20011742 권혁재 Computer Graphics 과제2 보고서

이번 과제의 결과물은, 기존 베이스 코드를 변형시켜서 2를 눌렀을 때 나타나는 빨간 Target Point에 Robot Arm의 끝 단이 위치할 수 있도록 각도를 자동으로 계산하고, 시각적으로 완성도 있게 꾸미는 것이다.

1. 시각적으로 완성도 개선: 우선은 시작코드와 결과물의 코드는, 크기, 만들어야 할 개수 등이 모두 달랐기 때문에, drawRobotArm 함수부터 고치도록 하였다. 일단 하단 받침대가 2개로 나뉘어져 있고, 연결해주는 나사가 추가되었기 때문에, 하단 BASE파트를 2개로 나누고, 연결부 파트의 코드를 추가하여 시각적으로 변화하였다. 또한 팔 또한 Lower Arm부분이 2개로 나뉘어져 있고, 팔을 연결해주는 나사가 추가되어야 하기 때문에, Lower Arm 파트를 2개로 나뉘어 짜주고, 연결부 파트의 코드를 추가해주었다. 마지막으로 팔부분의 연결부 나사가 추가되있기 때문에, 연결부 파트의 코드를 추가해 주었다. 또한, 기존 파트의 크기와 위치등을 조정하여, 최대한 결과물과 비슷하게 코드를 작성하였다.

2. 시행착오: 우선, ComputeAngle의 코드를 작성하여 전역변수 flag1, flag2, flag3을 추가하여 팔들의 회전하는 방향을 지정하여, 방향전환을 하도록 ComputeAngle의 코드를 작성하였다. 이 코드를 작성하니, 일부동작은 정상적으로 작동하였으나, 과제의 2번째 목표인 빨간 Target Point에 Robot Arm의 끝 단이 위치하는 동작이 정상적으로 작동하지 못하였고, 이상하게 작동 혹은 아에 조건에 만족하지 못하게 작동함을 확인하였다.

3. 수행과정: 그러므로, 빨간점을 따라갈수 있게 해주는 추가적인 코드의 작성이 필요하였고, 이에 따라, CalcEnd함수(로봇 팔의 끝 부분이 현재 위치에서 어디에 있는지를 계산해주는 함수) + (주어진 각도 ang1, ang2, ang3에 따라 팔의 끝점 좌표를 구하여 vec4 d에 저장해준다) 와 distance함수(로봇 팔 끝과 빨간 점 사이의 거리를 계산해주는 함수, 팔의 현재 회전 각도가 90도 혹은 270도를 넘어가게 되면 팔이 이상하게 꺾이기 때문에 x좌표를 반전시켜준다) 함수를 추가하여, computeAngle함수에 targetPosition(빨간 점의 위치) 변수를 추가해줘서, 로봇 팔이 빨간 점을 따라가게 하고, 빨간 점으로부터 일정 거리 이상이 넘어가게 되면, 거리를 재계산하여 위치를 정상적으로 변경시켜 주었다. 최적의 각도를 찾지는 못한듯하지만, 조건은 착실히 따를수 있도록 각도를 조정시켜 주었다.

distance함수:

```
calcEnd(ang1, ang2, ang3); // 현재 팔 끝의 위치

if ((int)(g_time * 30) % 360 >= 90.0 && (int)(g_time * 30) % 360 <= 270.0) {

return ((d.x + p.x) * (d.x + p.x)) + ((d.y + 0.4 - p.y) * (d.y + 0.4 - p.y)) + ((d.z - p.z) * (d.z - p.z)); // 팔 끝과 목표 위치 사이의 거리 제곱을 반환 (x축 반전의 경우)

 }

else {

return ((d.x - p.x) * (d.x - p.x)) + ((d.y + 0.4 - p.y) * (d.y + 0.4 - p.y)) + ((d.z - p.z) * (d.z - p.z)); // 일반적인 거리 제곱 계산 (x축 반전하지 않음)

 }
```

CalcEnd함수:

```
mat4 temp = CTM; // 현재 매트릭스를 저장하여 나중에 복원

d = vec4(0.5, 0, 0, 1);

CTM = Translate(0, -0.4, 0) * RotateY(g_time * 30);

CTM *= RotateZ(ang1);

CTM *= Translate(0, 0.4, 0) * RotateZ(ang2);

CTM *= Translate(0, 0.4, 0) * RotateZ(ang3);

d = CTM * Scale(0.4, 0, 0) * d; // 각각의 회전과 위치의 적용을 통한 팔 끝점의 위치를 계산하고 vec4 d 변수에 저장해준다.

CTM = temp; // CTM을 기존 상태로 복원하여, 다른 연산에 영향X
```

4. 배운점: 수업시간에 배웠던 대로 rotate, translage의 순서가 중요함을 재확인할 수 있었으며, 색깔을 바꿔주는 fshader의 변화없이도 크기, 위치를 변화시켜주니 기존과 색상이 조금 다르게 나오는 것에 대해 조금 신기하였다. 빨간점을 따라다니는 코드의 구현이 어려웠으며, 목표를 따라다니는 단순한 동작에 대해 거리를 계산하고 각도를 조정해서 따라가야 하는 어려운 작업이 바탕으로 깔려야 한다는 점이 신기하였다.