



R E P O R T

컴퓨터그래픽스및실습 실습과제 05

과목명	컴퓨터그래픽스및실습
분반	01
교수	최영규
학번	2020136129
이름	최수연
제출일	2022년 10월 10일 월요일

문제 분석 및 해결 방법

[문제 분석]

- 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)`
- 팝업 메뉴 만들기
- 복합변환: Z축 중심 회전
- 복합변환: 신축
- 자신만의 복합변환 만들기
- 정점 변환처리: `glkTransform(double* m, double* p, double* q)`
- 기타 그리기 함수들: 선분 그리기, 삼각형 그리기, 좌표축 그리기

[해결 방법]

- 헤더 파일을 따로 만들어 행렬 구성을 위해 함수를 하나 생성하고, 인자를 받아 행렬 곱을 계산하는 함수를 하나 만든다.
- 헤더 파일에는 여러 변환행렬을 계산해줄 함수들을 생성하고 선분, 삼각형, 좌표축을 생성한다.
- 소스 파일에는 위 헤더 파일을 `include`하고, 초기 삼각형과 변환행렬을 통해 변환된 삼각형을 출력하기 위해 삼각형 정적 변수를 선언한다.
- 화면에 삼각형을 출력하는 함수와 정점 변환 처리하는 함수, key 별로 변환행렬이 적용된 삼각형을 표현하기 위해 함수를 생성한다.
- 마우스와 관련된 `mouseClick`과 `mouserMotion` 함수를 생성한다.
- 팝업 메뉴를 위해 메뉴와 키보드를 연결하는 `myMenu` 함수와 팝업 메뉴의 목록을 생성하는 `initMenu` 함수를 생성한다.
- 그리고 제대로 된 객체 확인을 위해 `reshape` 콜백 함수도 등록한다.

주요 설명 코드

<2020136129_5.h>

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

// 변환행렬 생성
inline void glkMatSet(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); // mat를 m에 복사
}
inline void glkMatMult(double* m1, double* m2) { // 변환행렬 곱하기(m1 *= 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);
}
inline void glkMatPrint(double* m) { // 행렬 출력
    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");
}
```

```

inline void glkMatIdentity(double* m) { // 항등행렬
    glkMatSet(m, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
}
inline void glkMatTrans(double* m, double tx, double ty, double tz) { // 이동
    glkMatSet(m, 1, 0, 0, tx, 0, 1, 0, ty, 0, 0, 1, tz, 0, 0, 0, 1);
}
inline void glkMatScale(double* m, double sx, double sy, double sz) { // 신축
    glkMatSet(m, sx, 0, 0, 0, 0, sy, 0, 0, 0, 0, sz, 0, 0, 0, 0, 1);
}
inline void glkMatRotateZ(double* m, double a) { // Z축 중심 회전
    glkMatSet(m, COS(a), -SIN(a), 0, 0, SIN(a), COS(a), 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
}
inline void glkMatRotateX(double* m, double a) { // X축 중심 회전
    glkMatSet(m, 1, 0, 0, 0, 0, COS(a), -SIN(a), 0, 0, SIN(a), COS(a), 0, 0, 0, 0, 1);
}
inline void glkMatRotateY(double* m, double a) { // Y축 중심 회전
    glkMatSet(m, COS(a), 0, SIN(a), 0, 0, 1, 0, 0, -SIN(a), 0, COS(a), 0, 0, 0, 0, 1);
}
inline void glkMatShearX(double* m, double dy, double dz) { // X축 방향 밀림
    glkMatSet(m, 1, dy, dz, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
}
inline void glkMatShearY(double* m, double dx, double dz) { // Y축 방향 밀림
    glkMatSet(m, 1, 0, 0, 0, dx, 1, dz, 0, 0, 0, 1, 0, 0, 0, 0, 1);
}
inline void glkMatShearZ(double* m, double dx, double dy) { // Z축 방향 밀림
    glkMatSet(m, 1, 0, 0, 0, 0, 1, 0, 0, dx, dy, 1, 0, 0, 0, 0, 1);
}
inline void glkTransform(double* m, double* p, double* q) { // 정점의 변환함수 [q] = [M] [p]
    q[0] = q[1] = q[2] = q[3] = 0;
    for (int i = 0; i < 4; i++) {
        q[0] += m[i] * p[i];
        q[1] += m[i + 4] * p[i];
        q[2] += m[i + 8] * p[i];
        q[3] += m[i + 12] * p[i];
    }
}

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

```

```

inline void glkTriangle4d(double* p) { // 삼각형
    glBegin(GL_TRIANGLES);
    glVertex4dv(p);
    glVertex4dv(p + 4);
    glVertex4dv(p + 8);
    glEnd();
}

inline void glkCoord() { // 좌표계
    glBegin(GL_LINES);
    glColor3d(1, 0, 0); // 색 glVertex3d(-0.1, 0, 0); // 시작 위치 glVertex3d(1, 0, 0); // 끝 위치
    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();
}

```

<2020136129_5.cpp>

```

#include "2020136129_5.h"
#include <windows.h>

// 동차좌표계로 표시
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 }; // 기존 P삼각형 세 점의 좌표
static double Q[12]; // 변환된 Q삼각형
static double R[12]; // 변환된 R삼각형
static double S[12]; // 변환된 S삼각형
static double T[12]; // 변환된 T삼각형
static double U[12]; // 변환된 U삼각형

void display() {
    glClear(GL_COLOR_BUFFER_BIT); // 화면 초기화
    glColor3f(1.0, 0.0, 0.0); // 빨강
    glkTriangle4d(P); // P삼각형
    glColor3f(0.0, 0.0, 1.0); // 파랑
    glkTriangle4d(Q); // Q삼각형
    glColor3f(0.0, 1.0, 0.0); // 연두
    glkTriangle4d(R); // R삼각형
    glColor3f(1.0, 0.0, 1.0); // 노랑
    glkTriangle4d(T); // T삼각형
    glColor3f(1.0, 1.0, 0.0); // 분홍
    glkTriangle4d(S); // S삼각형
    glColor3f(0.0, 1.0, 1.0); // 하늘
    glkTriangle4d(U); // U삼각형
    glkCoord(); // 좌표계
    glFlush(); // 화면 갱신
}

```

```

void transformTri(double* m, double* p, double* q) { // 정점 변환 처리(e.g. m 행렬을 통해 P 삼각형의 좌
표들을 각각 Q 삼각형의 좌표로 변환)
    glkTransform(m, p, q);
    glkTransform(m, p + 4, q + 4);
    glkTransform(m, p + 8, q + 8);
}

void keyboard(unsigned char key, int x, int y) {
    double m1[16], m2[16], m3[16]; // 3개의 행렬 선언
    if (key == 'i') { // 초기화(i)
        printf("초기화(i)\n");
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
    } else if (key == 't') { // 이동(t)
        printf("이동(t)\n");
        glkMatTrans(m1, -1, -0.5, 0); // (-1, -0.5, 0) 행렬을 통해 이동 변환행렬 생성
        glkMatPrint(m1); // 변환행렬 화면에 출력
        transformTri(m1, P, Q); // 정점 변환 처리
    } else if (key == 's') { // 신축(s)
        printf("신축(s)\n");
        glkMatScale(m1, 1.5, 1.2, 1.4); // (1.5, 1.2, 1.4) 행렬을 통해 신축 변환행렬 생성
        glkMatPrint(m1); // 변환행렬 화면에 출력
        transformTri(m1, P, Q); // 정점 변환 처리
    } else if (key == 'x') { // X축 회전(x)
        printf("X축 회전(x)\n");
        glkMatRotateX(m1, 60); // 60을 헤더 파일의 함수의 매개변수로 받아 x축 회전 변환행렬 생성
        glkMatPrint(m1); // 변환행렬 화면에 출력
        transformTri(m1, P, Q); // 정점 변환 처리
    } else if (key == 'y') { // Y축 회전(y)
        printf("Y축 회전(y)\n");
        glkMatRotateY(m1, 60); // 60을 헤더 파일의 함수의 매개변수로 받아 y축 회전 변환행렬 생성
        glkMatPrint(m1); // 변환행렬 화면에 출력
        transformTri(m1, P, Q); // 정점 변환 처리
    } else if (key == 'z') { // Z축 회전(z)
        printf("Z축 회전(z)\n");
        glkMatRotateZ(m1, 60); // 60을 헤더 파일의 함수의 매개변수로 받아 z축 회전 변환행렬 생성
        glkMatPrint(m1); // 변환행렬 화면에 출력
        transformTri(m1, P, Q); // 정점 변환 처리
    } else if (key == 'Z') { // 복합변환: 회전(Z)
        printf("복합변환: 회전(Z)\n");
        glkMatTrans(m1, -P[0], -P[1], -P[2]); // 삼각형 꼭짓점 중 왼쪽 밑 꼭짓점 중심으로 회전
        glkMatRotateZ(m2, 60); // 60만큼 회전
        glkMatTrans(m3, P[0], P[1], P[2]); // 기존 삼각형
        glkMatMult(m3, m2); // 변환행렬 곱
    }
}

```

```

        glkMatMult(m3, m1); // 변환행렬 곱
        glkMatPrint(m3); // 변환행렬 화면에 출력
        transformTri(m3, P, Q); // 정점 변환 처리
    } else if (key == 'c') { // 복합변환: 신축(c)
        printf("복합변환: 신축(c)\n");
        glkMatTrans(m1, P[0], P[1], P[2]); // 기존 삼각형
        glkMatScale(m2, 1.5, 1.2, 1.4); // 확장할 크기
        glkMatTrans(m3, -P[0], -P[1], -P[2]); // 삼각형 꼭짓점 중 왼쪽 밑 꼭짓점 중심으로 신축
        glkMatPrint(m3);
        glkMatPrint(m2);
        glkMatPrint(m1);
        glkMatMult(m1, m2); // 변환행렬 곱
        glkMatMult(m1, m3); // 변환행렬 곱
        glkMatPrint(m1); // 변환행렬 화면에 출력
        transformTri(m1, P, Q); // 정점 변환 처리
    } else if (key == 'h') { // 밀림(h)
        printf("밀림(h)\n");
        glkMatShearX(m1, 0.3, 0);
        glkMatPrint(m1); // 변환행렬 화면에 출력
        transformTri(m1, P, Q); // 정점 변환 처리
    } else if (key == 'k') { // 자신만의 복합변환(k): 별 그리기
        printf("자신만의 복합변환: 별 그리기(k)\n");
        // 이동 + z축 중심 회전(복합변환: 회전(Z) 사용)
        glkMatTrans(m1, -P[0], -P[1], -P[2]);
        glkMatRotateZ(m2, 60);
        glkMatTrans(m3, P[0], P[1], P[2]);
        glkMatMult(m3, m2); // 변환행렬 곱
        glkMatMult(m3, m1); // 변환행렬 곱
        // 이동을 통해 별 모양 잡기
        glkMatTrans(m2, -0.4, 0, 0);
        glkMatMult(m3, m2); // 변환행렬 곱
        glkMatPrint(m3);
        // 각 삼각형을 위 변환행렬로 정점 변환 처리
        transformTri(m3, P, Q);
        transformTri(m3, Q, R);
        transformTri(m3, R, S);
        transformTri(m3, S, T);
        transformTri(m3, T, U);
    } else if (key == 'q') exit(0); // 종료(q)
    glutPostRedisplay();
}

static double PrevX, PrevY; // 마우스 위치 저장
void mouseClicked(int button, int state, int x, int y) { ... 생략 ... }

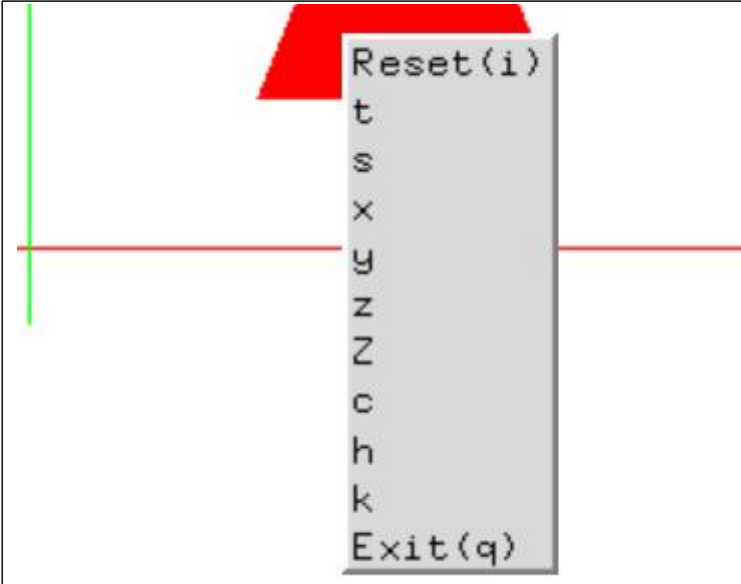
```

```

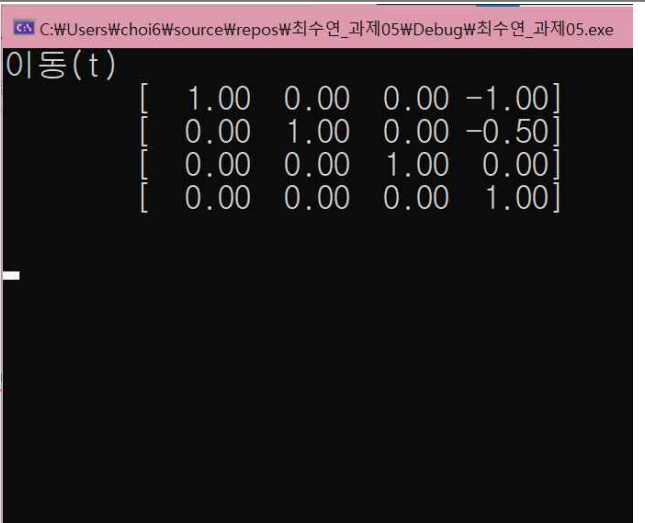
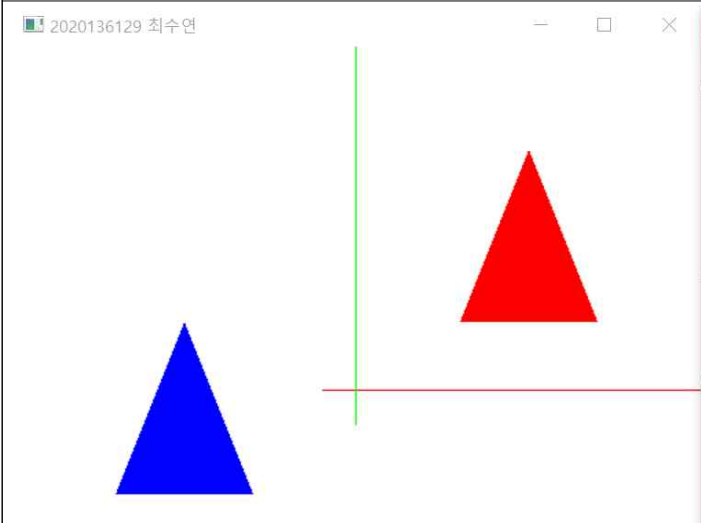
void mouseMotion(GLint x, GLint y) { // 마우스에 따라 화면 전환 ... 생략 ... }
void myMenu(int id) { // 팝업 메뉴 그리기
    if (id == 0) keyboard('i', 0, 0);
    else if (id == 1) keyboard('t', 0, 0);
    ... 생략 ...
    else if (id == 10) exit(0);
    glutPostRedisplay();
}
void initMenu(void) { // 우클릭 시 팝업 메뉴
    GLint MyMainMenuID = glutCreateMenu(myMenu);
    glutAddMenuEntry("Reset(i)", 0);
    glutAddMenuEntry("t", 1);
    ... 생략 ...
    glutAddMenuEntry("Exit(q)", 10);
}
void reshape(int width, int height) { ... 생략 ... }
int main(int argc, char** argv) { ... 생략 ... }

```


테스트 결과

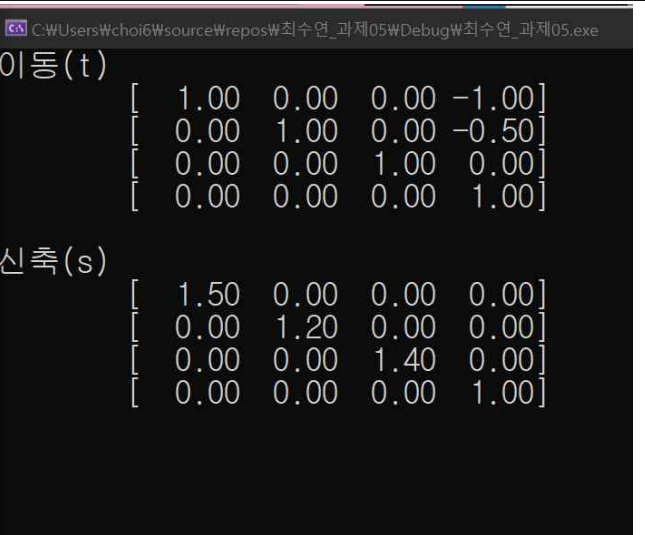
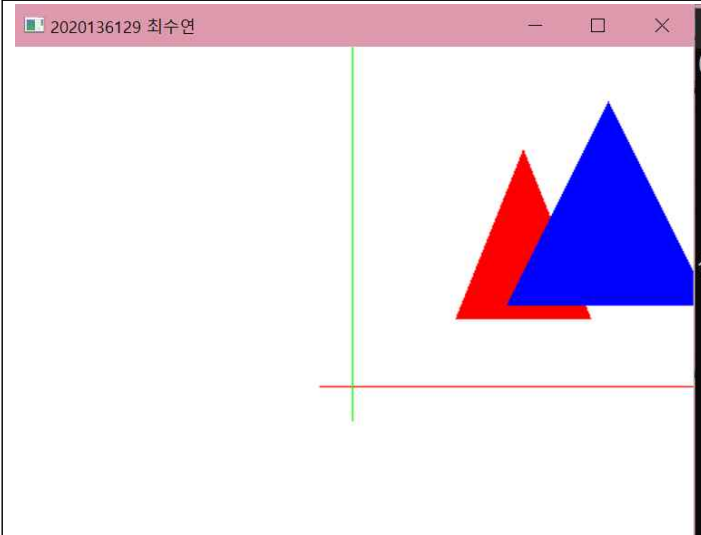


Reset(i)
t
s
x
y
z
Z
c
h
k
Exit(q)



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]

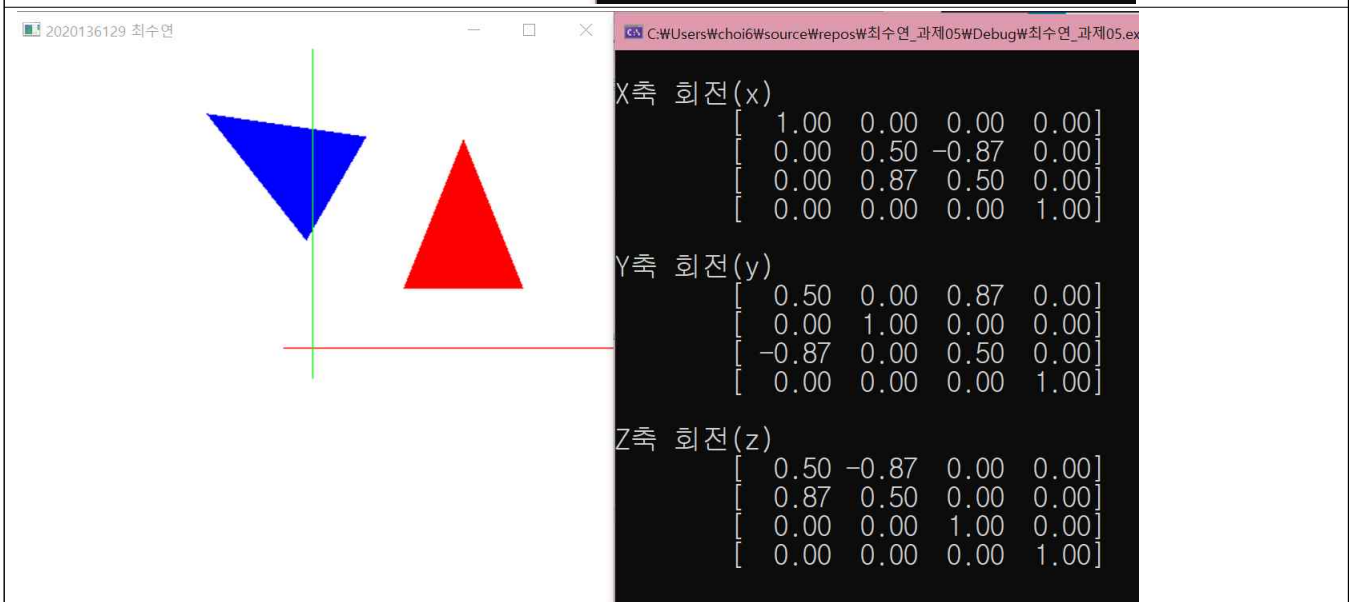
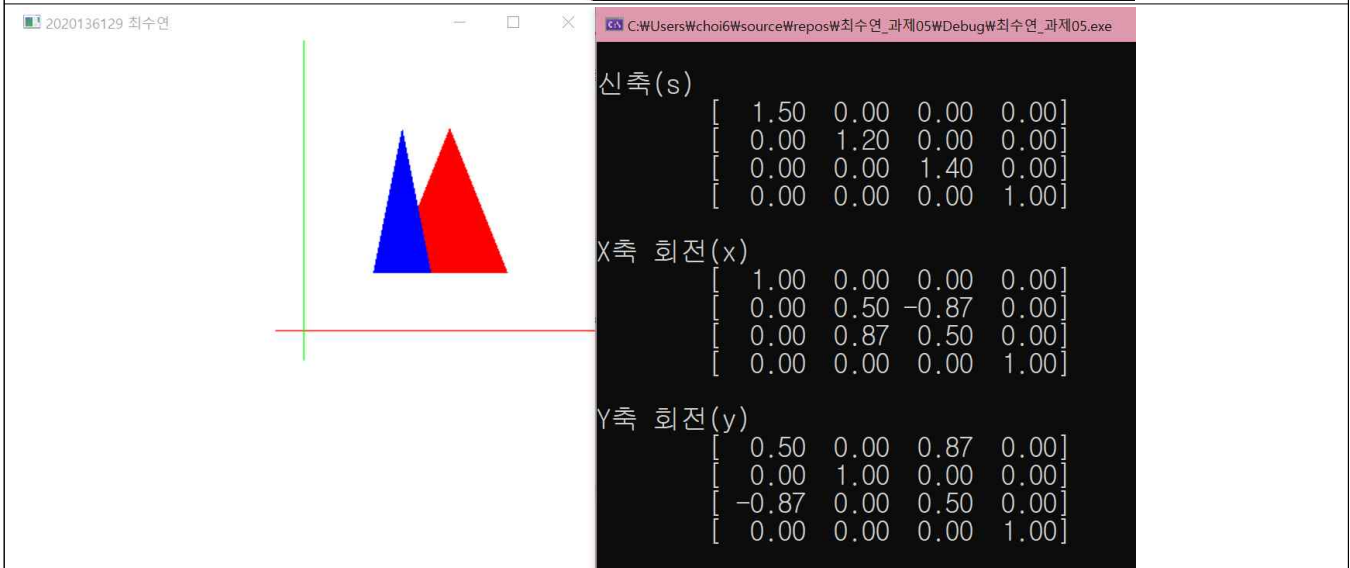
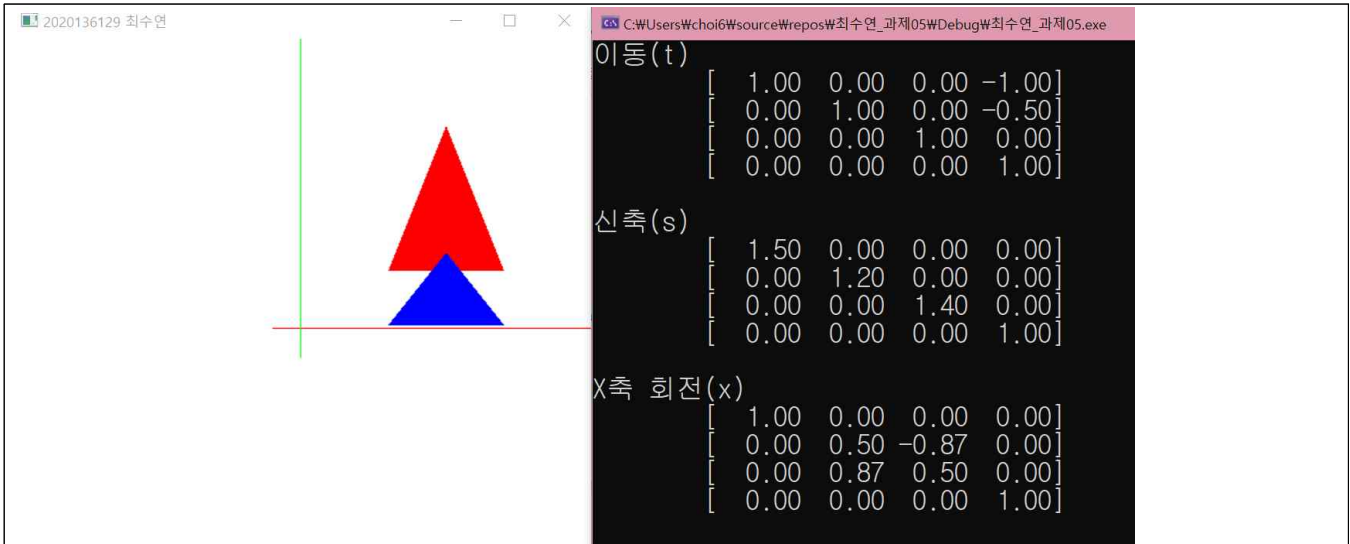


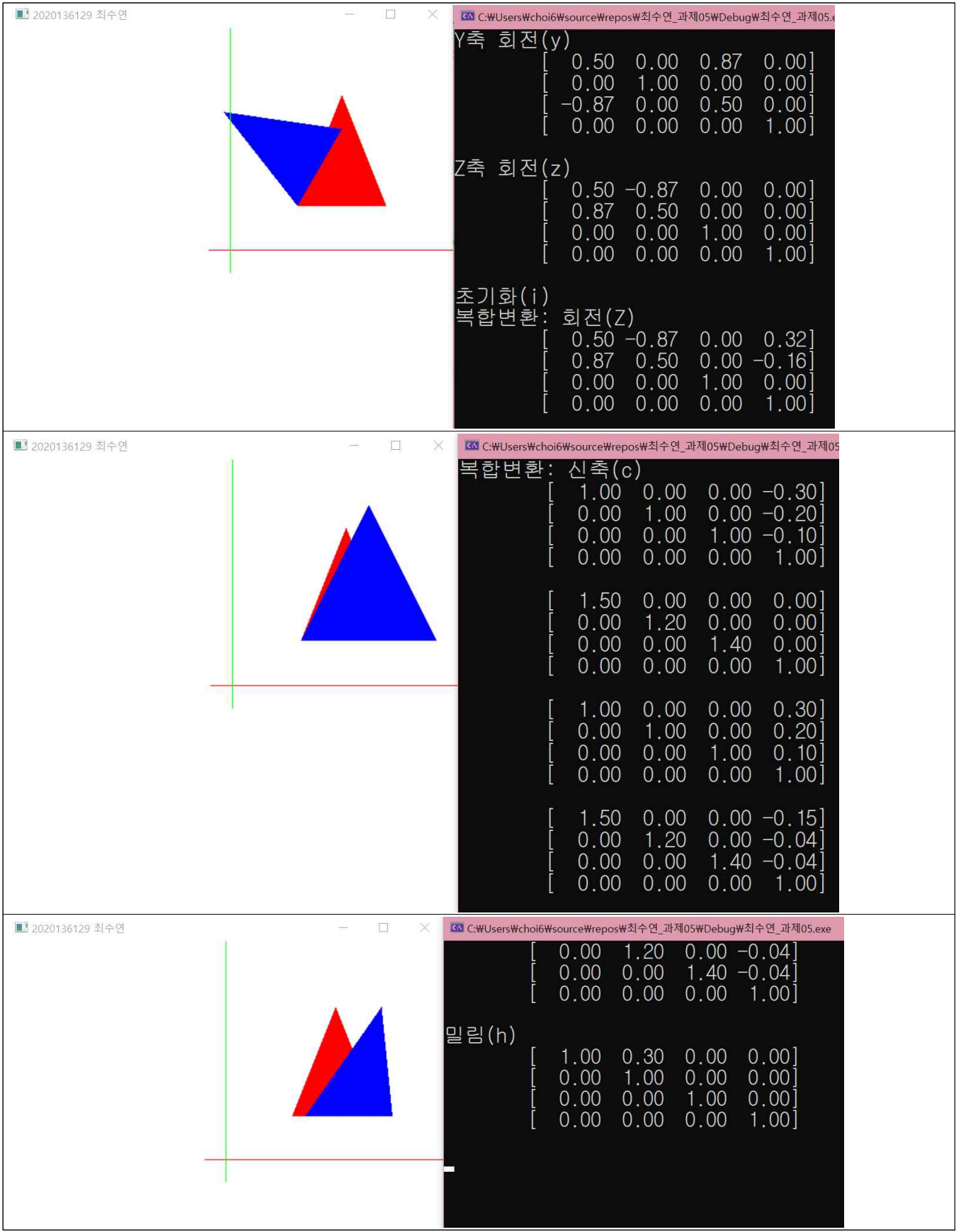
이동(t)

[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]

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]





2020136129 최수연



C:\Users\Wchoi6\source\repos\최수연_과제05\Debug\최수연_과제05.exe

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]

2020136129 최수연

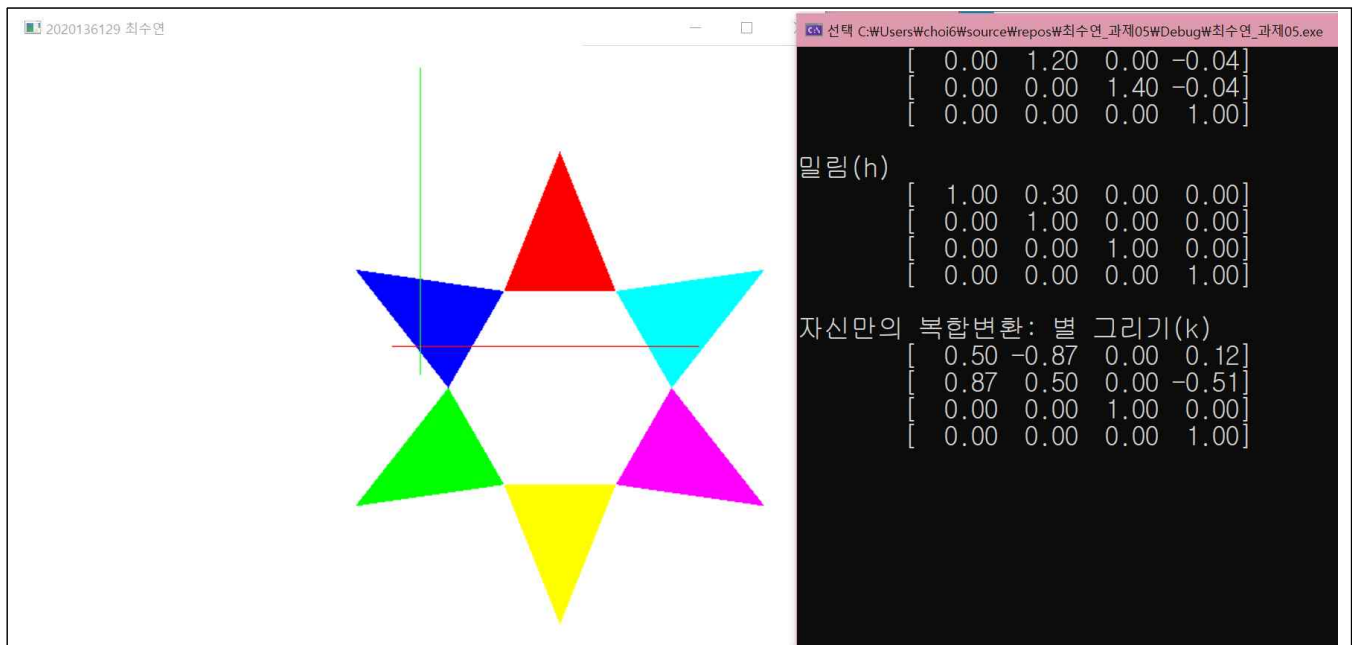


C:\Users\Wchoi6\source\repos\최수연_과제05\Debug\최수연_과제05.exe

밀림 (h)

[0.00	1.20	0.00	-0.04]
[0.00	0.00	1.40	-0.04]
[0.00	0.00	0.00	1.00]

[1.00	0.30	0.00	0.00]
[0.00	1.00	0.00	0.00]
[0.00	0.00	1.00	0.00]
[0.00	0.00	0.00	1.00]



고찰 및 느낀점

4장 수업에서 배운 행렬과 벡터 내용을 너무 오랜만에 해서 그런지 헷갈리는 부분도 많았고 처음 과제를 받았을 때 한 번에 이해가 제대로 되지 않았다. 그래도 교수님께서 강의 영상을 올려주셔서 좀 더 쉽게 이해할 수 있었고 덕분에 과제도 잘 마무리할 수 있었다. 아직 모든 변환행렬에 대해 어떤 변환행렬이 어떤 화면을 구현하는지는 완벽히 이해하진 못했지만, 과제를 통해 어느 정도 여러 행렬을 사용해봄으로써 처음보단 좀 더 감을 잡을 수 있었던 것 같다.