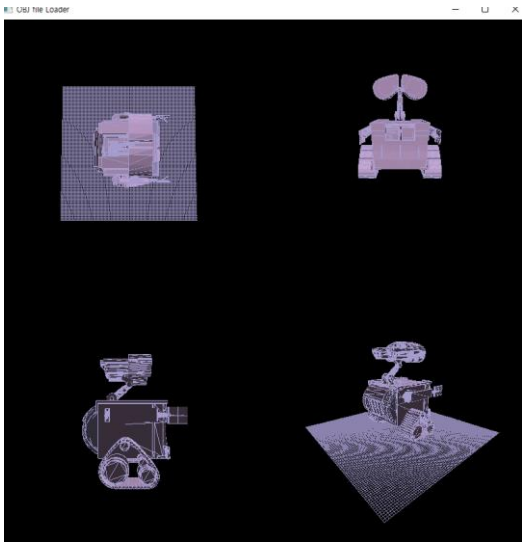
1-3 실행창



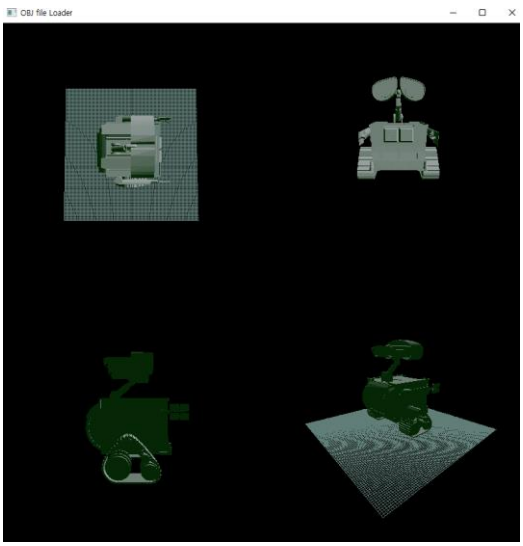
초기 모습



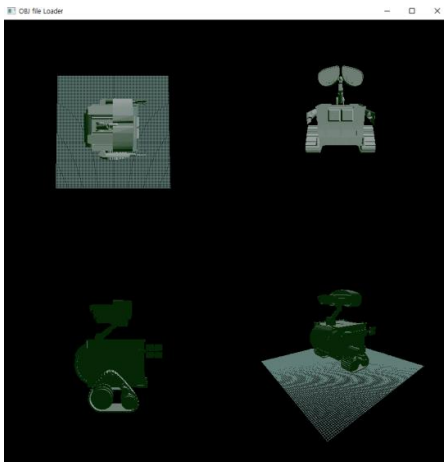
회전 + 줌 인



회전 + 줌 아웃



재질 1 적용



텍스쳐 off, SMOOTH 모드(만들어진 오브젝트가 너무 각져, FLAT 과 SMOOTH 의 차이가 드러나지가 않는다.)

1-4 느낀 점

오브젝트의 모델링이 조금 아쉬워서 Shading 모델의 적용이 눈에 띄지가 않는다. 이 부분이 좀 아쉬웠다. 그리고 obj 파일을 저장하는 방식에 따라 오브젝트를 읽어들이는 작업이 많이 달라져서 그 부분이 조금 힘들었다.