R E P O R T

실습과제 05

내용

과제 명세
동영상에서와 같은 기본 기능 구현
복합 변환 구현
현재의 모델뷰 행렬 출력
메뉴 콜백 구현
고찰 및 느낀점

과제 명세

실습과제 05

- 강의자료 4장 5절의 프로그램 구현하기
  - 동영상 참조
    - 그래픽스-4장-5절-1
    - 그래픽스-4장-5절-2
  - 동영상에서와 같은 기본 기능 구현
  - 복합 변환 구현
    - 동영상에서 설명한 프로그램에 있는 기능
    - 자신만의 복합 변환 1가지만 더 추가
  - 현재의 모델뷰 행렬 출력
    - 동영상 참조
  - 구현된 다양한 변환을 메뉴로 선택할 수 있도록 메뉴 콜백을 처리하라. 이를 위해 메뉴를 만들고, 처리하는 함수를 구현해야 한다.

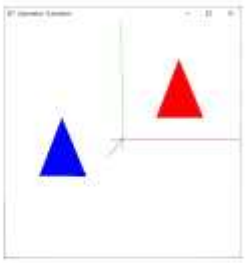
11

4.5 기하 변환의 구현

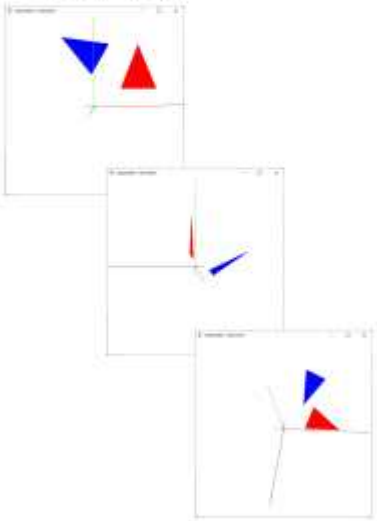
- 자신만의 기하변환 라이브러리 만들기
  - 다양한 변환 테스트
  - 행렬 적용 이해
  - GLUT 활용
- 구현 예

46

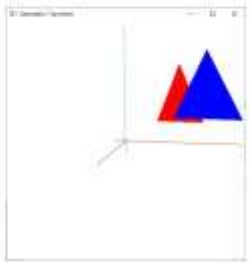
- 이동(t)



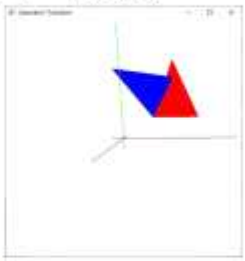
- 회전(z, x, y)



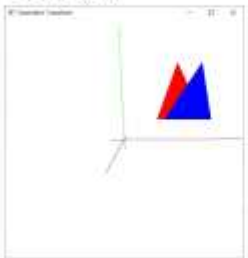
- 신축(s)



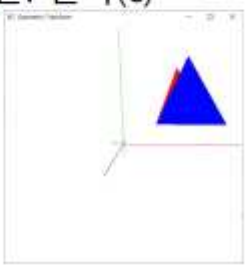
- 복합변환: 회전(Z)

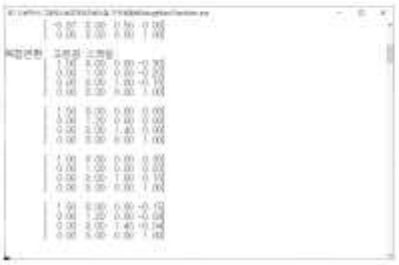


- 밀림(h)



- 복합변환: 신축(c)





자신만의 기하변환 라이브러리 만들기

- OpenGL에서는 기하변환을 위한 여러 가지 함수 제공
- 직접 만들어 보자.
  - 변환행렬의 생성: glkMatSet()
  - 변환행렬을 곱하기: glkMatMult(double* m1, double m2);
  - 행렬을 화면에 출력: glkMatPrint(double *m)
  - 변환행렬 생성
    - 항등행렬: glkMatIdentity(double* m)
    - 이동: glkMatTrans(double* m, double tx, double ty, double tz)
    - 신축: glkMatScale(double* m, double sx, double sy, double sz)
    - 회전(X,Y,Z축 중심): glkMatRotateZ(double* m, double a)
    - 밀림(X,Y,Z축 방향): glkMatShearX(double* m, double dy, double dz)
  - 정점 변환처리: glkTransform(double* m, double* p, double* q)
  - 기타 그리기 함수들: 선분 그리기, 삼각형 그리기, 좌표축 그리기

동영상에서와 같은 기본 기능 구현

- gllc.h

헤더파일

```
#pragma once
#include <stdio.h>
#include <memory.h> // memcpy 사용하기 위함
#include <gl/glut.h> //  glut를 사용하기 위함
#define _USE_MATH_DEFINES // M_PI 등을 사용하기 위함.
#include <math.h>
#define SIN(x) sin(x*M_PI / 180.) // degree 각을 이용한 sin
#define COS(x) cos(x*M_PI / 180.) // degree 각을 이용한 cos
```

3차원 행렬 구성 및 출력

```
// 변환 행렬 생성//
inline void glMatSet(double* m,
        double m00, double m01, double m02, double m03,
        double m10, double m11, double m12, double m13,
        double m20, double m21, double m22, double m23,
        double m30, double m31, double m32, double m33) {
        double mat[16] = { m00, m01, m02, m03, m10, m11, m12, m13, m20, m21, m22,
m23, m30, m31, m32, m33 };
        memcpy(m, mat, sizeof(double) * 16); // 메모리 복사 함수. strncpy와 유사하게 사용.
}
// 변환 행렬 곱하기 //
inline void glMatMult(double* m1, double* m2) {
        double n[16];
        for (int k = 0; k < 16; k++) {
                n[k] = 0;
                double* p = m1 + (k / 4) * 4;
                double* q = m2 + (k % 4);
                for (int i = 0; i < 4; i++)
                        n[k] += p[i] * q[i * 4];
        }
        memcpy(m1, n, sizeof(double) * 16);
}
// 행렬을 화면에 출력 //
inline void glMatPrint(double* m) {
        for (int i = 0; i < 4; i++) {
                printf("₩t[");
                for (int j = 0; j < 4; j++)
                        printf("%6.2f", m[i * 4 + j]);
                printf("]₩n");
        }
        printf("₩n");
}
```

변환 행렬 생성

```
// 항등행렬(i)
inline void glMatIdentity(double* m) {
	glMatSet(m, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
}
// 이동(t)
inline void glMatTrans(double* m, double tx, double ty, double tz) {
	glMatSet(m, 1, 0, 0, tx, 0, 1, 0, ty, 0, 0, 1, tz, 0, 0, 0, 1);
}

// 신축(s)
inline void glMatScale(double* m, double sx, double sy, double sz) {
	glMatSet(m, sx, 0, 0, 0, 0, sy, 0, 0, 0, 0, sz, 0, 0, 0, 0, 1);
}

// 회전(Z축 중심)
inline void glMatRotateZ(double* m, double a) {
	glMatSet(m, COS(a), -SIN(a), 0, 0, SIN(a), COS(a), 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
}

// 회전(X축 중심)
inline void glMatRotateX(double* m, double a) {
	glMatSet(m, 1, 0, 0, 0, 0, COS(a), -SIN(a), 0, 0, SIN(a), COS(a), 0, 0, 0, 0, 1);
}
// 회전(Y축 중심)
inline void glMatRotateY(double* m, double a) {
	glMatSet(m, COS(a), 0, SIN(a), 0, 0, 1, 0, 0, -SIN(a), 0, COS(a), 0, 0, 0, 0, 1);
}

// X축 밀림
inline void glMatShearX(double* m, double dy, double dz) {
	glMatSet(m, 1, dy, dz, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
}

// Y축 밀림
inline void glMatShearY(double* m, double dx, double dz) {
	glMatSet(m, 1, 0, 0, 0, dx, 1, dz, 0, 0, 0, 1, 0, 0, 0, 0, 1);
}

// Z축 밀림
inline void glMatShearZ(double* m, double dx, double dy) {
	glMatSet(m, 1, 0, 0, 0, 0, 1, 0, 0, dx, dy, 1, 0, 0, 0, 0, 1);
}
```

```
// 간단한 그리기 함수들 : 선분
inline void glLine(double x1, double y1, double z1, double x2, double y2, double z2) {
		glBegin(GL_LINES);
		glVertex3d(x1, y1, z1);
		glVertex3d(x2, y2, z2);
		glEnd();
}

// 간단한 그리기 함수들 : 삼각형
inline void glTriangle4d(double* p) {
		glBegin(GL_TRIANGLES);
		glVertex4dv(p);
		glVertex4dv(p + 4);
		glVertex4dv(p + 8);
		glEnd();
}
// 간단한 그리기 함수들 : 좌표축
inline void glCoord() {                              // 축의 색 표현
		glBegin(GL_LINES);
		glColor3d(1, 0, 0); glVertex3d(-0.1, 0, 0);          glVertex3d(1, 0, 0);
		glColor3d(0, 1, 0); glVertex3d(0, -0.1, 0); glVertex3d(0, 1, 0);
		glColor3d(0, 0, 1); glVertex3d(0, 0, -0.1);          glVertex3d(0, 0, 1);
		glEnd();
}
```

- **BasicTransform.cpp**

변수 정의

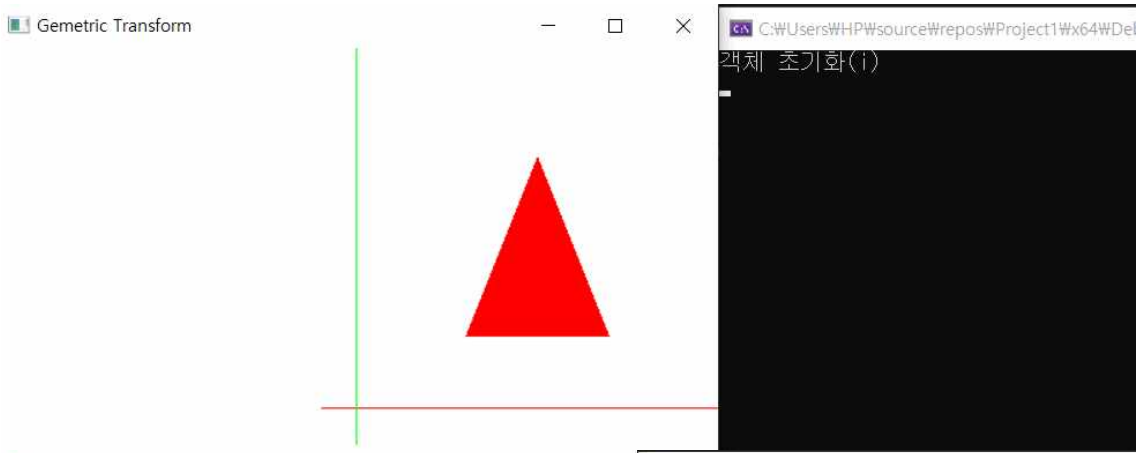
```
#include "gllc.h"
#include <windows.h>
// 동차좌표계로 표시함
static double P[12] = { 0.3, 0.2, 0.1, 1,        0.7, 0.2, 0.1, 1,    0.5, 0.7, 0.1, 1 };
static double Q[12];

static char objNum = -1;                    // 객체 번호. 메뉴 이벤트로 선택 예정
static int PrevX, PrevY;                // 마우스 X, Y  좌표 위치
```

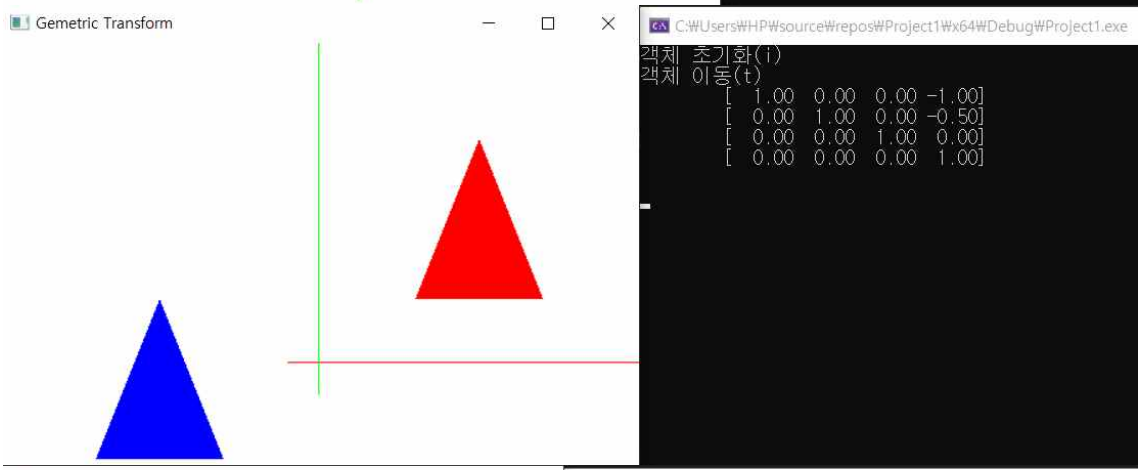
기본 기능 및 복합 기능 구현

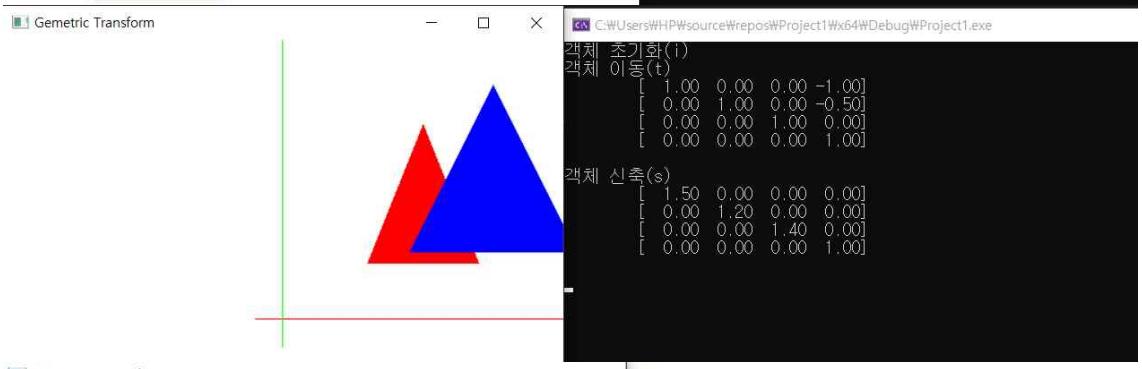
```
void display() {
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0, 0.0, 0.0);
        glTriangle4d(P);
        glColor3f(0.0, 0.0, 1.0);
        glTriangle4d(Q);
        glCoord();
        glFlush();
}
void transformTri(double* m, double* p, double* q) {
        glTransform(m, p, q);
        glTransform(m, p + 4, q + 4);
        glTransform(m, p + 8, q + 8);
}
void keyboard(unsigned char key, int x, int y) {
        double m1[16], m2[16], m3[16];

        if (key == 'i') {                                   // 초기화
                printf("객체 초기화(i)₩n");
                glMatrixMode(GL_MODELVIEW);
                glLoadIdentity();
        }
        else if (key == 't') {                              // 이동
                printf("객체 이동(t)₩n");
                glMatTrans(m1, -1, -0.5, 0);
                glMatPrint(m1);
                transformTri(m1, P, Q);
        }
        else if (key == 's') {                              // 신축
                printf("객체 신축(s)₩n");
                glMatScale(m1, 1.5, 1.2, 1.4);
                glMatPrint(m1);
                transformTri(m1, P, Q);
        }
```

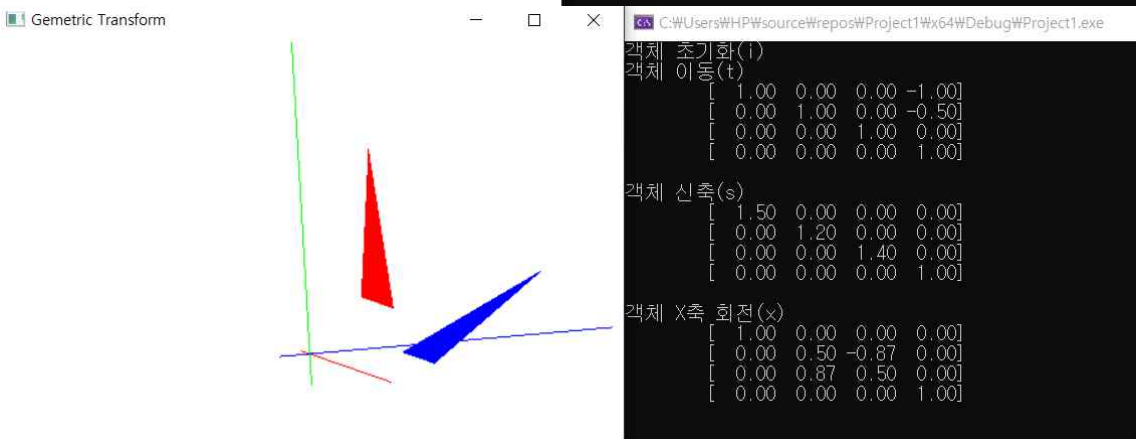
```
        else if (key == 'x') {                          // X축 회전
                printf("객체 X축 회전(x)₩n");
                glMatRotateX(m1, 60);
                glMatPrint(m1);
                transformTri(m1, P, Q);    }
        else if (key == 'y') {                                  // Y축 회전
                printf("객체 Y축 회전(y)₩n");
                glMatRotateY(m1, 60);
                glMatPrint(m1);
                transformTri(m1, P, Q);    }
        else if (key == 'z') {                          // Z축 회전
                printf("객체 Z축 회전(z)₩n");
                glMatRotateZ(m1, 60);
                glMatPrint(m1);
                transformTri(m1, P, Q);    }
        else if (key == 'Z') {                          // 복합변환 : 회전
                glMatTrans(m1, -P[0], -P[1], -P[2]);
                glMatRotateZ(m2, 60);
                glMatTrans(m3, P[0], P[1], P[2]);

                printf("복합변환 : 회전(Z)₩n");
                glMatMult(m3, m2);
                glMatMult(m3, m1);
                transformTri(m3, P, Q);
        }
        else if (key == 'c') {                          // 복합변환 : 신축
                glMatTrans(m1, P[0], P[1], P[2]);
                glMatScale(m2, 1.5, 1.2, 1.4);
                glMatTrans(m3, -P[0], -P[1], -P[2]);
                printf("복합변환 : 신축(c)₩n");
                glMatPrint(m1);
                glMatPrint(m2);
                glMatPrint(m3);
                glMatMult(m1, m2);
                glMatMult(m1, m3);
                glMatPrint(m1);
                transformTri(m1, P, Q);
        }
        else if (key == 'h') {                          // 밀림
                printf("밀림(h)₩n");
                glMatShearX(m1, 0.3, 0);
                transformTri(m1, P, Q);
        }
        else if (key == 'q') exit(0);
        glutPostRedisplay();      }
```

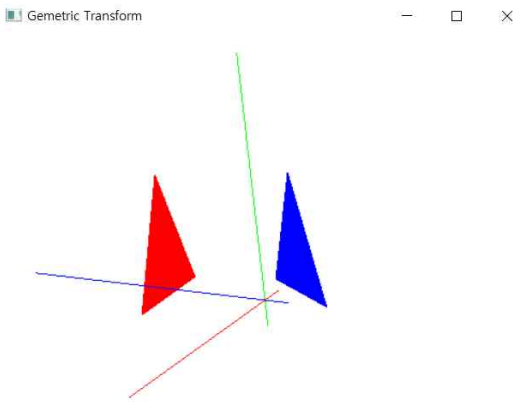
- 출력결과 (객체 초기화, 이동, 신축 등)



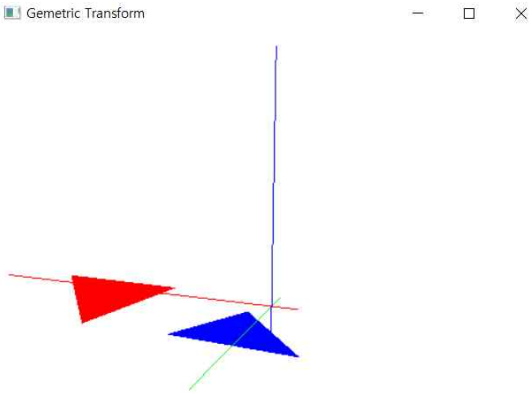




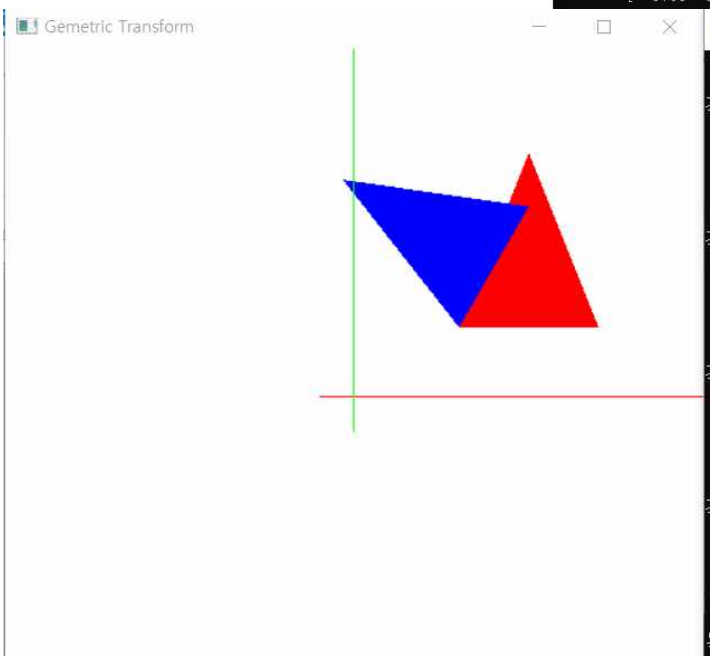




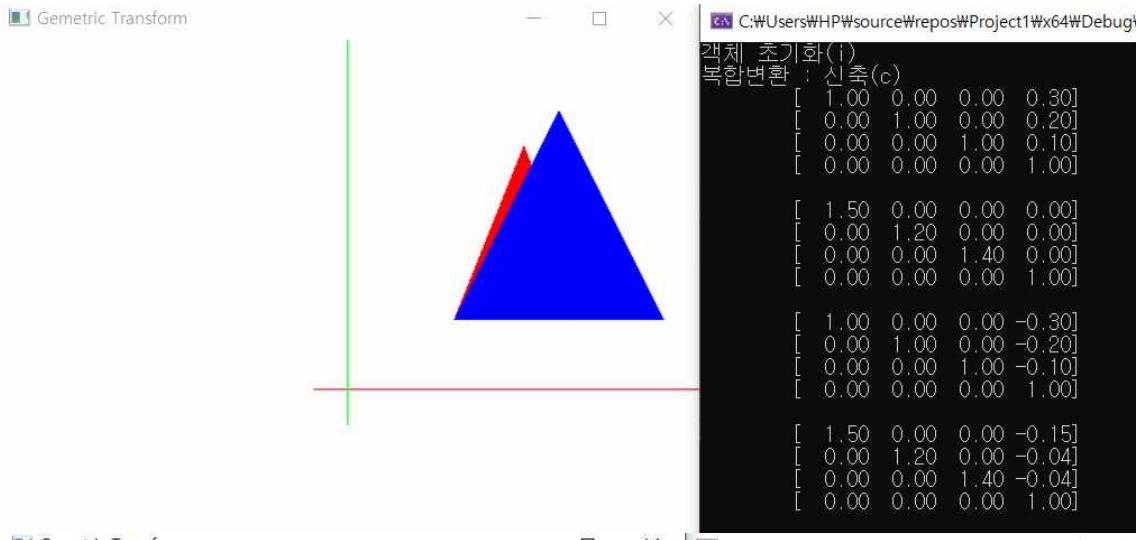

```
C:₩Users₩HP₩source₩repos₩Project1₩x64₩Debug₩Project1.exe
객체 초기화(i)
객체 이동(t)
        [  1.00  0.00  0.00 -1.00]
        [  0.00  1.00  0.00 -0.50]
        [  0.00  0.00  1.00  0.00]
        [  0.00  0.00  0.00  1.00]

객체 신축(s)
        [  1.50  0.00  0.00  0.00]
        [  0.00  1.20  0.00  0.00]
        [  0.00  0.00  1.40  0.00]
        [  0.00  0.00  0.00  1.00]

객체 X축 회전(x)
        [  1.00  0.00  0.00  0.00]
        [  0.00  0.50 -0.87  0.00]
        [  0.00  0.87  0.50  0.00]
        [  0.00  0.00  0.00  1.00]

객체 Y축 회전(y)
        [  0.50  0.00  0.87  0.00]
        [  0.00  1.00  0.00  0.00]
        [ -0.87  0.00  0.50  0.00]
        [  0.00  0.00  0.00  1.00]
```

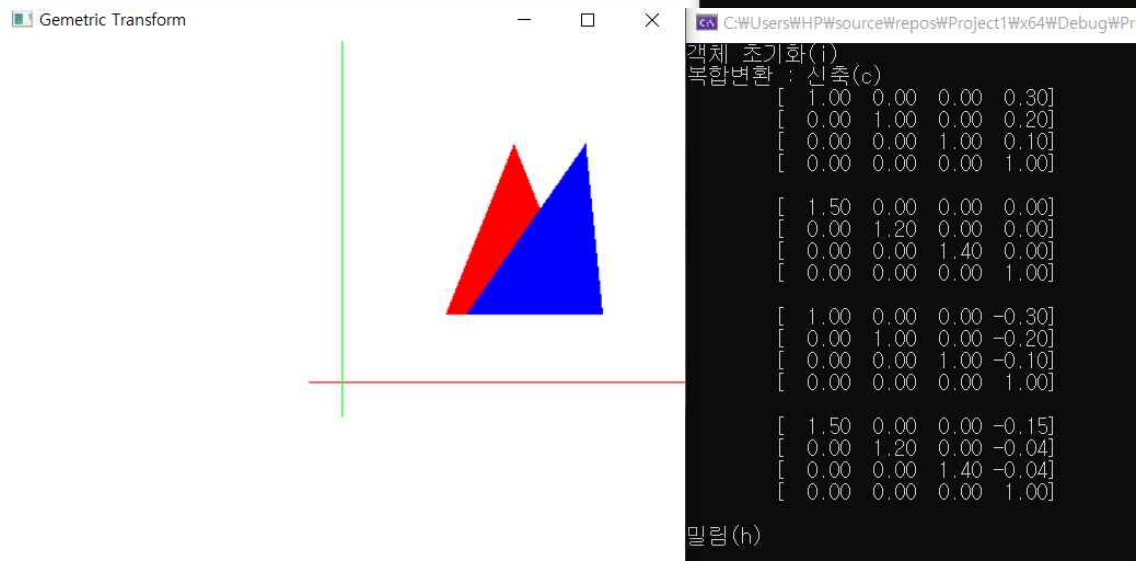



```
C:₩Users₩HP₩source₩repos₩Project1₩x64₩Debug₩Project1.exe
        [  1.00  0.00  0.00 -1.00]
        [  0.00  1.00  0.00 -0.50]
        [  0.00  0.00  1.00  0.00]
        [  0.00  0.00  0.00  1.00]

객체 신축(s)
        [  1.50  0.00  0.00  0.00]
        [  0.00  1.20  0.00  0.00]
        [  0.00  0.00  1.40  0.00]
        [  0.00  0.00  0.00  1.00]

객체 X축 회전(x)
        [  1.00  0.00  0.00  0.00]
        [  0.00  0.50 -0.87  0.00]
        [  0.00  0.87  0.50  0.00]
        [  0.00  0.00  0.00  1.00]

객체 Y축 회전(y)
        [  0.50  0.00  0.87  0.00]
        [  0.00  1.00  0.00  0.00]
        [ -0.87  0.00  0.50  0.00]
        [  0.00  0.00  0.00  1.00]

객체 Z축 회전(z)
        [  0.50 -0.87  0.00  0.00]
        [  0.87  0.50  0.00  0.00]
        [  0.00  0.00  1.00  0.00]
        [  0.00  0.00  0.00  1.00]
```




```
C:₩Users₩HP₩source₩repos₩Project1₩x64₩Debug
        [  0.00  0.00  0.00  1.00]

객체 신축(s)
        [  1.50  0.00  0.00  0.00]
        [  0.00  1.20  0.00  0.00]
        [  0.00  0.00  1.40  0.00]
        [  0.00  0.00  0.00  1.00]

객체 X축 회전(x)
        [  1.00  0.00  0.00  0.00]
        [  0.00  0.50 -0.87  0.00]
        [  0.00  0.87  0.50  0.00]
        [  0.00  0.00  0.00  1.00]

객체 Y축 회전(y)
        [  0.50  0.00  0.87  0.00]
        [  0.00  1.00  0.00  0.00]
        [ -0.87  0.00  0.50  0.00]
        [  0.00  0.00  0.00  1.00]

객체 Z축 회전(z)
        [  0.50 -0.87  0.00  0.00]
        [  0.87  0.50  0.00  0.00]
        [  0.00  0.00  1.00  0.00]
        [  0.00  0.00  0.00  1.00]

복합변환 : 회전(Z)
```


Gemetric Transform
C:₩Users₩HP₩source₩repos₩Project1₩x64₩Debug₩
객체 초기화(i)
복합변환 : 신축(c)
[1.00 0.00 0.00 0.30]
[0.00 1.00 0.00 0.20]
[0.00 0.00 1.00 0.10]
[0.00 0.00 0.00 1.00]
[1.50 0.00 0.00 0.00]
[0.00 1.20 0.00 0.00]
[0.00 0.00 1.40 0.00]
[0.00 0.00 0.00 1.00]
[1.00 0.00 0.00 -0.30]
[0.00 1.00 0.00 -0.20]
[0.00 0.00 1.00 -0.10]
[0.00 0.00 0.00 1.00]
[1.50 0.00 0.00 -0.15]
[0.00 1.20 0.00 -0.04]
[0.00 0.00 1.40 -0.04]
[0.00 0.00 0.00 1.00]


Gemetric Transform
C:₩Users₩HP₩source₩repos₩Project1₩x64₩Debug₩Pr
객체 초기화(i)
복합변환 : 신축(c)
[1.00 0.00 0.00 0.30]
[0.00 1.00 0.00 0.20]
[0.00 0.00 1.00 0.10]
[0.00 0.00 0.00 1.00]
[1.50 0.00 0.00 0.00]
[0.00 1.20 0.00 0.00]
[0.00 0.00 1.40 0.00]
[0.00 0.00 0.00 1.00]
[1.00 0.00 0.00 -0.30]
[0.00 1.00 0.00 -0.20]
[0.00 0.00 1.00 -0.10]
[0.00 0.00 0.00 1.00]
[1.50 0.00 0.00 -0.15]
[0.00 1.20 0.00 -0.04]
[0.00 0.00 1.40 -0.04]
[0.00 0.00 0.00 1.00]
밀림(h)

복합 변환 구현

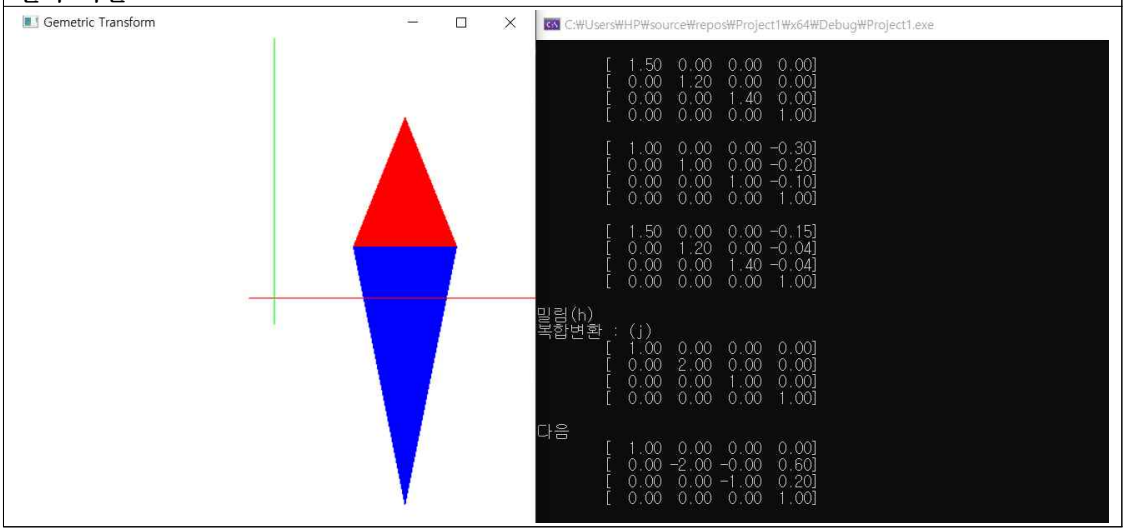
복합 변환

: 객체 신축, 이동, X축 회전과 행렬 곱셈을 통해 빨간 삼각형의 밑변을 공유하고 높이가 더 긴 파란 삼각형을 만들어 내었다.

Y축을 2배 신축했고 위치를 빨간 삼각형 밑면과 겹치도록 이동시켰고 X축을 중심으로 180도 회전시켜 다음과 같은 결과가 나왔다.

```
        else if (key == 'j') {                  // 자신만의 복합변환
                glMatScale(m1, 1, 2.0, 1);
                glMatTrans(m2, 0, 0.3, 0.2);
                glMatRotateX(m3, 180);
                printf("복합변환 : (j)\n");
                glMatPrint(m1);

                glMatMult(m1, m2);
                glMatMult(m1, m3);
                printf("다음\n");

                glMatPrint(m1);
                transformTri(m1, P, Q);
        }
```

결과 화면



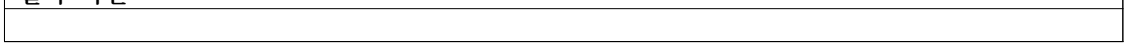
메뉴 콜백 구현

```
void mouseClick(int button, int state, int x, int y)
        // 오른쪽 마우스 클릭 시 메뉴 접근 ( 자세한 사항 initMenu, myMenu에 구현 )
        glutAttachMenu(GLUT_RIGHT_BUTTON);
```

메뉴 콜백

```
void myMenu(int id) { // 메뉴를 그리는 함수
        if (id == 0) keyboard('i', 0, 0);
        else if (id == 1) keyboard('t', 0, 0);
        else if (id == 2) keyboard('s', 0, 0);
        else if (id == 3) keyboard('x', 0, 0);
        else if (id == 4) keyboard('y', 0, 0);
        else if (id == 5) keyboard('z', 0, 0);
        else if (id == 6) keyboard('Z', 0, 0);
        else if (id == 7) keyboard('c', 0, 0);
        else if (id == 8) keyboard('h', 0, 0);
        else if (id == 9) keyboard('j', 0, 0);
        else if (id == 10) exit(0);
        glutPostRedisplay();
}
void initMenu(void) {       // 우클릭시 메뉴 접근
        GLint MyMainMenuID = glutCreateMenu(myMenu);
        glutAddMenuEntry("Initial(i)", 0);
        glutAddMenuEntry("t", 1);
        glutAddMenuEntry("s", 2);
        glutAddMenuEntry("x", 3);
        glutAddMenuEntry("y", 4);
        glutAddMenuEntry("z", 5);
        glutAddMenuEntry("Z", 6);
        glutAddMenuEntry("c", 7);
        glutAddMenuEntry("h", 8);
        glutAddMenuEntry("j", 9);
        glutAddMenuEntry("Exit(q)", 10);
}
```

결과 화면




Initial(i)
t
s
x
y
z
Z
c
h
j
Exit(q)

고찰 및 느낀점

과제를 하면서 "교수님의 코드 없이 맨몸으로 이걸 할 수 있을까?" 하는 의문이 들기 시작했다. 나만의 변환을 만들면서 처음에는 모래시계처럼 점대칭과 같은 모양을 만들고 싶었는데 막상 출력하려고 하니 파란 삼각형이 보이지 않아서 실패했었고, 어디서 실수한 건지 파악을 하다 실패하여 그림자 같은 모양의 밑변을 공유하는 삼각형을 만들게 되었다. 로봇을 조립할 때는 어떻게 해야 할지 걱정이 들기 시작했다. 이번 과제를 되돌아보는 시간을 가지면서 조금 더 깊은 이해를 해야겠다고 느꼈다.