

OpenGL 코드리뷰

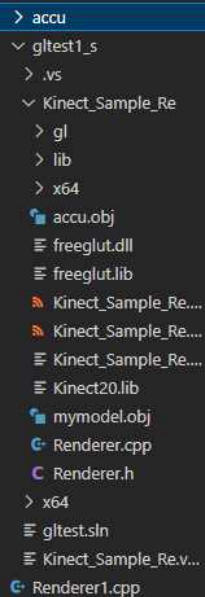
2017104034 최시원

OpenGL이란?

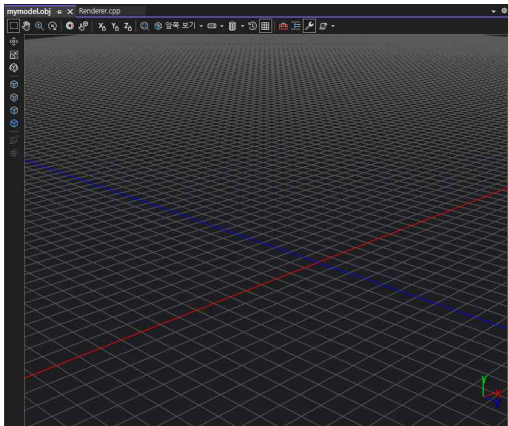
- 그래픽 및 이미지를 조작하는 데 사용할 수 있는 많은 기능을 제공하는 API
- 언어가 아니며 250여 개의 함수로 구성된 라이브러리

예제 폴더 구조

- visual studio project를 실행할 수 있는 gltest.sln 파일
- gl 라이브러리와 freeglut 라이브러리
- accu 폴더의 accu.obj, mymodel.obj를 gltest1_s/Kinect_Sample_Re에 옮긴 후 해당 폴더 내의 Renderer.cpp를 실행시킴으로써 진행
- 바깥의 Renderer1.cpp는 Renderer.cpp와 약간 다른 기능을 가진 cpp 코드

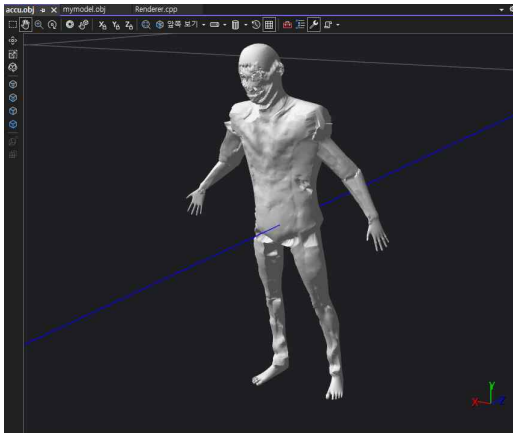


mymodel.obj



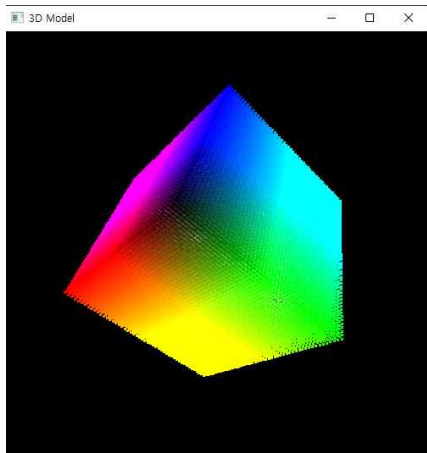
- x, y, z 축으로 구성된 3차원 공간에서의 2차원 격자 평면의 모습

accu.obj



- 2차원 격자 평면 위에 3차원 사람 모델링의 모습

최초 실행 결과



- Renderer.cpp 실행 결과
- 해당 정육면체 모델을 돌리고 확대하는 등의 조작을 할 수 있는 window가 나타남
- 좌클릭은 모델 회전, 스크롤은 모델 확대/축소, 우클릭은 모델 시점 이동

draw_center

```
void draw_center(void)
{
    glBegin(GL_LINES);
    glColor3f(1.0f, 0.0f, 0.0f); /* R */
    glVertex3f(0.0f, 0.0f, 0.0f);
    glVertex3f(0.2f, 0.0f, 0.0f);
    glEnd();
    glRasterPos3f(0.2f, 0.0f, 0.0f);
    glutBitmapCharacter(GLUT_BITMAP_9_BY_15, 'x');

    glBegin(GL_LINES);
    glColor3f(0.0f, 1.0f, 0.0f); /* G */
    glVertex3f(0.0f, 0.2f, 0.0f);
    glVertex3f(0.0f, 0.0f, 0.0f);
    glEnd();
    glRasterPos3f(0.0f, 0.2f, 0.0f);
    glutBitmapCharacter(GLUT_BITMAP_9_BY_15, 'y');

    glBegin(GL_LINES);
    glColor3f(0.0f, 0.0f, 1.0f); /* B */
    glVertex3f(0.0f, 0.0f, -0.2f);
    glVertex3f(0.0f, 0.0f, 0.0f);
    glEnd();
    glRasterPos3f(0.0f, 0.0f, -0.2f);
    glutBitmapCharacter(GLUT_BITMAP_9_BY_15, 'z');
}
```

- glBegin : 도형 그리기 시작
- glEnd : 도형 그리기 끝. 도형을 다 정의함
- glColor3f : 세가지 색상 요소의 강도를 실수로 지정함
- glVertex3f : glBegin / glEnd 쌍 내에서 점, 선 및 다각형 꼭짓점을 지정하는 데 사용됨. 현재 색, 법선 및 질감 좌표는 glVertex가 호출될 때 꼭짓점과 연결됨.
- glRasterPos3f : 창 좌표에서 3차원 위치를 유지함. '래스터 위치(픽셀을 기반으로 함)'
- glutBitmapCharacter : 비트맵 폰트. 2d 폰트를 의미한다. 3d상에서 이를 표현하나, 굽기가 없고 회전 및 크기변환을 할 수 없다. 이 함수들이 x, y, z를 그린다.
- .obj 파일에서 확인한 것과 같은 평면을 그리는 기능으로 생각됨

idle

```
void idle() {  
    static GLuint previousClock = glutGet(GLUT_ELAPSED_TIME);  
    static GLuint currentClock = glutGet(GLUT_ELAPSED_TIME);  
    static GLfloat deltaT;  
  
    currentClock = glutGet(GLUT_ELAPSED_TIME);  
    deltaT = currentClock - previousClock;  
    if (deltaT < 1000.0 / 20.0) { return; }  
    else { previousClock = currentClock; }  
  
    //char buff[256];  
    //sprintf_s(buff, "Frame Rate = %f", 1000.0 / deltaT);  
    //frameRate = buff;  
  
    glutPostRedisplay();  
}
```

- callback으로 등록되는 함수중 하나
- glutGet : 정수로 표시되는 단순한 GLUT state를 표시한다.
- GLUT_ELAPSED_TIME : 위 함수의 인자. glutinit이 호출된 순간부터의 millisecond 단위의 숫자를 반환한다. (또는 glutGet(GLUT_ELAPSED_TIME)으로 실행될때)
- glutPostRedisplay : 현재의 윈도우를 재생하도록 요구함. 예로 이전 윈도우에 그려진 내용이 프로그램에 의해 변경된다면, 이를 새로 가져오기 위해 쓰임
- idle 상태일 때 일정 시각마다 윈도우를 새로 가져오는 기능

close

```
void close()  
{  
    glDeleteTextures(1, &dispBindIndex);  
    glutLeaveMainLoop();  
    CloseHandle(hMutex);  
}
```

- callback 함수중 하나
- glDeleteTextures : 명명된 텍스처를 삭제함.
- glutLeaveMainLoop : freeglut 라이브러리의 함수. 이 함수를 통해 종료 후 후처리 가능함
- CloseHandle(n) : 닫고자 하는 열린 쓰레드 핸들인 n을 닫는다.
- 윈도우 종료 기능

add_quats

```
void add_quats(float q1[4], float q2[4], float dest[4])
{
    static int count = 0;
    float t1[4], t2[4], t3[4];
    float tf[4];

    vcopy(q1, t1);
    vscale(t1, q2[3]);

    vcopy(q2, t2);
    vscale(t2, q1[3]);

    vcross(q2, q1, t3);
    vadd(t1, t2, tf);
    vadd(t3, tf, tf);
    tf[3] = q1[3] * q2[3] - vdot(q1, q2);

    dest[0] = tf[0];
    dest[1] = tf[1];
    dest[2] = tf[2];
    dest[3] = tf[3];

    if (++count > RENORMCOUNT) {
        count = 0;
        normalize_quat(dest);
    }
}
```

- vcopy, vscale 등 코드 내부에 있는 v~ 함수들은
- 인자로 받은 값에 맞는 동작을 한다. 인자는 v[0], v[1], v[2]들이 연산되는걸로 보아 3차원 벡터로 생각된다
- q1을 t1에 복사 후, q2[3] 만큼 t1을 scale 한다
- q2를 t2에 복사 후, q1[3] 만큼 t2를 scale 한다
- t1과 t2를 외적 연산 vcross 한 후, 이를 t3에 복사한다.
- $tf = t1 + t2$, $tf = tf + t3$
- $tf[3] = q1[3] * q2[3] - (q1 \text{과 } q2 \text{의 내적})$
- dest 에 tf[1, 2, 3, 4]를 저장하고 정규화한다

reshape

```
void reshape(int width, int height)
{
    glViewport(0, 0, width, height);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(58, (double)width / height, 0.1, 100);
    glMatrixMode(GL_MODELVIEW);
}
```

- callback으로 등록되는 함수중 하나
- glViewport : viewport를 설정한다. (0, 0, width, height); 이면 정방향이다.
- glMatrixMode(GL_MODELVIEW) : 특정 좌표 0,0,0에 도형을 그린다면 GL_MODELVIEW 매트릭스를 곱해서 실제적 위치 지정
- glMatrixMode(GL_PROJECTION) : 위의 GL_MODELVIEW에서 그려진 도형에 대한 실제 위치라고 하면 GL_PROJECTION에 있는 매트릭스를 곱해 최종적으로 어떻게 화면에 뿌릴까에 대한 부분
- glLoadIdentity(); : 해당 행렬을 항등행렬로 만들
- gluPerspective(58, (double)width / height, 0.1, 100);
- : 원근 투영을 나타내는 함수. 시야각 fovy:58과 종횡비 aspect:width/height를 사용해 원근 투영 행렬을 만든다. 0.1, 100은 각각 near, far으로 near, far 클립 평면의 거리이다.
- window size가 변경되었을 경우 콜백함수로 호출되어 동작

```

void motion(int x, int y)
{
    GLfloat spin_quat[4];
    float gain;
    gain = 2.0; /* trackball gain */

    if (drag_state == GLUT_DOWN)
    {
        if (button_state == GLUT_LEFT_BUTTON)
        {
            trackball(spin_quat,
                (gain + rot_x - 500) / 500,
                (500 - gain + rot_y) / 500,
                (gain + x - 500) / 500,
                (500 - gain + y) / 500);
            add_quats(spin_quat, quat, quat);
        }
        else if (button_state == GLUT_RIGHT_BUTTON)
        {
            t[0] -= (((float)trans_x - x) / 500);
            t[1] += (((float)trans_y - y) / 500);
        }
        else if (button_state == GLUT_MIDDLE_BUTTON)
            t[2] -= (((float)trans_z - y) / 500 + 4);
        else if (button_state == 3 || button_state == 4) // scroll
        {
        }

        //glutPostRedisplay();
    }

    rot_x = x;
    rot_y = y;

    trans_x = x;
    trans_y = y;
    trans_z = y;
}

```

motion

- trackball : 3차원 공간 탐색을 위한 트랙볼 설정
- drag_state에 태그를 지정함으로써 특정 동작에 맞추어 기능을 구현할 수 있음
- GLUT_DOWN 마우스 누르고 있음
- GLUT_LEFT_BUTTON 왼쪽 클릭
- GLUT_RIGHT_BUTTON 오른쪽 클릭
- GLUT_MIDDLE_BUTTON 스크롤 조작
- 공통적으로 마우스 입력을 유지할때를 기준으로, 왼쪽은 모델 rotate, 오른쪽은 모델 이동, 스크롤은 모델 확대/축소로 생각됨

```

void mouse(int button, int state, int x, int y)
{
    if (state == GLUT_DOWN)
    {
        if (button == GLUT_LEFT_BUTTON)
        {
            rot_x = x;
            rot_y = y;

            //t[0] = t[0] + 1;
        }
        else if (button == GLUT_RIGHT_BUTTON)
        {
            trans_x = x;
            trans_y = y;
        }
        else if (button == GLUT_MIDDLE_BUTTON)
        {
            trans_z = y;
        }
    }
    else if (button == 3 || button == 4)
    {
        const float sign = (static_cast<float>(button)-3.5f) * 2.0f;
        t[2] -= sign * 500 * 0.00015f;
    }
}

drag_state = state;
button_state = button;
}

void vzero(float*);
void vset(float*, float, float, float);
void vsub(const float*, const float*, float*);
void vcopy(const float*, float*);
void vcross(const float *v1, const float *v2, float *cross);
float vlength(const float *v);
void vscale(float *v, float div);
void vnormal(float *v);
float vdot(const float *v1, const float *v2);
void vadd(const float *src1, const float *src2, float *dst);

```

mouse

- 역시 마우스 관련된 동작을 처리

v~ 함수들

- 각 이름에 맞는 기능을 수행한다.

initializeWindow -1

```
void InitializeWindow(int argc, char* argv[])
{
    // initialize glut settings
    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_ALPHA | GLUT_DEPTH);
    glutInitWindowSize(1000 / 2, 1000 / 2);

    glutInitWindowPosition(0, 0);

    dispWindowIndex = glutCreateWindow("3D Model");

    trackball(&quat, 90.0, 0.0, 0.0, 0.0);

    glutIdleFunc(idle);
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutSpecialFunc(special);
    glutMotionFunc(motion);
    glutMouseFunc(mouse);
    glutCloseFunc(close);

    glutSetOption(GLUT_ACTION_ON_WINDOW_CLOSE, GLUT_ACTION_GLUTMAINLOOP_RETURNS);

    // bind textures
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    glEnable(GL_DEPTH_TEST);
    reshape(1000, 1000);

    /*glBindTextures(1, &dispBindIndex);
    glBindTexture(GL_TEXTURE_2D, dispBindIndex);
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR); */
}
```

- glutInit : glut 라이브러리를 초기화하고 기반플랫폼의 윈도우 시스템과 연결함
- glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_ALPHA | GLUT_DEPTH);
- : 디스플레이 표면의 주요 특징들을 결정함
- 트루컬러모드, 더블버퍼 사용, 색상에 알파값 사용, 깊이버퍼를 사용한 모습
- glutInitWindowSize : 3d모델을 표현할 윈도우의 폭width과 높이 height를 결정함
- glutInitWindowPosition : 윈도우의 x,y 시작좌표를 결정함
- glutCreateWindow : 윈도우는 상단에 제목을 가지는데, 이를 문자열 인수로 지정함. 윈도우를 지칭하는 유일한 id가 return됨
- glutIdleFunc(idle); app의 휴면idle 시간에 호출될 함수를 직접 만든 idle로 지정

initializeWindow - 2

```
void InitializeWindow(int argc, char* argv[])
{
    // initialize glut settings
    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_ALPHA | GLUT_DEPTH);
    glutInitWindowSize(1000 / 2, 1000 / 2);

    glutInitWindowPosition(0, 0);

    dispWindowIndex = glutCreateWindow("3D Model");

    trackball(quat, 90.0, 0.0, 0.0, 0.0);

    glutIdleFunc(idle);
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutSpecialFunc(special);
    glutMotionFunc(motion);
    glutMouseFunc(mouse);
    glutCloseFunc(close);

    glutSetOption(GLUT_ACTION_ON_WINDOW_CLOSE, GLUT_ACTION_GLUTMAINLOOP_RETURNS);

    // bind textures
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    glEnable(GL_DEPTH_TEST);
    reshape(1000, 1000);

    //glBindTextures(1, &dispBindIndex);
    glBindTexture(GL_TEXTURE_2D, dispBindIndex);
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR); +/
}
```

- glutReshapeFunc(reshape); 사용자가 창 위치 변경, 크기 변경 등의 행위 시 동적으로 반응하는 콜백함수 reshape()를 지정해 리세이프 이벤트를 등록함.
- 리세이프 이벤트 발생시 glut는 변경된 윈도우 폭과 높이를 콜백함수에게 넘겨줌
- glutSpecialFunc(special); : F1~F12 / 화살표 / PgUp / PgDn / Home / End / Insert 에 대한 입력을 받음
- glutMotionFunc(motion); 마우스 콜백. 마우스 버튼을 누를때 또는 움직일때 발생한다.
- glutMouseFunc(mouse); 마우스 콜백. 마우스 클릭 처리를 위한 콜백 함수
- glutCloseFunc(close); : 윈도우 종료에 대한 콜백으로 생각됨
- glutSetOption : glutLeaveMainLoop()를 통한 종료 후 후처리 기능을 사용하기 위해 메인루프 실행 전에 하는 glut 설정
- glClearColor : 설정한 색상으로 화면을 지움
- glEnable(GL_DEPTH_TEST); 주어진 옵션을 활성화함. 여기서는 깊이 테스트를 설정

display

```
void display()
{
    // clear buffers
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    // set matrix
    glMatrixMode(GL_PROJECTION);
    //glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    //glPushMatrix();
    //glOrtho(-2, 2, -2, 2, -1000, 1000);
    gluPerspective(60, 1, 1, 2000);
    glTranslatef(t[0], t[1], t[2] - 1.0f);
    glScalef(1, 1, 1);

    //glTranslatef(1,0,0);
    glRotatef(0,1,1,1);

    GLfloat m[4][4], m1[4][4];
    build_rotmatrix(m, quat);
```

- `glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);` 버퍼들을 미리 설정된 값으로 지운다. 컬러 쓰기과 깊이 버퍼를 지우도록 설정됨
- `glLoadIdentity();` 공간상에 두번째 물체를 그릴 때 첫번째 물체의 변환행렬이 남아있는 것을 초기화하기 위함
- `gluPerspective` 원근투영 나타냄
- `glTranslatef` 정점의 이동을 해줌.
- `glScalef` 정점간 거리, 즉 벡터의 길이를 늘리고 줄여줌.
- `glRotatef` 지정된 angle만큼 한 축을 기준으로 회전함
- `glMultMatrixf` 현재 행렬에 임의의 행렬을 곱함
- `glPointSize` 래스터화 된 점의 지름을 정함
- `glutSwapBuffers` 백버퍼와 전면버퍼를 통째로 교체한다. 백 버퍼에 미리 준비해둔 그림이 바로 나타나게 된다 이 버퍼 교체작업 자체가 출력이다

main

```
int main(int argc, char* argv[])
{
    vertex = new Vertex[READ_SIZE];
    vertex_color = new Vertex[READ_SIZE];

    InitializeWindow(argc, argv);

    display();

    glutMainLoop();
    delete[] vertex;
    delete[] vertex_color;
    return 0;
}
```

- main 함수이다. InitializeWindow, display 함수를 실행시키고 glutMainLoop로 들어간다
- glutMainLoop : 이벤트 처리 루프이다. 이 루프 안에서 이벤트가 발생하면 일치하는 콜백 함수가 등록된 경우, 등록된 함수가 호출된다

Renderer1.cpp



accu



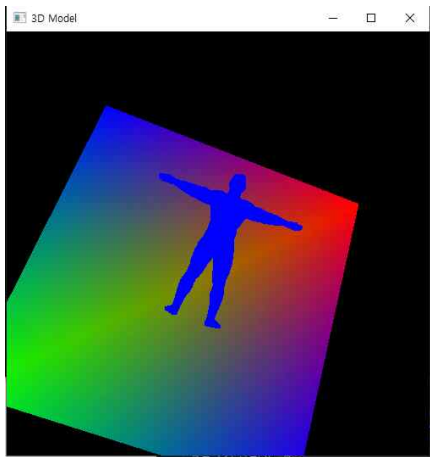
gltest1_s



Renderer1.cpp

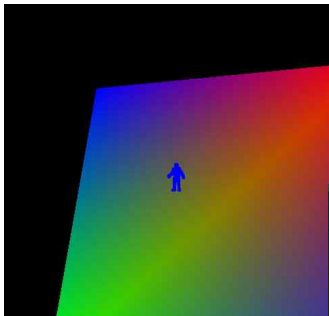
- 기존 gltest1_s 안에 있던 Renderer.cpp 코드와 구분하기 위하여 이름을 바꾸었다
- 실제 프로젝트를 실행할 때는 이름을 변경하지 않고, 해당 cpp파일을 gltest1_s 내부의 Renderer.cpp 파일에 그대로 덮어씌워 진행하였다

Renderer1.cpp 실행결과 - mymodel.obj

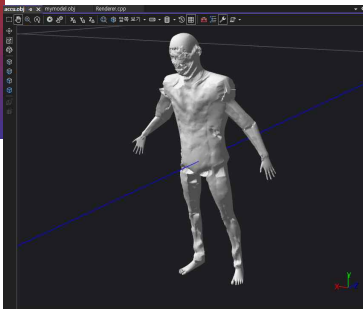


- main 함수에서 mymodel.obj 파일을 읽어들이는 경우이다
- 2d평면과 그 위에 떠 있는 팔을 벌린 사람의 모델링이 표현된다

Renderer1.cpp 실행결과 - accu.obj



- main 함수에서 accu.obj 파일을 읽어들이는 경우이다
- 2d평면과 좀 작게 표현된 사람의 modeling이 표현된다. 이 모델링은 최초 accu.obj 파일을 그냥 열었을 때 나타난 모델링으로 생각된다



Renderer1.cpp의 변경점 - main

```
int main(int argc, char* argv[])
{
    vertex = new Vertex[100000];
    vertex_color = new Vertex[100000];
    //myMesh = new Meshmodel[100000];

    FILE* fp;
    fp = fopen("accu.obj", "r");
    int count = 0;
    int num = 0;
    char ch;
    float x, y, z;

    for (register int j = 0; j < 100000; j = j + 1)
    {
        count = fscanf(fp, "%c %f %f %f", &ch, &x, &y, &z);
        if (count == 4 && ch == 'v')
        {
            vertex[j].X = x / scale;
            vertex[j].Y = y / scale;
            vertex[j].Z = z / scale;
        }
    }

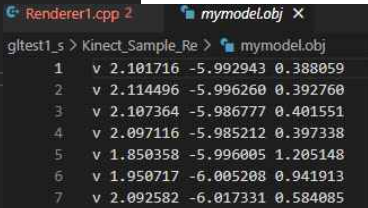
    fclose(fp);

    InitializeWindow(argc, argv);

    display();

    glutMainLoop();
    delete[] vertex;
    delete[] vertex_color;
    return 0;
}
```

- .obj 파일을 읽어들이는 부분이 추가되었다. fopen 함수로 같은 폴더 내에 있는 .obj 파일 이름을 지정하고 연다.
- 이후 vertex 변수에 해당 파일의 정보를 옮긴다.
- 3d 형식으로 표현되지 않은 .obj 파일의 구성이다
- 맨 앞 character 이후로 3차원 위치정보가 표현되어 있다.



```
Renderer1.cpp 2 mymodel.obj X
gltest1_s > Kinect_Sample_Re > mymodel.obj
1 v 2.101716 -5.992943 0.388059
2 v 2.114496 -5.996260 0.392760
3 v 2.107364 -5.986777 0.401551
4 v 2.097116 -5.985212 0.397338
5 v 1.850358 -5.996005 1.205148
6 v 1.950717 -6.005208 0.941913
7 v 2.092582 -6.017331 0.584085
```

Renderer1.cpp의 변경점 - display

```
//set: gluPerspective(10, 1, 1, 200); // labtest 9
glPointSize(3);
glBegin(GL_POINTS);
for (register int j = 0; j < 100000; j=j+1)
{
    glColor3f(0, 0, 1);
    glVertex3f(vertex[j].X, vertex[j].Y, vertex[j].Z);
}
glEnd();

glPointSize(7);
glBegin(GL_TRIANGLES);

glColor3f(0, 0, 1);
glVertex3f(-1, -1, 1);
glColor3f(0, 1, 0);
glVertex3f(1, -1, 1);
glColor3f(1, 0, 0);
glVertex3f(-1, 1, 1);

glColor3f(0, 0, 1);
glVertex3f(1, 1, 1);
glColor3f(0, 1, 0);
glVertex3f(1, -1, 1);
glColor3f(1, 0, 0);
glVertex3f(-1, 1, 1);

glEnd();
glutSwapBuffers();
```

- display 함수가 다음과 같이 다르다.
- 자료의 윗부분은 읽어들이는 .obj 파일을 표현하는 부분으로 생각된다
- 밑부분은 2d평면을 표현한다.
- .obj 파일만을 열람하였을 때는 accu.obj 파일에서만 사람 모델링을 확인할 수 있었으나, mymodel을 지정하여 Renderer.cpp을 실행시킨 결과 accu.obj보다 더욱 큰 사람 모델링이 window에 표현된 것을 볼 수 있다.

Renderer1.cpp의 기타 실행 옵션

```
void display()
{
    // clear buffers
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    // set matrix
    glMatrixMode(GL_PROJECTION);
    //glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    //glPushMatrix();
    glOrtho(-2, 2, -2, 2, -2, 2); // labtest 1
    glOrtho(-2, 2, -2, 2, 0, 2); // labtest 1
    //glOrtho(0, 2, -2, 2, -2, 2); // labtest 1
    //gluPerspective(10, 1, 1, 20); // labtest 2
    gluPerspective(60, 1, 1, 20); // labtest 2

    //glFrustum(-1, 1, -1, 1, 1, 20); // labtest 2-1
    //glFrustum(-0.1, 0.1, -0.1, 0.1, 1, 20); // labtest 2-1
    glTranslatef(t[0], t[1], t[2] - 1.0f);

    //glTranslatef(t[0], t[1], t[2] - 3.0f); // labtest 3
    //glScalef(3, 3, 3); // labtest 3

    //glScalef(1, 2, 3); // labtest 4
    glScalef(1, 1, 1); // labtest 4
    //glTranslatef(1,0,0);

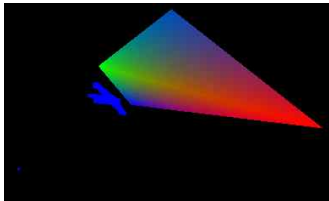
    //glRotatef(35.26, 1.0, 0.0, 0.0); // labtest 5
    //glRotatef(45.0, 0.0, 1.0, 0.0); // labtest 5

    //gluLookAt(0, 0, 3, 0, 0, 0, 0, 1, 0); // labtest 6
    //gluLookAt(0, 0, 3, 0, 0, 0, 0, -1, 0); // labtest 6
    //gluLookAt(0, 0, 3, 0, 0, 100, 0, 1, 0); // labtest 6

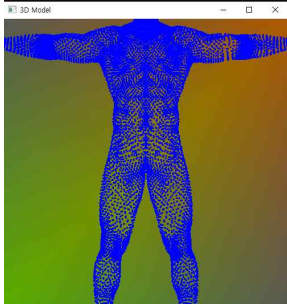
    GLfloat m[4][4], ml[4][4];
    build_rotmatrix(m, quat);
}
```

- Renderer1.cpp의 display 함수엔 자료와 같이 주석 처리된 labtest들이 있다.
- 이들을 하나씩 적용하고 실행해보며 결과를 확인해 보았다.
- 기존 .obj 파일은 mymodel.obj를 이용하였다. 모델링이 크고 확실하게 보이기 때문이다.

labtest 1, 2

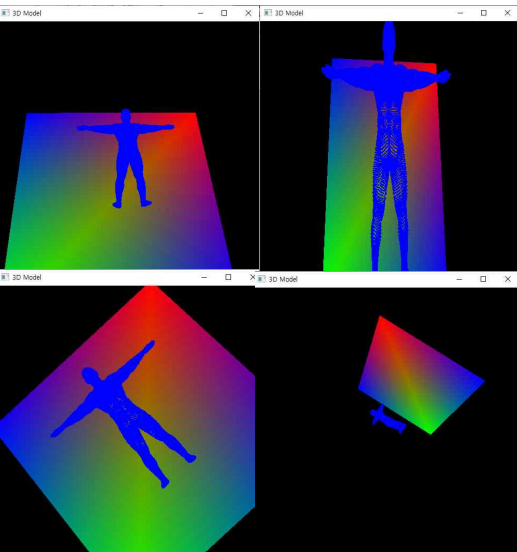


- labtest 1은 glOrtho 함수를 이용한다. viewport로 인해 왜곡된 것을 바로잡는데 사용한다



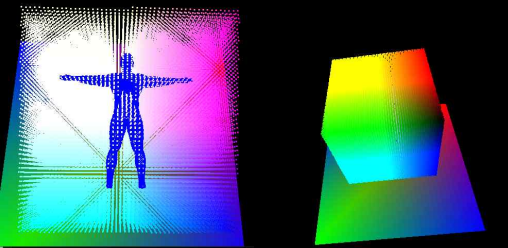
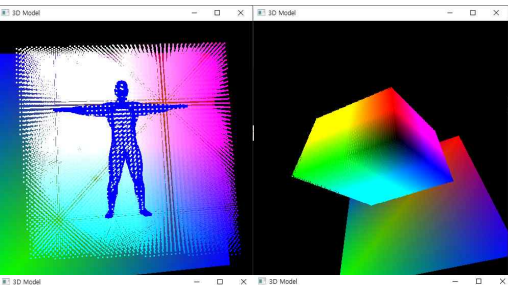
- labtest 2는 원근 투영을 나타내는 gluPerspective 함수를 이용한다. 원래 사용하던 시야각 60에서 10으로 변경하여 진행하니 매우 가깝게 표시된 모습이다

labtest3, 4, 5, 6



- labtest3에서는 정점 이동을 해주는 `glTranslatef`, 정점간 거리를 늘리고 줄이는 `glScalef` 함수가 쓰인다
- labtest4에서는 `glScalef`가 쓰인다. 기존 (1,1,1)이었던 것을 (1,2,3)으로 바꾸어 상하로 길어진 모습
- labtest5는 지정된 angle만큼 한 축을 기준으로 회전시키는 `glRotatef`가 쓰였다. 회전된 상태로 window에 나타난다
- labtest6에서는 `gluLookAt` 함수가 쓰인다. 카메라가 관찰대상을 어떻게 바라볼 지 결정하는데 쓰인다.

labtest7, 8



```
xx = j / 500.0;  
yy = i / 500.0;  
zz = z / 500.0;  
xx1 = m1[0][0] * xx + m1[0][1] * yy + m1[0][2] * zz;  
yy1 = m1[1][0] * xx + m1[1][1] * yy + m1[1][2] * zz;  
zz1 = m1[2][0] * xx + m1[2][1] * yy + m1[2][2] * zz;  
//glVertex3f(xx1,yy1,zz1); // labtest 7 with ortho  
// labtest 8 see perspect  
//glVertex3f(xx, yy, zz);
```

- labtest 8에서는 xx, yy, zz
- labtest 7에서는 $m1$ 과 xx, yy, zz 를 내적인 값을 사용한다
- `glVertex3f`가 자료와 같은 정육면체를 나타내는것으로 생각된다