



R E P O R T

컴퓨터그래픽스및실습 실습과제 07_08

과목명	컴퓨터그래픽스및실습
분반	01
교수	최 영 규
학번	2020136129
이름	최 수 연
제출일	2022년 11월 7일 월요일

[문제 분석 및 해결 방법]

[문제 분석]

- 로봇에 달리기 동작 넣기 (과제7)
- 자신만의 재미있는 동작 추가하기 (과제7)
- 뷰포트 나누기 및 원근 투상 적용 (과제8)
- 마우스를 이용한 화면 조작 (과제8)
 - 마우스: 왼쪽 버튼을 누른 채 이동 → 기능
 - 평면도: X-Z 평면에서의 이동
 - 정면도: 회전 (예: y축, x축 중심)
 - 측면도: 신축 (로봇의 크기 변경)
 - 원근 투상: Zoom in/out

[해결 방법]

- 본 과제7, 과제8은 이전 과제6의 코드를 사용하여 코드를 추가하는 방식으로 구현된다.
- (과제7) 로봇에 달리기 동작을 넣으려면 Robot.h 파일에 애니메이션 모델 변환에 대한 코드를 넣어야 한다. 그리고 RunningRobot.cpp 파일에서 Robot 클래스의 객체를 생성하여, 키보드를 통해 로봇이 달렸다 멈추는 동작을 조작할 수 있도록 구현한다.
- (과제7 추가 기능) 자신만의 재미있는 동작 추가하기 또한 위와 같은 방식으로 구현하면 되는데, 이때 키보드 작동 시 위의 달리기 동작과 겹치거나 충돌하여 화면에 제대로 구현되지 않는 것을 처리하는 코드도 추가해야 한다.
- (과제8) 뷰포트 나누기는 RunningRobot.cpp 파일에서 glViewport() 함수를 통해 총 4개의 구역으로 나눈다. 또한 뷰마다 시점을 다르게 하여 정면도, 평면도, 측면도를 구현한다.
- (과제8) 나머지 하나의 원근투상 적용을 위한 뷰는 glMatrixMode() 함수에서 GL_PROJECTION 행렬 모드를 사용하여 glFrustum() 함수를 통해 원근 투상을 적용하여 뷰를 구현한다.
- (과제8 추가 기능) 마우스를 이용한 화면 조작에서 정면도 회전의 경우에는 정면도에 해당하는 뷰포트 부분에 gluLookAt() 함수를 사용하여 기존의 mouseClicked()과 mouseMotion() 함수에서 저장되는 실시간 RotX, RotY 값을 통해 해당 뷰의 모델이 마우스 움직임에 따라 회전할 수 있도록 구현한다.

[주요 설명 코드] - 과제 7

<RunningRobot.cpp>

```
void display() {
    glClearColor(0.99, 0.97, 0.97, 1.0); // 배경 색상 설정
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // 버퍼 초기화
    glMatrixMode(GL_MODELVIEW); // 시점 변환
    glLoadIdentity(); // 모델뷰 행렬을 항등행렬로 바꿈
    gluLookAt(0, 0, 0, -SIN(RotY), -0.01 * RotX, -COS(RotY), 0, 1, 0); // 시점 좌표계 정의 함수
    robot.draw();
    glutSwapBuffers();
    glFlush();
} //모델 뷰 행렬로 화면 구현하고, 회전이 가능하도록 RotX, RotY를 사용하여 gluLookAt 함수를 구현했다.

void timerCallback(int tId) {
    if (bCamRot && tId == 1) {
        RotY += 5;
        if (RotY > 360) RotY = 0;
        glutTimerFunc(50, timerCallback, 1);
    }
    if (bRobotRun && tId == 2) {
        robot.run();
        glutTimerFunc((int)(tSpeed), timerCallback, 2);
    }
    if (bColdRun && tId == 3) {
        robot.coldRun();
        glutTimerFunc((int)(tSpeed), timerCallback, 3);
    }
    glutPostRedisplay();
} // 타이머 콜백 함수로, 아래 키보드 함수에서 특정 키보드를 누르면, 그에 해당하는 tId 번호가 해당 함수
의 인자로 전달되어, 각각의 기능을 실행한다. 이때 로봇이 달리는 동작과 추위에 떠는 동작 두 개 모두
Robot 클래스 객체를 생성하여 해당 클래스의 함수를 실행하도록 한다.

void keyboard(unsigned char key, int x, int y) {
    if (key == 'i') { // 단위행렬로 초기화
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        RotX = RotY = 0.0f;
    }
    else if (key == 'v') {
        bCamRot = !bCamRot;
        glutTimerFunc(100, timerCallback, 1);
    }
    else if (key == 'r') {
        bRobotRun = !bRobotRun; // Toggle Run
    }
}
```

```

        if (bRobotRun == true && bColdRun == true)
            bColdRun = false;
        if (bRobotRun)
            glutTimerFunc(40, timerCallback, 2); // 타이머 시작
        else robot.stop();
    }
    else if (key == 'c') {
        bColdRun = !bColdRun; // Toggle Run
        if (bRobotRun == true && bColdRun == true)
            bRobotRun = false;
        if (bColdRun)
            glutTimerFunc(40, timerCallback, 3); // 타이머 시작
        else robot.stop();
    }
    .....
    glutPostRedisplay(); // 화면에 모델을 다시 그리도록 요청
} // 키보드 함수로 i키를 누르면 단위 행렬로 초기화되고, v키를 누르면 모델이 y축을 중심으로 회전하며, r
키를 누르면 모델이 달리고, c키를 누르면 모델이 추위에 떠는 동작을 실행한다.

```

<Robot.h>

```

void draw() {
    glPushMatrix();
    bRun ? animateBody() : coldBody();
    Body.draw(0.5, 0.8, 0.8, scale, true); // 몸통 그리기

    glPushMatrix(); // 몸통 좌표계 저장
    glTranslated(0.0, 0.45, -0.07);
    glScalef(1.1f, 1.1f, 1.1f);
    bRun ? animateHead() : coldHead();
    Head.draw(0.8, 0.7, 0.7, scale); // 머리 그리기
    glPopMatrix(); // 몸통 좌표계로 돌아가기
    .....
} // 삼항연산자를 사용하여 bRun이 참이면 달리는 애니메이션에 해당하는 함수를 모두 실행하고, bRun이
거짓이면, 아래의 run()과 coldRun() 함수를 통해 bCold가 참이 되므로, 추위에 떠는 애니메이션에 해당하는
함수를 모두 실행한다.

```

```

double tAngle, sin_t = 0, sin_t2 = 0, cos_t = 0;
bool bRun, bCold;
void reset() {
    bRun = false;
    bCold = false;
    scale = 100;
    tAngle = 0;
} // 변수들을 리셋해주는 함수이다.
void stop() { bRun = false; bCold = false; } // 동작을 멈추는 함수이다.

```

```

void run() { // 달리는 동작을 실행하도록 하는 함수이다.
    bRun = true;
    bCold = false;
    tAngle += 20;
    if (tAngle >= 360)
        tAngle -= 360;
    sin_t = SIN(tAngle);
    cos_t = COS(tAngle);
    sin_t2 = SIN(2 * tAngle);
}

void coldRun() { // 추위에 떠는 동작을 실행하도록 하는 함수이다.
    bRun = false;
    bCold = true;
    tAngle += 20;
    if (tAngle >= 360)
        tAngle -= 360;
    sin_t = SIN(tAngle);
    cos_t = COS(tAngle);
    sin_t2 = SIN(2 * tAngle);
}

```

// 각 부품에 대한 애니메이션 모델 변환: RUN

```

void animateBody() {
    glTranslated(0.0, (sin_t2 * 0.08), 0);
    glRotatef(-sin_t2 * 15 - 10, 1, 0, 0);
    glRotatef(-sin_t * 13, 0, 1, 0);
}

void animateHead() {
    glRotatef(-cos_t * 20, 0, 1, 0);
}

void animateRightArm() {
    glRotatef(-cos_t * 65, 1, 0, 0);
}

void animateRightHand() {
    glRotatef(-cos_t * 45 + 70, 1, 0, 0);
    glRotatef(-sin_t * 20 - 20, 0, 1, 0);
}

void animateLeftArm() {
    glRotatef(cos_t * 65, 1, 0, 0);
}

void animateLeftHand() {
    glRotatef(cos_t * 45 + 70, 1, 0, 0);
    glRotatef(sin_t * 20 + 20, 0, 1, 0);
}

```

// 각 부품에 대한 애니메이션 모델 변환: COLD

```

void coldBody() {
    glTranslated(0.0, sin_t2 * 0.02, 0);
    glRotatef(-sin_t * 13, 0, 1, 0);
}

void coldHead() {
    glRotatef(-cos_t * 20, 0, 1, 0);
}

void coldRightArm() {
    glRotatef(-sin_t2 * 5, 0, 0, 1);
}

void coldRightHand() {
    glRotatef(sin_t2 * 5 + 130, 1, 0, 0);
    glRotatef(-sin_t2 * 5, 0, 0, 1);
}

void coldLeftArm() {
    glRotatef(sin_t2 * 5, 0, 0, 1);
}

void coldLeftHand() {
    glRotatef(sin_t2 * 5 + 130, 1, 0, 0);
    glRotatef(sin_t2 * 5, 0, 0, 1);
}

void coldRightLeg() {

```

```
void animateRightLeg() {
    glRotatef(cos_t * 45 + 35, 1, 0, 0);
}
void animateRightFoot() {
    glRotatef(sin_t * 50 - 55, 1, 0, 0);
    glRotatef(tAngle / 20, 0, -1, 0);
}
void animateLeftLeg() {
    glRotatef(-cos_t * 45 + 35, 1, 0, 0);
}
void animateLeftFoot() {
    glRotatef(-sin_t * 50 - 55, 1, 0, 0);
    glRotatef(-1 * (tAngle / 20), 0, -1, 0);
}
```

```
glRotatef(cos_t * 25 + 15, 1, 0, 0);
}
void coldRightFoot() {
    glRotatef(-cos_t * 45 - 45, 1, 0, 0);
}
void coldLeftLeg() {
    glRotatef(-cos_t * 25 + 15, 1, 0, 0);
}
void coldLeftFoot() {
    glRotatef(cos_t * 45 - 45, 1, 0, 0);
}
```

[주요 설명 코드] - 과제 8

<RunningRobot.cpp>

```
void DrawScene() { // 모델의 바닥 표현
    glPushMatrix(); // 현재 좌표계 저장
    glTranslatef(0.0, -1.0, 0.0); // y축으로 -1만큼 이동, 로봇의 발밑에 사각형이 그려지도록 함
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL); // 앞뒷면 채움
    glBegin(GL_POLYGON); // 사각형 그리기 시작
    glVertex3f(0.6, 0.0, 0.6);
    glVertex3f(0.6, 0.0, -0.6);
    glVertex3f(-0.6, 0.0, -0.6);
    glVertex3f(-0.6, 0.0, 0.6);
    glEnd(); // 사각형 그리기 종료
    glPopMatrix();
} // 모델의 바닥을 표현하기 위해 모델 발밑 부분에 사각형이 그려지도록 하였다.

void display() {
    glClearColor(0.99, 0.97, 0.97, 1.0); // 배경 색상 설정
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // 버퍼 초기화

    // 왼쪽 아래 화면, 정면도
    glViewport(-20, 10, 350, 350);
    glMatrixMode(GL_MODELVIEW); // 시점 변환
    glPushMatrix();
    glLoadIdentity(); // 모델뷰 행렬을 항등행렬로 바꿈
    gluLookAt(0, 0, 0, -SIN(RotY), -0.01 * RotX, -COS(RotY), 0, 1, 0); // 시점 좌표계 정의 함수
    // gluLookAt 함수를 사용하여 정면도가 마우스 움직임에 따라 회전될 수 있도록 하였다.
    DrawScene();
    robot.draw();
    glPopMatrix();

    // 오른쪽 아래 화면, 측면도
    glViewport(330, 10, 350, 350);
    glPushMatrix();
    gluLookAt(0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
    DrawScene();
    robot.draw();
    glPopMatrix();

    // 왼쪽 위 화면, 평면도
    glViewport(-20, 360, 350, 350);
    glPushMatrix();
    gluLookAt(0.0, -0.1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.0);
    DrawScene();
}
```

```

robot.draw();
glPopMatrix();

// 오른쪽 위 화면, 측면도(glFrustum() 사용하여 원근투상 표현)
glViewport(330, 360, 350, 350);
glMatrixMode(GL_PROJECTION);
glPushMatrix();
glLoadIdentity();
glFrustum(-1.0f, 1.0f, -1.0f, 1.0f, 8.0f, 30.0f);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glPushMatrix();
gluLookAt(-5.0, 5.0, -5.0, 0.1, -0.1, 0.0, 0.0, 1.0, 0.0);
DrawScene();
robot.draw();
glPopMatrix();
glMatrixMode(GL_PROJECTION);
glPopMatrix();

glMatrixMode(GL_MODELVIEW); // 시점 변환
glLoadIdentity(); // 모델뷰 행렬을 항등행렬로 바꿈
//gluLookAt(0, 0, 0, -SIN(RotY), -0.01 * RotX, -COS(RotY), 0, 1, 0); // 시점 좌표계 정의 함수

glutSwapBuffers();
glFlush();

```

} // 뷰포트를 윈도우 크기에 맞추어 4분할한다. gluLookAt() 함수를 사용하여 각각의 시점을 정면도, 평면도, 측면도로 설정한다. 1사분면 화면은 glFrustum() 함수를 사용하여 원근 투상을 적용한다.

```

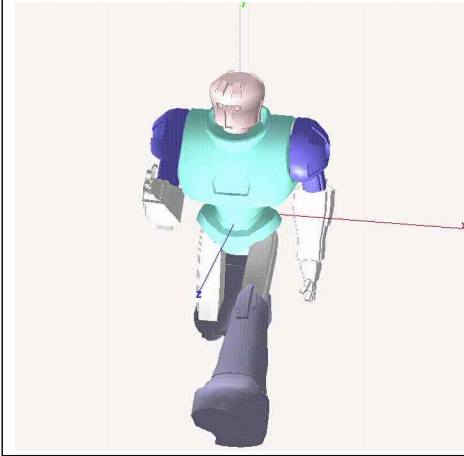
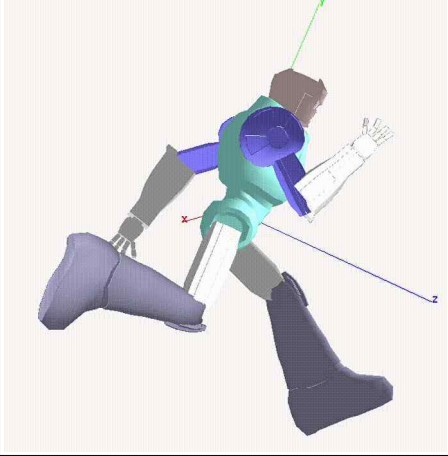
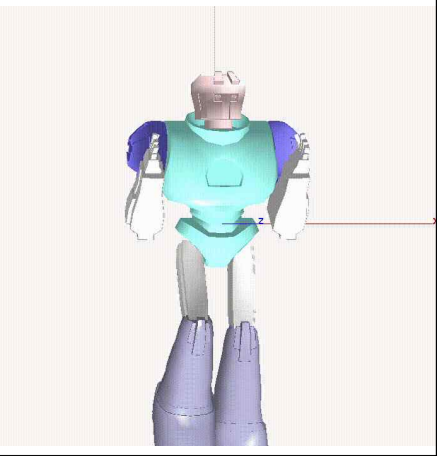
static int PrevX, PrevY;
void mouseClicked(int button, int state, int x, int y) {
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
        // 만약 마우스 왼쪽 버튼을 눌러 드래그하면
        PrevX = x; // 마우스의 x, y의 위치를 현재 위치로 변경
        PrevY = y;
    }
}

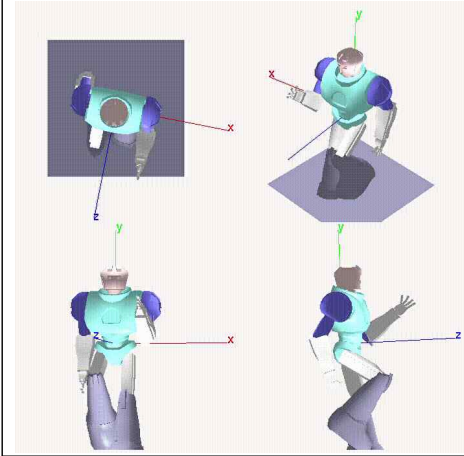
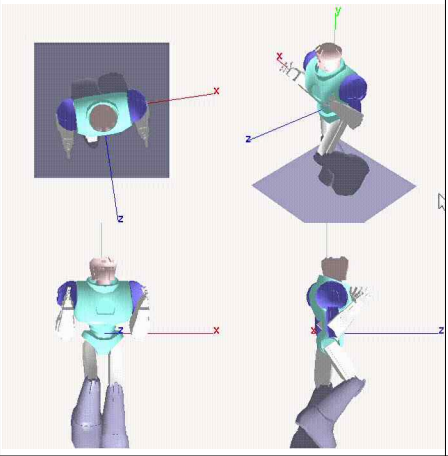
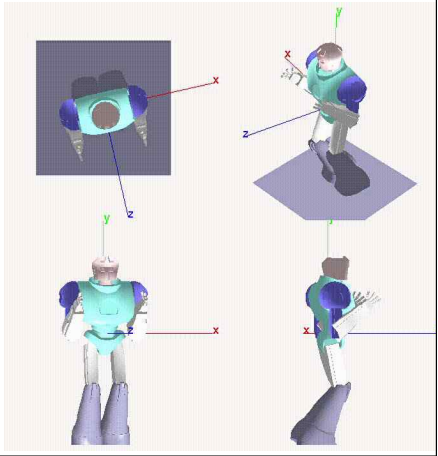
void mouseMotion(GLint x, GLint y) {
    RotX += (PrevY - y); // 각각 움직인 x, y만큼 회전되도록 설정
    RotY += (PrevX - x);
    PrevX = x;
    PrevY = y; // 다시 motion이 동작한만큼 PrevX와 PrevY에 각각 값을 저장
    glutPostRedisplay(); // 화면에 모델을 다시 그리도록 요청
}

```

} // 해당 마우스 클릭 함수와 마우스 모션 함수를 통해 실시간으로 변경되는 RotX, RotY를 사용하여 정면도 화면이 마우스에 따라 회전되도록 한다.

[테스트 결과]

		
달리는 모델 (r버튼)	달리면서 회전하는 모델 (v버튼)	추위에 떠는 모델 (c버튼)

		
달리는 모델 (r버튼)	추위에 떠는 모델 (c버튼)	마우스를 이용한 화면 조작 정면도: 회전

[고찰 및 느낀점]

이번 과제는 2주가 주어져서 생각보다 시간은 넉넉했지만, 생각보다 아쉬운 점이 많았던 과제이다. 먼저 나만의 동작을 만드는 과정에서 해당 동작은 손이 주먹 쥐어져야 더 자연스러운 동작인데, 모델의 손 자체가 펴져 있어서 주먹 쥐어지지 않는 점이 아쉬웠다. 그리고 8번 추가 문제를 모두 완성하지 못한 점도 너무 아쉬웠는데, 코드를 건드릴수록 더 이상해졌다. 뷰 별로 각각 마우스가 호환되는 부분을 제대로 해결하지 못했다. 마우스 콜백 함수와 `glMatrixMode()` 에 대해서 더 공부할 필요가 있는 것 같다.