목차

1. 문제에 대한 분석 및 해결 방법

실습 과제 05: 강의자료 4장 5절의 프로그램 구현하기

이번 과제는 수업 시간에서 배운 다양한 변환 행렬들을 직접 도형에 적용해보고, 이를 응용하여 복합 변환을 구현하는 것으로 파악했다. 따라서 행렬을 처리하기 위한 자료형과, 변환 함수들을 정의하고 이를 이용하여 도형을 변환하는 로직을 각각의 변환과 변환 순서에 따라 구현하였다. 또한 자신만의 복합 변환으로 임의의 축을 랜덤 생성하고, 이 축에 따라 도형을 회전하는 변환을 구현하였다.

$$
\begin{aligned}
&Step\ 1\text{: } T \\
&Step\ 2\text{: } R_y(\beta) \cdot R_x(\alpha) \\
&Step\ 3\text{: } R_z(\theta) \\
&Step\ 4\text{: } R_x^{-1}(\alpha) \cdot R_y^{-1}(\beta) \\
&Step\ 5\text{: } T^{-1}
\end{aligned}
$$

(T=이동 변환, R=축 회전 변환)

$$R(\theta) = T^{-1} \cdot R_x^{-1}(\alpha) \cdot R_y^{-1}(\beta) \cdot R_z(\theta) \cdot R_y(\beta) \cdot R_x(\alpha) \cdot T$$

(임의의 축 회전 변환 과정의 예시)

2. 자신이 구현한 주요 코드

실습 과제 05: 강의자료 4장 5절의 프로그램 구현하기

```
} else if (key == MENU_COMPLEX_ROTATE) {
    // 복합 변환 - 회전
    glkMatIdentity(m1);
    // 단위 행렬로 m1 초기화

    glkMatTrans(m2, -P[0], -P[1], -P[2]);
    glkMatMult(m2, m1, m1);
    // 원점으로 이동 변환

    static double angle = 0.0;
    glkMatRotateZ(m2, angle += isCalledByMenu ? 50 : 10);
    glkMatMult(m2, m1, m1);
    // z 축 회전 변환

    glkMatTrans(m2, P[0], P[1], P[2]);
    glkMatMult(m2, m1, m1);
    // 원래 지점으로 이동 변환

    printf("Complex rotate: \n");
    glkMatPrint(m1);
    transformTriangle(m1, P, Q);
}
```

회전 복합 변환

```
} else if (key == MENU_COMPLEX_SCALE) {
    // 복합 변환 - 신축
    glkMatIdentity(m1);
    // 단위 행렬로 m1 초기화
    glkMatTrans(m2, -P[0], -P[1], -P[2]);
    glkMatMult(m2, m1, m1);
    // 원점으로 이동 변환

    static double scale = 0.0;
```

```
    scale += isCalledByMenu ? 0.5 : 0.1;
    scale = scale < 1.5 ? scale : 0.1;
    glkMatScale(m2, scale * 1.2, scale, scale);
    glkMatMult(m2, m1, m1);
    // 신축 변환

    glkMatTrans(m2, P[0], P[1], P[2]);
    glkMatMult(m2, m1, m1);
    // 원래 지점으로 이동 변환

    printf("Complex scale: \n");
    glkMatPrint(m1);
    transformTriangle(m1, P, Q);
}
```

신축 복합 변환

```
} else if (key == MENU_COMPLEX_SHEERING) {
    // 복합 변환 - 밀림
    glkMatIdentity(m1);
    // 단위 행렬로 m1 초기화
    glkMatTrans(m2, -P[0], -P[1], -P[2]);
    glkMatMult(m2, m1, m1);
    // 원점으로 이동 변환

    static double sheer = 0.0;
    sheer += isCalledByMenu ? 0.5 : 0.1;
    sheer = sheer < 1.5 ? sheer : 0.1;
    glkMatShearX(m2, sheer, 1);
    glkMatMult(m2, m1, m1);
    // 전단(밀림) 변환

    glkMatTrans(m2, P[0], P[1], P[2]);
    glkMatMult(m2, m1, m1);
    // 원래 지점으로 이동 변환

    printf("Complex sheering: \n");
    glkMatPrint(m1);
    transformTriangle(m1, P, Q);
}
```

밀림 복합 변환

```
} else if (key == MENU_ROTATE_ARBITRARY_AXIS) {
    for (int i = 0; i < 4; i++) {
        positionL[i] = L[i % 4 + 4] - L[i];
    }
    // 직선 벡터 계산

    double yAngle = -(atan2(positionL[2], positionL[0]) / M_PI * 180);
    // 직선을 x 축으로 맞추기 위해 필요한 y 축 방향 회전각
    glkMatRotateY(m2, yAngle);
    // yAngle 회전 행렬

    glkTransform(m2, positionL, yRotatedL);
    // 직선 벡터를 yAngle 만큼 회전하여 Y 축 회전된 AB 벡터 계산

    double zAngle = -(atan2(yRotatedL[1], yRotatedL[0]) / M_PI * 180);
    // 직선을 x 축으로 맞추기 위해 필요한 z 축 방향 회전각

    glkMatIdentity(m1);
    // 단위 행렬로 m1 초기화

    glkMatTrans(m2, -L[0], -L[1], -L[2]);
    glkMatMult(m2, m1, m1);
    // 직선을 원점으로 이동

    glkMatRotateY(m2, yAngle);
    glkMatMult(m2, m1, m1);
    // y 축 회전 변환

    glkMatRotateZ(m2, zAngle);
    glkMatMult(m2, m1, m1);
    // z 축 회전 변환

    static double angle = 0.0;
    glkMatRotateX(m2, angle += isCalledByMenu ? 15 : 3);
    glkMatMult(m2, m1, m1);
    // x 축 회전 변환 (사용자 입력 각도 회전)

    glkMatRotateZ(m2, -zAngle);
    glkMatMult(m2, m1, m1);
```

```
    // z 축 회전 역변환

    glkMatRotateY(m2, -yAngle);
    glkMatMult(m2, m1, m1);
    // y 축 회전 역변환

    glkMatTrans(m2, L[0], L[1], L[2]);
    glkMatMult(m2, m1, m1);
    // 직선을 원래 위치로 이동

    printf("Rotate arbitrary axis: \n");
    glkMatPrint(m1);
    transformTriangle(m1, P, Q);
}
```

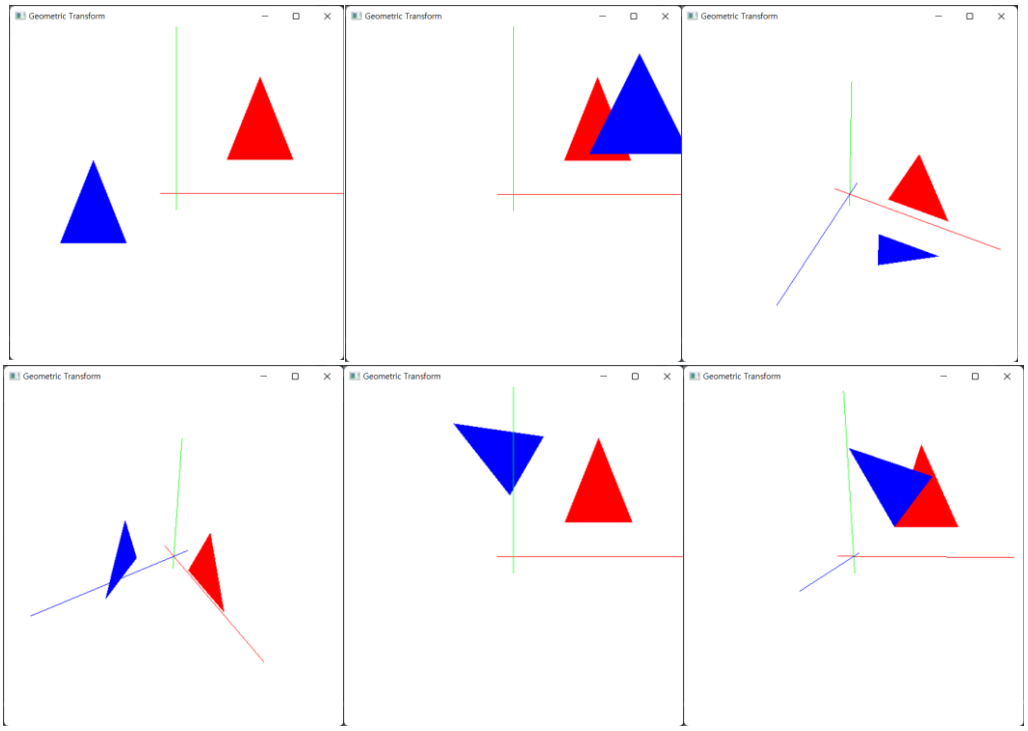
임의의 축 회전 복합 변환

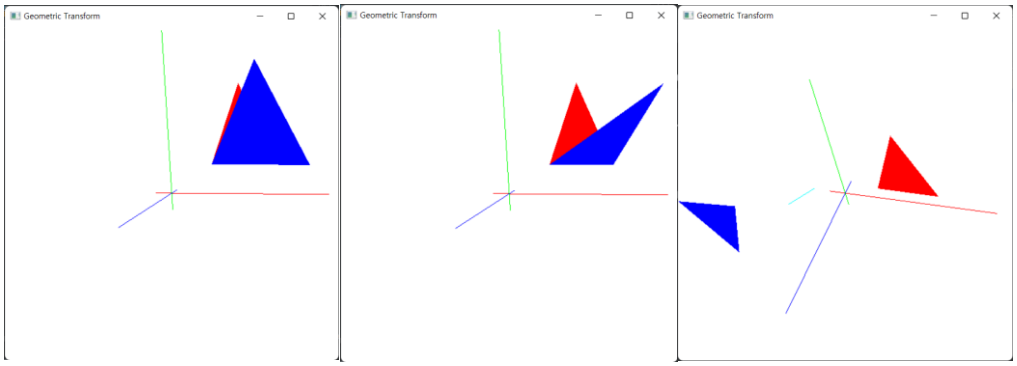
3. 테스트 결과

실습 과제 05: 강의자료 4장 5절의 프로그램 구현하기

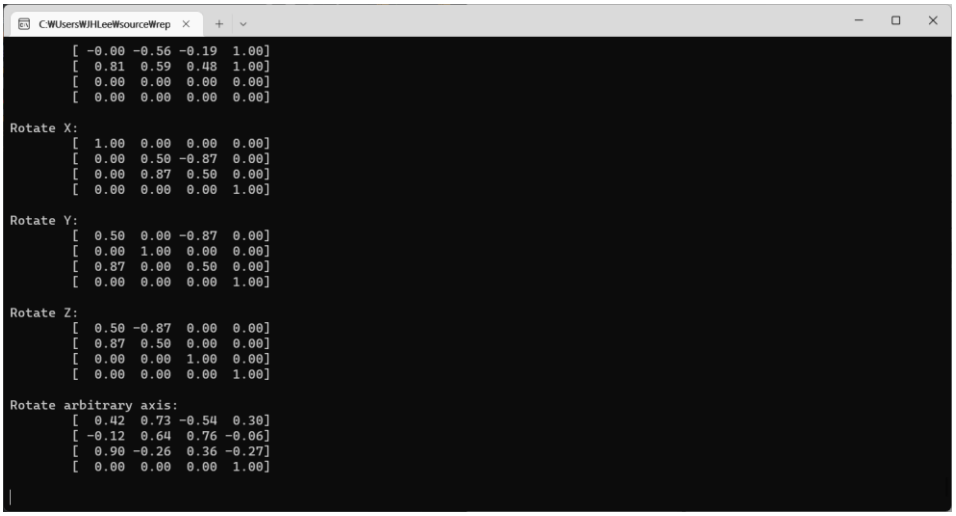
```
Init view (i)
Transform (t)
Scale (s)
Rotate X (x)
Rotate Y (y)
Rotate Z (z)
[Complex transform] Rotate (Z)
[Complex transform] Scale (c)
[Complex transform] Sheering (h)
[Own complex transform] Rotate arbitrary axis (r)
[Own complex transform] Set random axis (a)
Exit
```

*. Complex transform의 경우 해당 단축키를 계속 누르는 경우 변화량(각도, 좌표)이 계속 증가하도록 구현하였음





("Transform (t)" ~ "Rotate arbitrary axis (r)" 까지 실행 결과)



(모델뷰 행렬 출력 결과)

4. 느낀 점

과제에서 제시된 변환들을 구현하는 데에는 시간을 많이 사용하지 않았다. 그런데 임의의 축 변환을 구현하는데 각도 계산 코드 두 줄에서 실수를 하는 바람에, 이상하게 작동하는 모습이 관찰되었고 다른 변환을 구현하는 시간의 두배 이상의 시간을 들여서 이를 해결할 수 있었다. 완성 코드를 만들기 전에 다소 코드가 더럽더라도 빠르게 프로토타이핑해서 작동을 확인한 후 리팩토링을 하는 스타일이었는데, 행렬 또는 복잡한 수학 연산을 다루는 프로그램에서는 처음부터 가독성과 구조를 신경써서 코드를 작성하는 것이 중요하다는 것을 느낄 수 있었다.