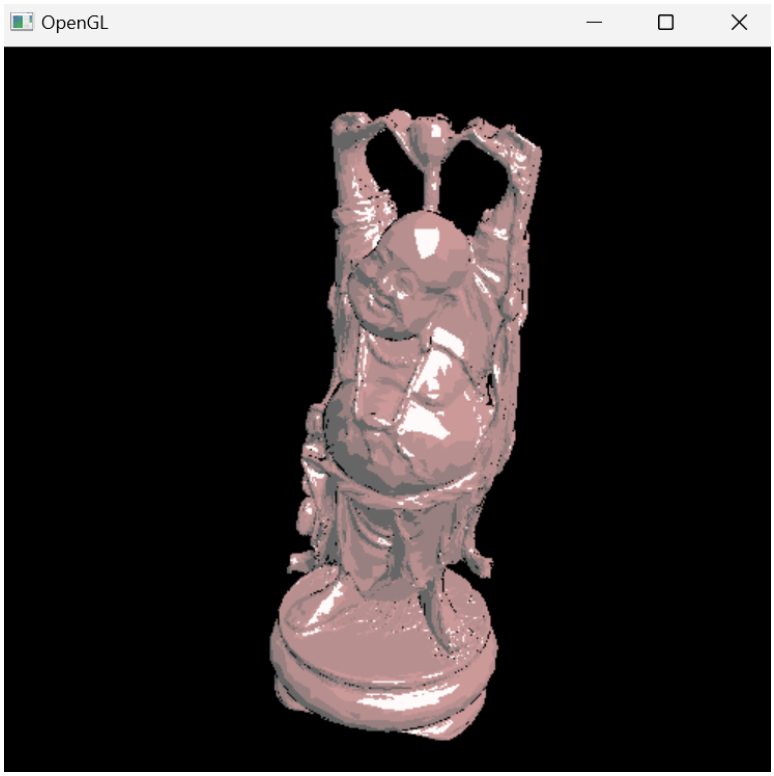
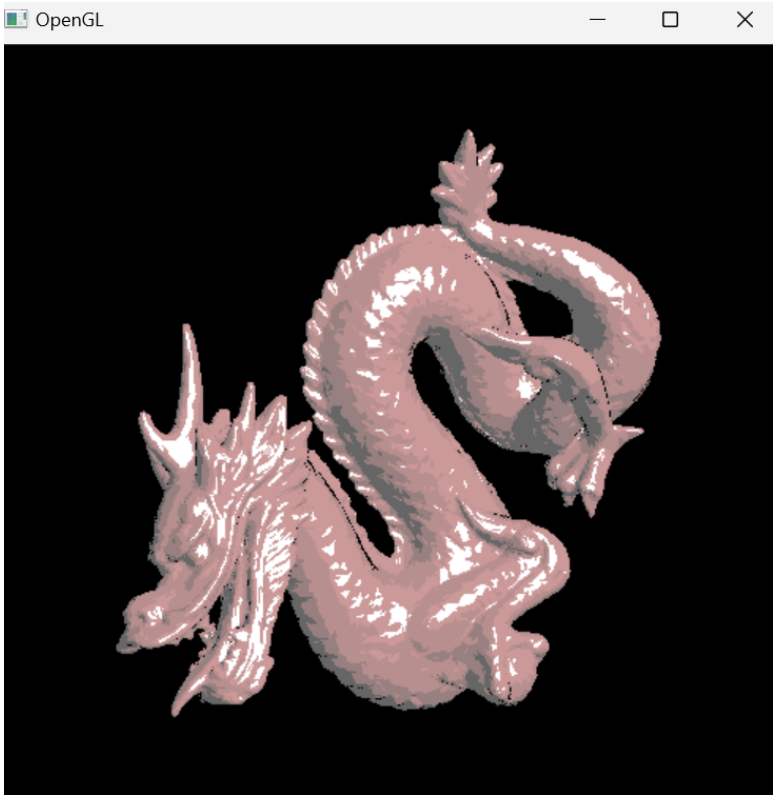
컴퓨터그래픽스 과제 4

소프트웨어학과 20003325 이태혁

실행화면





시행착오가 너무 많아서 처음 단계부터 어려움을 겪으며 구현했던 과정을 순서대로 작성하겠습니다.

```
char path[100];
FILE* s = NULL;
  while(true){
  printf("Input File Path:");
  scanf("%s", path);

  s = fopen(path, "rt");
  if (s == NULL)
    printf("File not Found!\n");
  else
    break;
```

파일경로를 path에 입력받도록 하고, 만약 파일을 찾지 못했다면 경고메세지를 출력 후 파일을 다시 입력받도록 하였다.

```
---------- 전역변수 ----------------
struct MyVertex
{
  vec4 points;
  vec4 colors;
  vec3 normals;
};


-------------- init 함수 내에서 동적할당 ------------------

  vertex = (vec4*)malloc(sizeof(vec4) * NumVertices);
  myVertices = (struct MyVertex*)malloc(sizeof(struct MyVertex) * planeNum * 3);
```

readV, readF 함수를 만들어서 obj파일을 공백을 기준으로 나누어서 myVertices 구조체의 vec4(points) 에 저장되도록 구현했다. readV, readF 함수의 구체적인 동작은 아래에 서술하였다.

myVertices 구조체 배열을 동적할당 하기 전에 미리 그 크기를 알아야 했다. 구조체 배열의 크기는 곧 점의 개수와 같았고, obj파일에서 ‘F’로 시작하는 문자열이 곧 삼각형 하나를 의미하므로, 줄의 개수를 세어서 거기에 3을 곱해주어 점의 개수를 구한 후 동적배열을 생성하였다.

myVertice 배열의 각 원소마다 normal과 color을 저장해주고, GPU로 넘겨주었다.

readV 함수 (vertex 읽기)

```
--------------- readV 함수 호출---------------

if (input[0] == 'v') {
      readV(input, vertex, vertIndex);
      vertIndex++;
    }

----------------- readV 함수 구현 -------------
void MyShape::readV(char input[], vec4 arr[], int index) {

  for (int i = 2, start = 2, mode = 1; i < strlen(input); i++) {

    char tmp[100] = { 0 };
    arr[index].w = 1.0f;
```

```
    if (input[i] == ' ' && mode == 1) {
      strncpy(tmp, &input[start], (i - start));
      arr[index].x = atof(tmp);
      start = i + 1;
      mode = 2;
      continue;
    }
    if (input[i] == ' ' && mode == 2) {
      strncpy(tmp, &input[start], (i - start));
      arr[index].y = atof(tmp);
      start = i + 1;
      mode = 3;
      continue;
    }
    if (input[i] == '\n' && mode == 3) {
      strncpy(tmp, &input[start], (i - start));
      arr[index].z = atof(tmp);
      return;
    }
  }
}
```

obj파일의 각 줄을 탐색하며 ‘v’로 시작하면 readV 함수를 호출한다.

readV 함수에서는 공백을 기준으로 공백 앞부분의 내용을 잘라내서 임시 문자열인 tmp에 넣고, 그 내용을 float형으로 변환해서 vec4타입 동적배열 vertex에 저장한다.

readF 함수 (face 읽기)

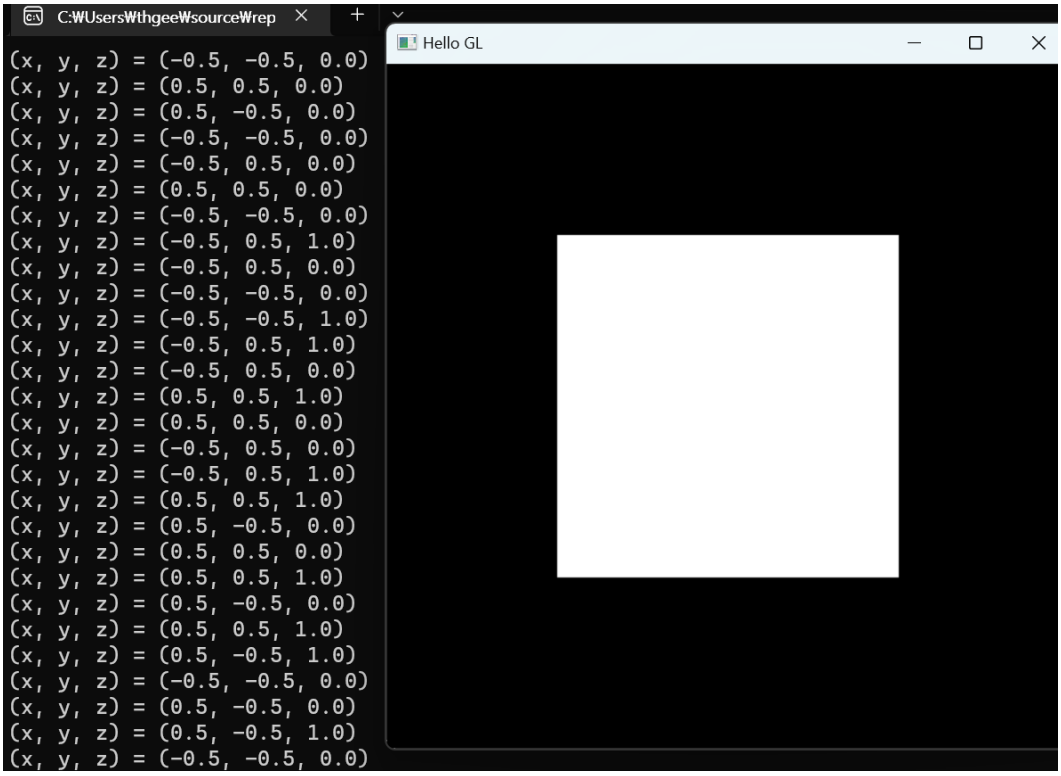
```
----------------- readF 함수 구현 -------------
void MyShape::readF(char input[], struct MyVertex vertices[], int* vertices_index) {
  for (int i = 2, start = 2, mode = 1; i < strlen(input); i++) {

    char tmp[100] = { 0 };
    if (input[i] == ' ' && mode == 1) {
      strncpy(tmp, &input[start], (i - start));
      vertices[*vertices_index].points.x = vertex[atoi(tmp) - 1].x;
      vertices[*vertices_index].points.y = vertex[atoi(tmp) - 1].y;
      vertices[*vertices_index].points.z = vertex[atoi(tmp) - 1].z;
      vertices[*vertices_index].points.w = 1.0f;

      (*vertices_index)++;
      start = i + 1;
      mode = 2;
      continue;
    }
    if (input[i] == ' ' && mode == 2) {
      ...

  }
```

readF함수의 경우 readV보다 조금 더 복잡하다.

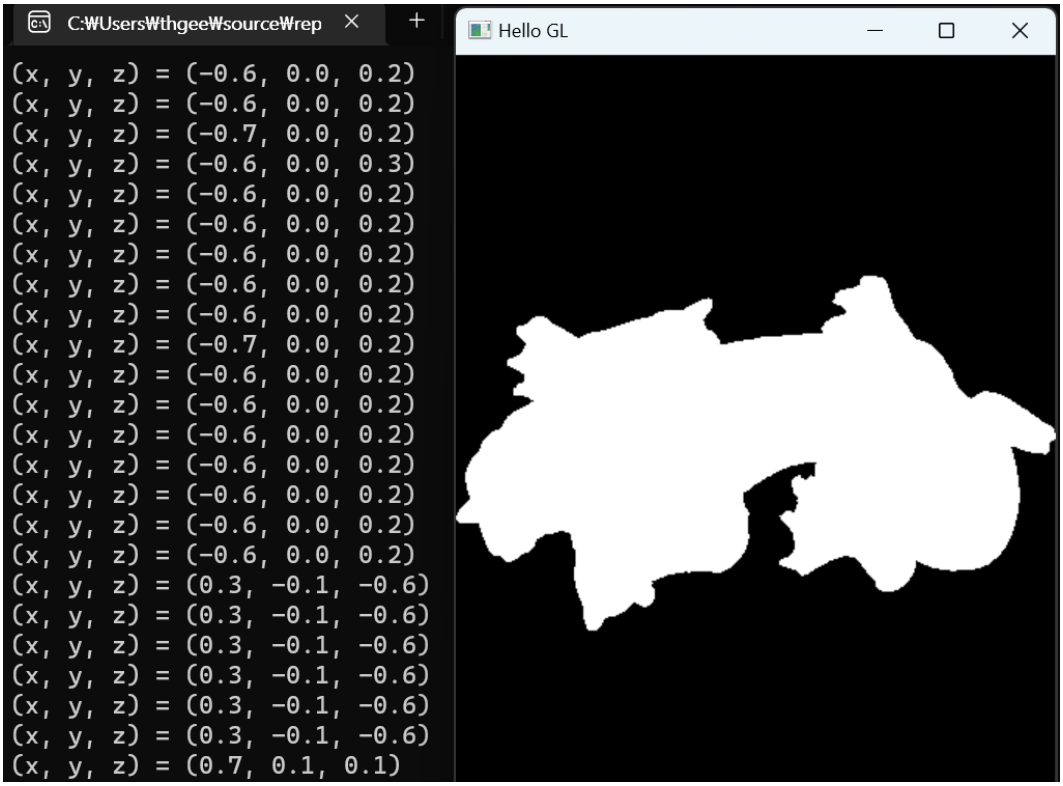
‘F’로 시작하는 문자열의 공백을 기준으로 분리해서 임시 문자열 tmp에 저장하는 것 까지는 readV와 같다. 그 이후에 readV에서 저장해둔 vertex배열에서 tmp에 저장된 문자열을 int 형으로 바꾼 후 그 인덱스에 해당하는 점을 구조체배열 vertices의 points.x에 저장해준다. points.y, z도 같은 방법으로 저장한다.이 과정을 마치고 나면 구조체배열 vertices의 points에는 (면의 개수 x 3) 만큼의 점이 저장된다.

초기 시행착오

obj파일을 vertex와 face로 나눠서 동적배열에 저장하는 것 까지는 성공했지만, 그 배열을 shader에 넘겨서 그림을 그리려고 하면 무슨 이유인지는 모르겠지만 그림이 그려지지 않았다. 왜 그려지지 않는지 하루종일 obj파일의 좌표도 수정해보고 scale을 조정해보는 등 이것저것 바꿔보았지만 결과창에는 검은 화면만 뜰 뿐이었다. 그러다가 혹시 동적배열이 문제인가 싶어서 시험삼아 제일 간단한 “cube.obj” 파일을 넣고 전역변수에 정적배열로 구현해보았는데 큐브가 정상적으로 그려지는 것을 확인했다.



그래서 하는김에 “dragon.obj” 파일도 정적배열로 그려보려고 시도했다. dragon파일의 vertex 개수는 50,000개, face개수는 100,000개 인 것을 이미 동적배열 구현할 때 알게 되어서 해당 숫자를 상수로 넣고 그대로 정적배열로 구현해보았다. color는 임시로 흰색으로 지정하여 그려보았다.



계속 코드를 뜯어보다가 동적배열로 구현했을 때 왜 그림이 그려지지 않는지 찾아내었다.

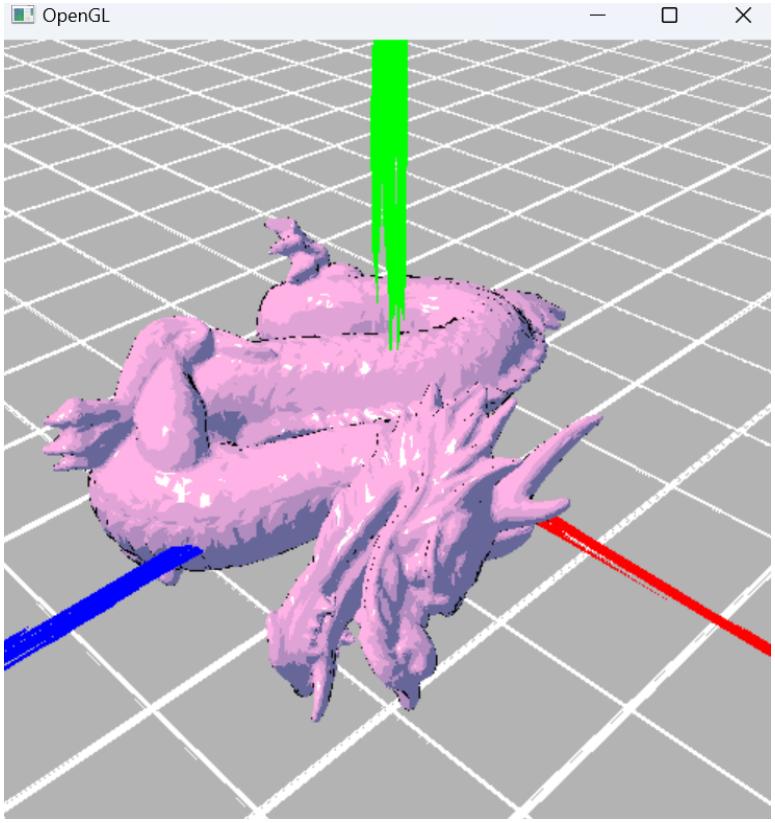
```
--- 잘못된 코드 ---
glBufferData(GL_ARRAY_BUFFER, sizeof(Vertices), Vertices, GL_STATIC_DRAW);
```

GPU로 데이터를 넘겨주는 과정에서 동적할당된 Vertices 자체를 넘겨주면 포인터를 넘겨주는 것이다. 그런데 넘겨주는 크기를 sizeof(Vertex) 로 해주면 포인터 하나의 크기인 4바이트를 넘겨주게 되어버린다.

```
--- 고친 코드 ---
glBufferData(GL_ARRAY_BUFFER, sizeof(vec4) * 점의개수, Vertices, GL_STATIC_DRAW);
```

따라서 위 코드처럼 모든 점의 크기를 합쳐서 직접 넘겨주어야 비로소 동적배열로도 그림이 그려진다.

이제 어떤 obj파일이 들어와도 동적배열에 저장할 수 있게 되었으니, diffuse와 specular 을 적용해보았다.



정상적으로 dragon의 형상이 그려지는 것을 확인하였다.

키보드 콜백

```
void keyboard(unsigned char c, int x, int y)
{
  switch (c)
  {
  case '3':
    printf("Increasing Specular Effect!\n");
    if (specNum >= 1.0f) break;
    specNum += 0.03f;
    break;
  case '4':
    printf("Decreasing Specular Effect!\n");
    if (specNum <= 0.02f) break;
    specNum -= 0.03f;
    break;
  case '5':
    printf("increasing shiness!\n");
    if (shiness >= 50) break;
    shiness += 2;
    break;
  case '6':
    printf("decreasing shiness!\n");
    if (shiness <= 2) break;
    shiness -= 2;
    break;

  case ' ':
    if (bPlay) printf("Stop!\n");
    else printf("Play!\n");
    bPlay = !bPlay;
    break;
  default:
    break;
  }
}
```

3~4번 키를 누르면 specular 의 밝기값이 조절되도록 하였고, 5~6번 키를 누르면 shiness 를 조절하여 shader에 넘겨주도록 구현하였다.

각종 예외처리

3번 키를 누르면 작동하는 Decreasing Specular Effect 작업에서 specular color 값이 vec4(0, 0, 0)까지 떨어지면 아래 사진처럼 정반사 부분이 까맣게 변해버린다. 따라서 0까지는 떨이지지 않도록 예외처리를 해주었다.

6번 키를 누르면 작동하는 decreasing shiness 작업에서 shiness가 0이 되면 아래 사진처럼 물체 전체가 하얗게 변해버리는 현상이 발생한다. 따라서 0이 되지 않도록 예외처리를 해주었다.

마우스 콜백

```
void myMouse(int btn, int states, int x, int y) {
  if (btn == GLUT_LEFT_BUTTON && states == GLUT_DOWN) {
    g_Time = 0.0f;
    CMT = ModelMat;
    bMoouseLeft = true;
    bMoouseMid = false;
    bMoouseRight = false;

  }
  if (btn == GLUT_MIDDLE_BUTTON && states == GLUT_DOWN) {
    g_Time = 0.0f;
    CMT = ModelMat;
    bMoouseLeft = false;
    bMoouseMid = true;
    bMoouseRight = false;
  }
  if (btn == GLUT_RIGHT_BUTTON && states == GLUT_DOWN) {
    g_Time = 0.0f;
    CMT = ModelMat;
    bMoouseLeft = false;
```

```
    bMoouseMid = false;
    bMoouseRight = true;
  }
}
```

마우스 버튼을 클릭하면 물체가 회전하도록 구현해야 하는데, 방향을 바꿀 때 마다 이전의 물체 방향이 유지되지 못한 채 물체의 방향이 확 바뀌면서 회전하는 현상이 발생했다.

mat4 CMT 행렬을 전역변수로 정의해주어서 마우스를 클릭할 때 마다 이전에 회전한 행렬을 저장해주고, 다음에 진행될 회전은 이전 방향까지의 회전행렬인 CTM을 먼저 적용해 준 후 진행되도록 하여서, 물체의 회전 방향이 바뀔 때 마다 현재의 방향을 유지한 채 자연스럽게 회전하도록 구현하였다.

과제 조건 구현현황

완성된 조건

1. Obj 파일 로드 성공

2. Viewing 관련:

   Perspective Projection을 통해 원근감이 느껴지도록 구현하였다.

   spacebar를 누르면 물체가 회전하며, 마우스 버튼을 이용해 회전 축을 자연스럽게 변경할 수 있도록 하였다.

3. Shading 관련:

   ‘3’과 ‘4’키를 통해 specular 효과를 높이거나 줄일 수 있으며, ‘5’와 ‘6’ 키를 통해 specular 효과의 shininess를 조절할 수 있도록 구현하였다.

미완성된 조건

1. 입력 받은 물체 자동으로 화면의 가운데에 위치하게 나오며, 화면에 알맞게 크기가 조절되어 있다.

2. Phong illumination을 이용하여 그리되, ‘1’ 키를 통해 vertex normal을 활용 (phong shading) 하거나, ‘2’키를 통해 surface normal을 활용(flat shading)하여 그릴 수 있다.