컴퓨터그래픽스및실습

실습과제 07-08

과목명/분반	컴퓨터그래픽스및실습/01	담당교수	최영규 교수님
학 과 명	컴퓨터공학부	제 출 일	2023 / 11 / 12
학번	2019136098	이름	이우진

실습과제07

• 로봇에 동작 넣기
– 강의 동영상에서 설명한 로봇 달리기 동작 만들기
– 다음 동영상 참조
  • 그래픽스-06장-4절-1
  • 그래픽스-06장-4절-2
• 추가 기능(최대 +2점)
– 자신만의 재미있는 동작 추가하기

각 문제에 대한 분석과 자신이 해결한 방법 설명

모델 변환을 이용하여 애니메이션을 만든다. 로봇에는 몸통, 머리, 팔, 다리 등이 있는데 각 부분의 동작을 계산하여 각도를 이용하여 실감나게 로봇이 움직이게 하였다.

과제 6에서 사용했던 코드를 이용하여 애니메이션을 동작할 수 있게 Robot.h와 RunningRobot.cpp의 내용을 수정했다. Robot.h에서는 각 부분의 애니메이션 함수가 정상적으로 동작할 수 있게 draw함수 내에 각 함수를 실행 할 수 있는 코드를 추가했다.

RunningRobot.cpp에서는 우선 시점을 고정했고, timerCallback 함수를 만들어 로봇 부분이 정상적으로 움직일 수 있게 했다. keyboard 입력 부분은 일부 버튼을 추가했는데 v를 누르면 고정되있던 시점을 돌면서 로봇을 볼 수 있게 하였고, r을 누르면 로봇이 동작을 실행하게 하였다.

달리는 동작은 교수님께서 올려주신 강의영상을 보면서 코드를 구현하였으며 그 코드를 이용하여 자신만의 재미있는 동작을 추가했고, 그 동작은 걷는 동작을 추가했다. draw함수내에서 실행할 때 사용하는 코드는 우선 달리기 동작으로 입력이 되있으며 자신만의 재미있는 동작은 주석으로 처리했다.

자신이 구현한 주요 코드 및 코드 설명

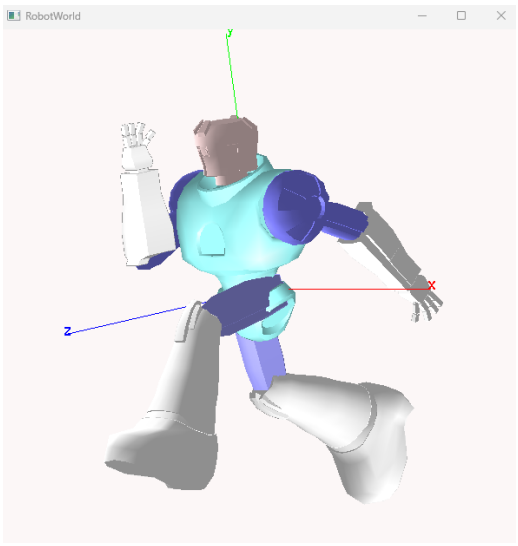
```
    // 애니메이션 코드
    double tAngle, sin_t = 0, sin_t2 = 0, cos_t = 0; // 애니메이션에 사용될 각도와 삼각함수 값들
    bool bRun; // 애니메이션이 실행 중인지를 나타내는 flag
    void reset() {
        bRun = false; // 애니메이션 중지
        scale = 100;
        tAngle = 0;
        // scale, tAngle 값 초기화
    } // 애니메이션 상태를 초기화하는 함수
    void stop() {
        bRun = false;
    } // 애니메이션을 중지하는 함수
    void run() {
        bRun = true; // 애니메이션 시작
        tAngle += 20;
        if (tAngle >= 360) {
            tAngle -= 360;
        } // 각도가 360 넘으면 360 빼기
```

```
        sin_t = SIN(tAngle);
        cos_t = COS(tAngle);
        sin_t2 = SIN(2 * tAngle);
        // 삼각함수 계산
    } // 애니메이션을 실행하는 함수

    // 각 부품에 대한 애니메이션 변환(모델 변환)
    void animateBody() {
        glTranslated(0.0, (sin_t2 * 0.08), 0); // 몸통의 위치 설정
        glRotatef(-sin_t2 * 15 - 10, 1, 0, 0); // 몸통의 상하회전 설정
        glRotatef(-sin_t * 13, 0, 1, 0); // 몸통의 좌우회전 설정
    } // 몸통 애니메이션
    void animateHead() {
        glRotatef(-cos_t * 20, 0, 1, 0); // 머리 좌우 회전
    } // 머리 애니메이션
    void animateRightArm() {
        glRotatef(-cos_t * 65, 1, 0, 0); // 오른팔 상하 회전
    } // 오른팔 애니메이션
    void animateRightHand() {
        glRotatef(-cos_t * 45 + 70, 1, 0, 0); // 오른손 상하 회전
    } // 오른손 애니메이션
    void animateLeftArm() {
        glRotatef(cos_t * 65, 1, 0, 0); // 왼팔 상하 회전
    } // 왼팔 애니메이션
    void animateLeftHand() {
        glRotatef(cos_t * 45 + 70, 1, 0, 0); // 왼손 상하 회전
    } // 왼손 애니메이션
    void animateRightLeg() {
        glRotatef(cos_t * 45 + 35, 1, 0, 0); // 오른쪽 다리 상하 회전
    } // 오른쪽 다리 애니메이션
    void animateRightFoot() {
        glRotatef(sin_t * 50 - 55, 1, 0, 0); // 오른발 상하 회전
    } // 오른발 애니메이션
    void animateLeftLeg() {
        glRotatef(-cos_t * 45 + 35, 1, 0, 0); // 왼쪽 다리 상하 회전
    } // 왼쪽 다리 애니메이션
    void animateLeftFoot() {
        glRotatef(-sin_t * 50 - 55, 1, 0, 0); // 왼발 상하 회전
    } // 왼발 애니메이션

    void animateRightArmForWalk() {
        glRotatef(sin_t * 45, 1, 0, 0); // 오른팔 상하 회전
    }
```

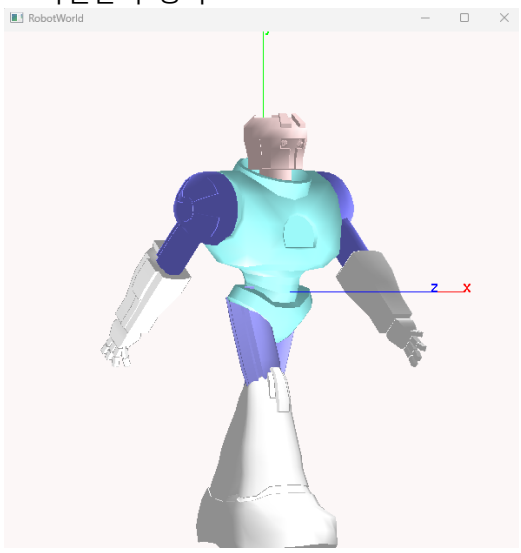
```
void animateLeftArmForWalk() {
    glRotatef(-sin_t * 45, 1, 0, 0); // 왼팔 상하 회전
}
void animateRightLegForWalk() {
    glRotatef(-sin_t * 20, 1, 0, 0); // 오른쪽 다리 약간 상하 회전
}
void animateLeftLegForWalk() {
    glRotatef(sin_t * 20, 1, 0, 0); // 왼쪽 다리 약간 상하 회전
}
void animateRightFootForWalk() {
    glRotatef(sin_t * 10 - 5, 1, 0, 0); // 오른발 상하 회전
}
void animateLeftFootForWalk() {
    glRotatef(-sin_t * 10 - 5, 1, 0, 0); // 왼발 상하 회전
}
```

다양한 입력에 대한 테스트 결과

<달리는 동작>



<자신만의 동작>



해당 문제에 대한 고찰

이번 과제를 통해 OpenGL을 이용하여 모델 변환을 통해 특정 부위를 독립적으로 움직이게 만드는 것을 이해하는 시간을 가졌다. 삼각함수를 이용하여 각 부분의 동작을 일일이 설정을 하여 해당 객체가 OpenGL에서 동작을 할 때 어떤 방식으로 구현이 되는지 알게된 시간이었다.

실습과제08

• 뷰포트 나누기 및 원근투상 적용

– 3D Max와 유사하게 화면을 4개의 뷰포트로 만들어라. 각 뷰포트에 자신이 만든 로봇에 대한 정면도/측면도/평면도를 그리고, 나머지 하나에는 그림과 같이 glFrustum()을 이용한 원근투상 결과를 그려라.

• 추가 기능(최대 +2점): 마우스를 이용한 화면 조작

– 마우스: 왼쪽 버튼을 누른채 이동 -> 기능

– 평면도: X-Z 평면에서의 이동
– 정면도: 회전 (예: y축, x축 중심)
– 측면도: 신축 (로봇의 크기 변경)
– 원근 투상: Zoom in/out

각 문제에 대한 분석과 자신이 해결한 방법 설명

7장 투상변환에 대한 내용을 적용해보는 과제였다. 해당 과제에서 사용하는 함수는 가시부피를 설정하는 함수인 OpenGL 함수 glOrtho를 사용하였으며 OpenGL 함수 중 원근투상 가시부피 설정 함수인 glFrustum을 사용하였다. 그리고 glViewport를 사용하여 뷰포트를 설정하였다.

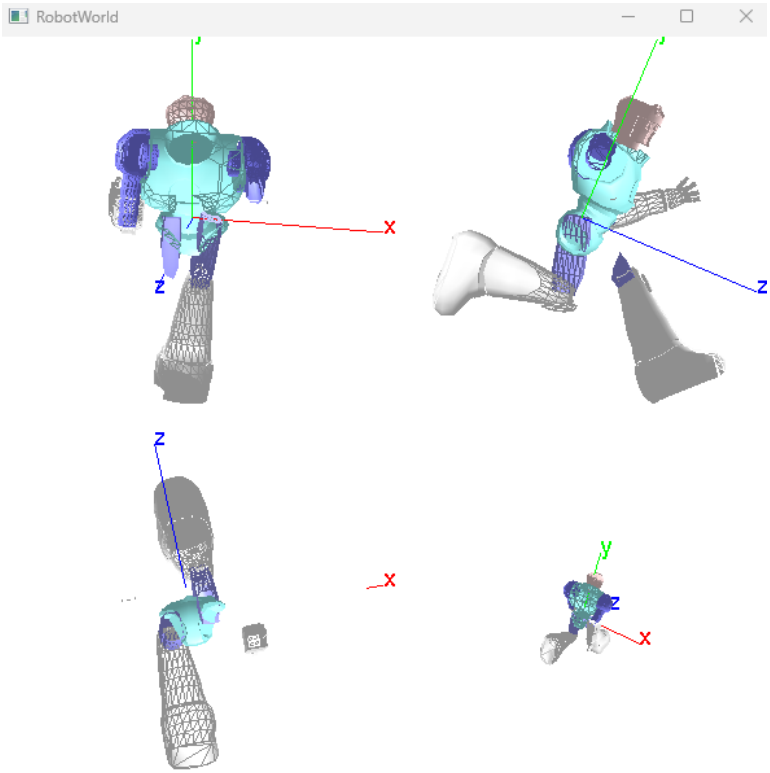
과제 7 코드를 이용하여 과제를 진행하였으며 뷰포트를 나누는 과제이므로 다른 파일은 수정하지 않았으며 RunningRobot.cpp의 display함수를 수정하고, 일부 함수를 추가하였다.

자신이 구현한 주요 코드 및 코드 설명

```
void setCameraOrtho() {
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(-1, 1, -1, 1, 1, 20); // 직교 투영을 설정
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
} // 카메라를 직교 투영 모드로 설정하는 함수

void setCameraPerspective() {
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluPerspective(60, 1, 1, 20); // 원근 투영을 설정
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
} // 카메라를 원근 투영 모드로 설정하는 함수

void display() {
        int width = glutGet(GLUT_WINDOW_WIDTH);
        int height = glutGet(GLUT_WINDOW_HEIGHT);
        // 원도우의 높이와 너비를 가져옴

        glClearColor(1.0, 1.0, 1.0, 1.0);
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        // 정면도
        glViewport(0, height / 2, width / 2, height / 2); // 첫 번째 뷰포트 설정(왼쪽 상단)
        setCameraOrtho(); // 직교 투영 설정
        gluLookAt(0, 0, 1, 0, 0, 0, 0, 1, 0); // 카메라 위치 및 방향 설정
        robot.draw(); // 로봇 그리기
```

```
        // 측면도
        glViewport(width / 2, height / 2, width / 2, height / 2); // 두 번째 뷰포트 설정 (오른쪽 상단)
        setCameraOrtho(); // 직교 투영 설정
        gluLookAt(1, 0, 0, 0, 0, 0, 0, 1, 0); // 카메라 위치 및 방향 설정
        robot.draw(); // 로봇 그리기

        // 평면도
        glViewport(0, 0, width / 2, height / 2); // 세 번째 뷰포트 설정 (왼쪽 하단)
        setCameraOrtho(); // 직교 투영 설정
        gluLookAt(0, 1, 0, 0, 0, 0, 0, 0, -1); // 카메라 위치 및 방향 설정
        robot.draw(); // 로봇 그리기

        // 원근 투상 결과
        glViewport(width / 2, 0, width / 2, height / 2); // 네 번째 뷰포트 설정 (오른쪽 하단)
        setCameraPerspective(); // 원근 투영 설정
        gluLookAt(3, 3, 3, 0, 0, 0, 0, 1, 0); // 카메라 위치 및 방향 설정
        robot.draw(); // 로봇 그리기

        glutSwapBuffers(); // 화면에 그림을 출력
} // 화면을 그리는 함수
```

다양한 입력에 대한 테스트 결과



해당 문제에 대한 고찰
해당 과제에서는 투상변환에서 직교 투영과 원근 투영을 이용하여 뷰포트를 분할하는 내용을 진행했다. 직교 투영이랑 원근 투영에 대해 아직 자세히 이해는 못한거 같지만 해당 OpenGL함수인 glOrtho과 glFrustum 함수를 사용하는 방식에 대해 알게된 시간이였던 거 같다. 추가 기능 과제인 마우스를 이용한 화면 조작 내용은 해결하지 못했다.

이번 과제에 대한 느낀점
모델변환과 투상변환에 대해 배운 내용을 실습에 적용을 하면서 뭔가 모호했던 부분에 대해 알게 된 시간이였다. 강의 처음시간에 OpenGL을 이용하여 로봇이 움직이는 것을 한다고 했을 때 어떻게 움직이게 할지 궁금했었는데 이번 시간을 통해 직접 해보는 시간을 가지게 되었고, 역시 여기서도 수학이 사용이 됐다. 그래픽 분야에서 수학이 많이 사용된다고 알고 있는데 이렇게 사용이 된다는 것을 알게 되었던 시간이었다.
궁금한 점이나 건의사항
궁금한 점이나 건의사항은 없습니다.