목차

1. 문제에 대한 분석 및 해결 방법

실습 과제 07: 로봇에 동작 넣기 (과제07) + 뷰포트 나누기 및 원근투상 적용

이번 과제의 핵심은 로봇의 각 부품에 변환 행렬을 고려하여 절차적으로 애니메이션을 적용하고 재생시키는 것과, OpenGL에서 제공하는 뷰포트, 투상 제어 함수 등을 이용하여 뷰포트를 나누고 원근투상을 적용하는 것으로 파악했다. 따라서 각 부품의 애니메이션 로직을 가지고 있는 애니메이션 클래스를 모델링하고, 기존의 로봇 코드에서 로봇을 그리기 전에 애니메이션 클래스의 객체를 이용하여 애니메이션을 실행시키게끔 처리하였다. 또한 glOrtho, glFrustum, glViewport 등의 함수를 이용하여 뷰포트를 나누고 원근투상을 적용한 렌더링 결과를 얻을 수 있었다.

2. 자신이 구현한 주요 코드

실습 과제 07: 로봇에 동작 넣기 (과제07) + 뷰포트 나누기 및 원근투상 적용

```
class Animation {
public:
    unsigned int tick = 0;

    virtual void baseTransform() = 0;
    virtual void animateHead() = 0;
    virtual void animateBody() = 0;
    virtual void animateLeftArm() = 0;
    virtual void animateLeftHand() = 0;
    virtual void animateRightArm() = 0;
    virtual void animateRightHand() = 0;
    virtual void animateLeftLeg() = 0;
    virtual void animateLeftFoot() = 0;
    virtual void animateRightLeg() = 0;
    virtual void animateRightFoot() = 0;
};
```

<애니메이션 클래스 정의>

```
void draw() {
   animation->baseTransform();
   glPushMatrix();
      animation->animateBody();
      Body.draw(TO_COLOR_F(0x88C8C8), scale, bCoord);

      glPushMatrix();
         glTranslated(TO_SCALED_F(0, 28, 0, scale));
         animation->animateHead();
         Head.draw(TO_COLOR_F(0x01ADF0), scale);
      glPopMatrix();

      glPushMatrix();
         glTranslated(TO_SCALED_F(16, 8, 0, scale));
         animation->animateLeftArm();
         LeftArm.draw(TO_COLOR_F(0xA8A7AC), scale);
            glPushMatrix();
               glTranslated(TO_SCALED_F(0, -26, 0, scale));
               animation->animateLeftHand();
               LeftHand.draw(TO_COLOR_F(0x0D457A), scale);
            glPopMatrix();
      glPopMatrix();

      glPushMatrix();
         glTranslated(TO_SCALED_F(5, -21, 0, scale));
         animation->animateLeftLeg();
         LeftLeg.draw(TO_COLOR_F(0xA8A7AC), scale);
            glPushMatrix();
```

```
            glTranslated(TO_SCALED_F(0, -19, 0, scale));
            animation->animateLeftFoot();
            LeftFoot.draw(TO_COLOR_F(0x0D457A), scale);
        glPopMatrix();
    glPopMatrix();

    glPushMatrix();
       glTranslated(TO_SCALED_F(-16, 8, 0, scale));
       animation->animateRightArm();
       RightArm.draw(TO_COLOR_F(0xA8A7AC), scale);
          glPushMatrix();
             glTranslated(TO_SCALED_F(0, -26, 0, scale));
             animation->animateRightHand();
             RightHand.draw(TO_COLOR_F(0x0D457A), scale);
          glPopMatrix();
    glPopMatrix();

    glPushMatrix();
       glTranslated(TO_SCALED_F(-5, -21, 0, scale));
       animation->animateRightLeg();
       RightLeg.draw(TO_COLOR_F(0xA8A7AC), scale);
          glPushMatrix();
             glTranslated(TO_SCALED_F(0, -19, 0, scale));
             animation->animateRightFoot();
             RightFoot.draw(TO_COLOR_F(0x0D457A), scale);
          glPopMatrix();
    glPopMatrix();
  glPopMatrix();
}
```

<애니메이션 클래스를 이용한 로봇 애니메이션 적용>

```
/**
 * 4분할 뷰 렌더링 함수
 */
void renderView(int x, int y, int width, int height,
                double upx, double upy, double upz,
                double eyex, double eyey, double eyez, bool useOrtho = true) {
    glViewport(x, y, width, height);
    glPushMatrix();
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    if (useOrtho) {
        glOrtho(
                -1, 1, -1, 1,
                0, 10
        );
    } else {
        glFrustum(
                -1, 1, -1, 1,
                0, 10
        );
    }

    glMatrixMode(GL_MODELVIEW);
    gluLookAt(
            eyex, eyey, eyez,
            0, 0, 0,
            upx, upy, upz
    );

    handleViewLogic();
    glTranslated(ModelX, 0, ModelY);
    glRotated(RotX, 0, -1, 0);
    glRotated(RotY, 1, 0, 0);
    glScalef(modelZoom, modelZoom, modelZoom);

    robot.draw();
    glPopMatrix();
}
```

<원근투상 방법 등을 이용하여 4분할 뷰를 그리는 함수>

```
void display() {
    glClearColor(0.85, 1.0, 1.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
```

```
    int width = glutGet(GLUT_WINDOW_WIDTH);
    int height = glutGet(GLUT_WINDOW_HEIGHT);

    renderView(LEFT_UP_VIEW(width, height), Z_AXIS_VECTOR, Y_AXIS_EYE);
    renderView(LEFT_DOWN_VIEW(width, height), Y_AXIS_VECTOR, Z_AXIS_EYE);
    renderView(RIGHT_UP_VIEW(width, height), Y_AXIS_VECTOR, QUARTER_VIEW_EYE);
    renderView(RIGHT_DOWN_VIEW(width, height), Y_AXIS_VECTOR, X_AXIS_EYE);
    // 4 개의 뷰 그리기


    glutSwapBuffers();
    glFlush();
}
```

<렌더링 함수>

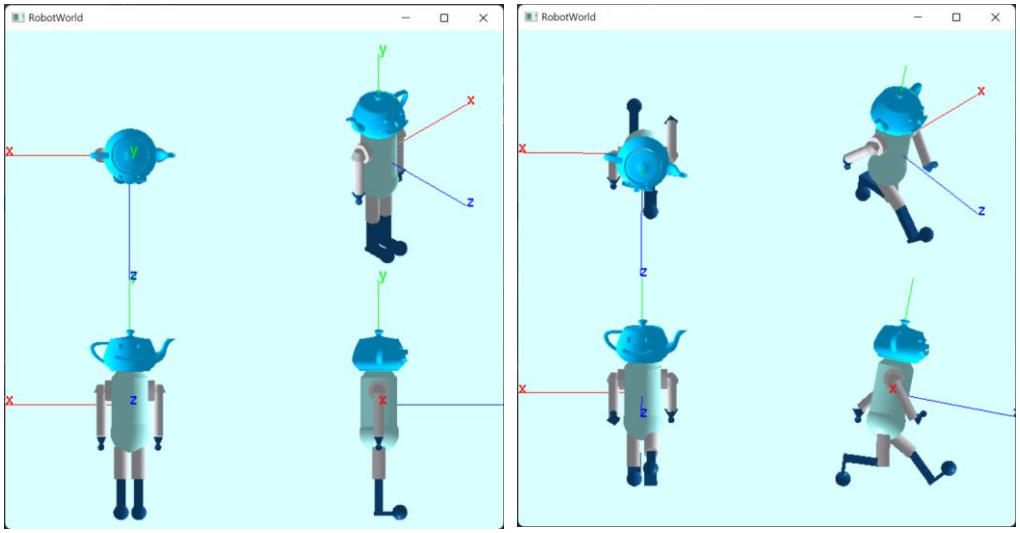
```
/**
 * 각기 다른 뷰 로직 처리 함수
 */
void handleViewLogic() {
    int width = glutGet(GLUT_WINDOW_WIDTH);
    int height = glutGet(GLUT_WINDOW_HEIGHT);

    Quadrant mouseQuadrant = getMouseQuadrant(PrevX - width / 2.0, PrevY - height /
2.0);
    if (previousQuadrant != mouseQuadrant) {
        DeltaX = DeltaY = 0;
        previousQuadrant = mouseQuadrant;
    }

    // 마우스의 사분면 뷰에 대한 기능 처리
    switch (mouseQuadrant) {
        case LEFT_UP_QUADRANT:
            ModelX = DeltaX / 100.0;
            ModelX = LIMIT_XY_LOCATION < ModelX ? LIMIT_XY_LOCATION : ModelX;
            ModelX = -LIMIT_XY_LOCATION > ModelX ? -LIMIT_XY_LOCATION : ModelX;
            ModelY = DeltaY / 100.0;
            ModelY = LIMIT_XY_LOCATION < ModelY ? LIMIT_XY_LOCATION : ModelY;
            ModelY = -LIMIT_XY_LOCATION > ModelY ? -LIMIT_XY_LOCATION : ModelY;
            break;
        case LEFT_DOWN_QUADRANT:
            RotX = DeltaX;
            RotY = DeltaY;
            break;
        case RIGHT_UP_QUADRANT:
            if (InstDeltaX < 0) {
                modelZoom += MAX_MODEL_ZOOM >= modelZoom ? 0.025 : 0;
            } else if (InstDeltaX > 0) {
                modelZoom += MIN_MODEL_ZOOM <= modelZoom ? -0.025 : 0;
            }
            break;
        case RIGHT_DOWN_QUADRANT:
            if (InstDeltaX > 0) {
                robot.scale += MIN_MODEL_SCALE >= robot.scale ? 0.75 : 0;
            } else if (InstDeltaX < 0) {
                robot.scale += MAX_MODEL_SCALE <= robot.scale ? -0.75 : 0;
            }
            break;
    }
}
```

<각 뷰마다 마우스 로직 처리>

3. 테스트 결과

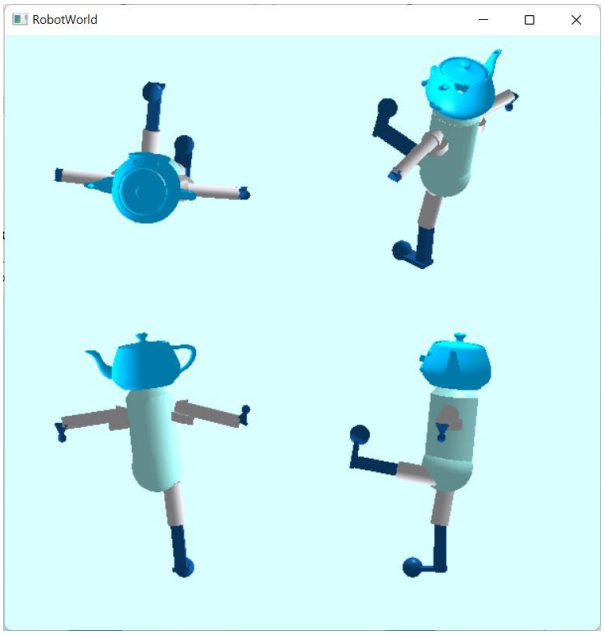
실습 과제 07: 로봇에 동작 넣기 (과제07) + 뷰포트 나누기 및 원근투상 적용

```
Toggle axis (c)
Init view (i)
Stop animation (s)
Running animation (r)
Fly kick Animation (f)
Animation speed down (z)
Animation speed up (x)
Exit (q)
```

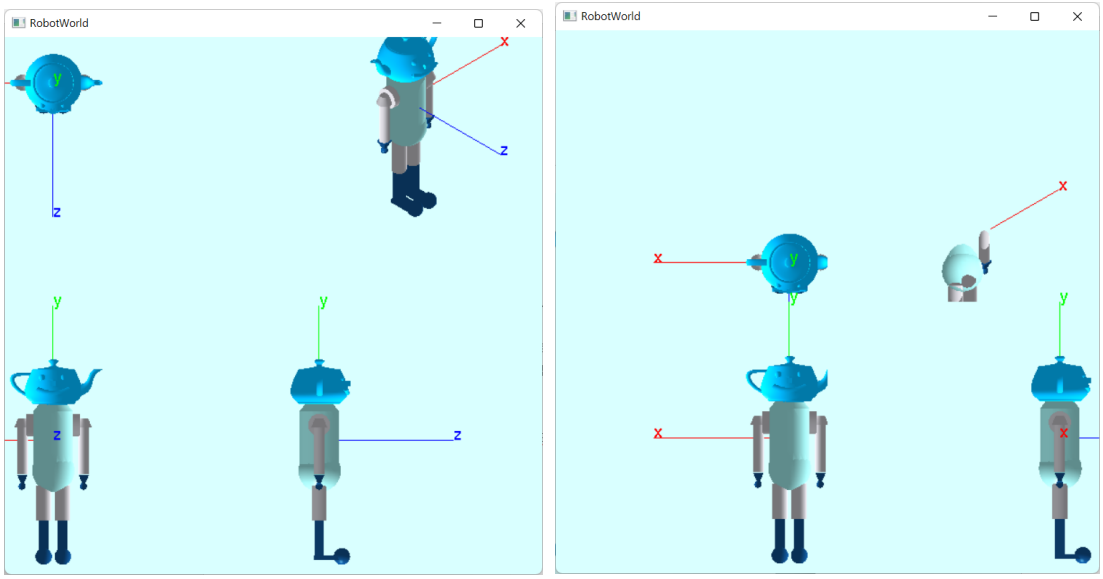


(좌: 실행 결과 (멈춤 상태) / 우: 달리기 애니메이션)

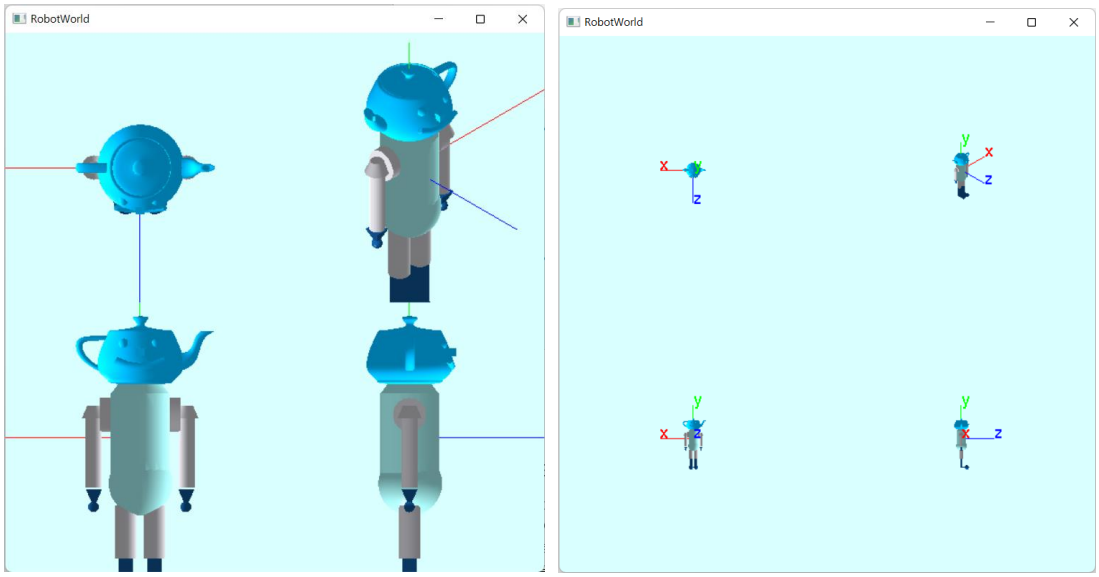
자신만의 재미있는 동작 추가하기 + 마우스를 이용한 화면 조작



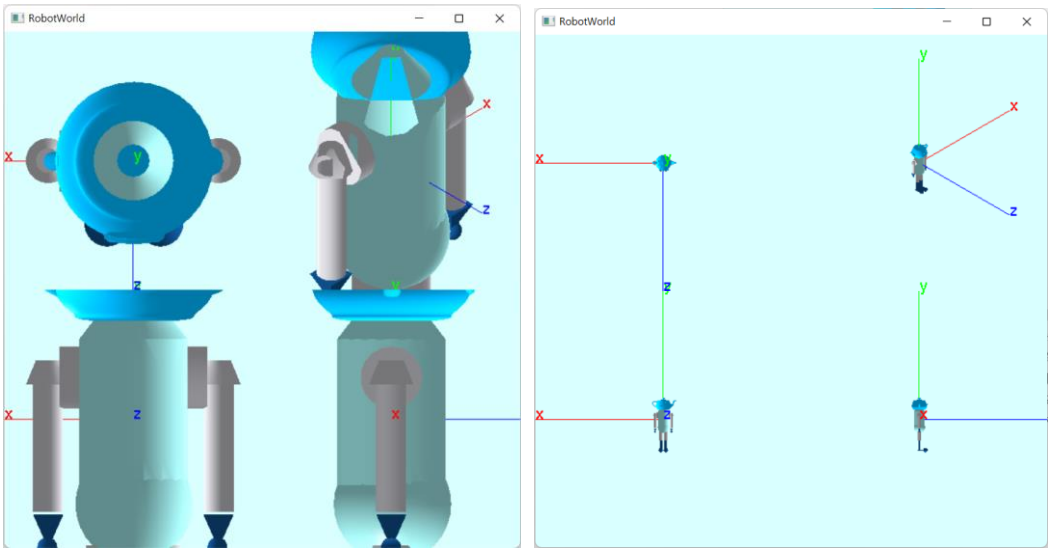
1080도 회전 발차기 동작 추가



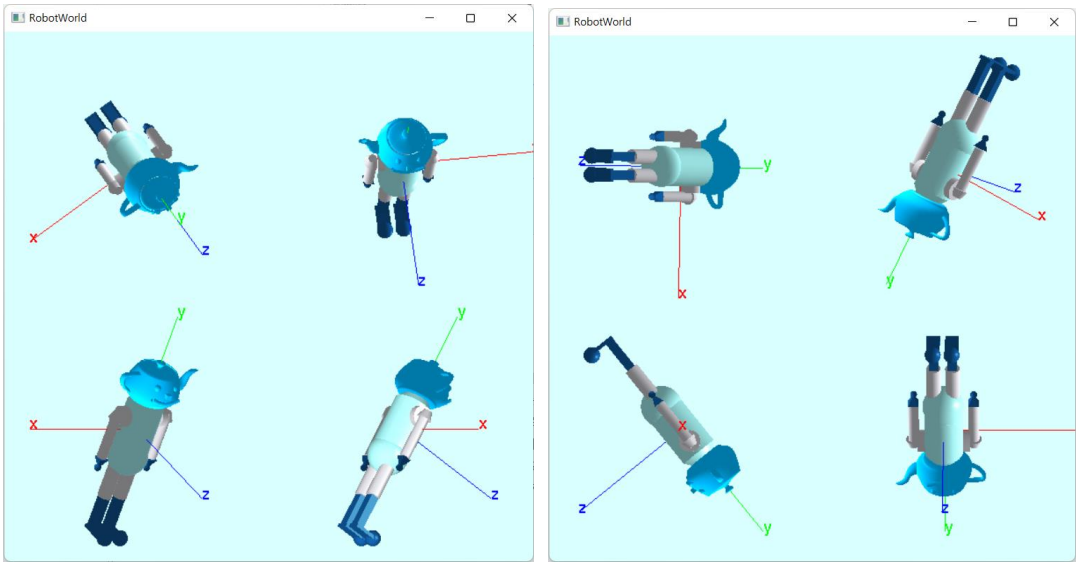
평면도: X-Z 평면에서의 이동



원근 투상: Zoom in/out



측면도: 신축 (로봇의 크기 변경)



정면도: 회전 (x축, y축 중심)

4. 느낀 점

로봇을 조립할 때에는 몰랐는데, 각 부품을 축 회전시키니까 축 중심이 맞지 않아서 모델을 3D Max에서 불러와서 모델 좌표계에서의 위치를 수정해줘야 하는 문제가 있어서 시간을 많이 소비했다. 또 자신만의 동작(1080도 날라차기)을 만드는 게 생각보다 쉽지 않았고, 우리가 영화나 게임 등에서 나오는 3D 그래픽 캐릭터의 자연스러운 동작을 만드는 것이 모든 부품의 동작에 세세하게 신경을 써 줘야 하는 일임을 몸소 체험해볼 수 있었다. (모션 캡처 기술이 나오기 전까지) 이렇게 로봇에 애니메이션까지 넣어보니 내가 만든 로봇이 살아 움직이는 것 같아서 엄청 뿌듯함을 느낄 수 있었던 과제였다.