**컴퓨터 그래픽스 HW3**

**학과: 컴퓨터공학부**

**학번: 201801569**

**이름: 송혜민**

**1.**

(1)우선 bull’s eye target을 보면 초록색 원이 가장 크고 보라색 원이 가장 작다.

Depth buffer와 직교 투영을 같이 이용하면 작아 보이는 원을 그대로 가져오지만 z값만 -near로 바꾸어 투영하게 된다.

따라서 원을 설정할 때 처음에 가장 큰 원의 z값과 직교투영시 z-near의 절대값을 똑같이 설정한다. 그 다음 원을 그릴 땐 z버퍼를 더 크게 주어서 원을 작아 보이게 만들고 원의 크기는 더 커도 작아 보이기 때문에 조금 더 큰 값으로 주었다.

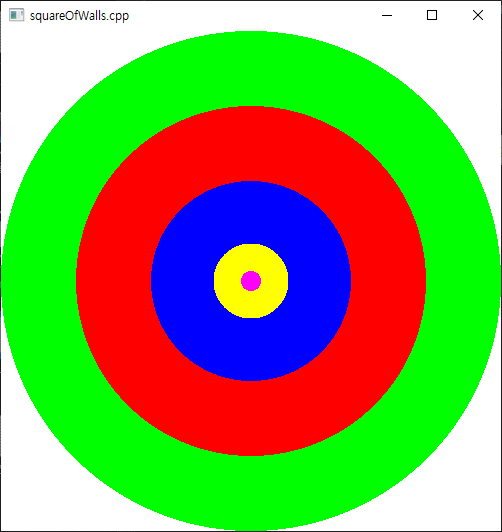
그 다음 원도 마찬가지로 적절히 설정하고 색상은 주어진 그림에 맞게 RGB색을 적절히 조절한다.

그림을 그리면 다음과 같다.

텍스트, 지도, 눈이(가) 표시된 사진

자동 생성된 설명

(2)Opengl을 이용하여 코드를 그리면 다음과 같은 결과가 나온다.



2.

우선 살 하나의 반대편을 그리기 위해 적당하게 좌표를 설정하여 그려주었다. 그린 결과는 다음 사진과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그 다음 이 살을 glscalef을 이용하여 반사를 시켜주는데 단순히 glscalef를 사용하고 끝내면 다 그린 뒤 반대로 뒤집힌 모습만 나와서 gl\_line\_strip을 한 번 더 사용하여 완전한 살 하나를 만들었다.

그 결과 살 하나만 출력해보면 다음과 같은 그림이 나온다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그 다음 이 살을 6번 출력하고 하나당 60도,120도,180,240도300도,360도 회전시켜주면 다음과 같은 결과가 나온다.

텍스트이(가) 표시된 사진

자동 생성된 설명

3.

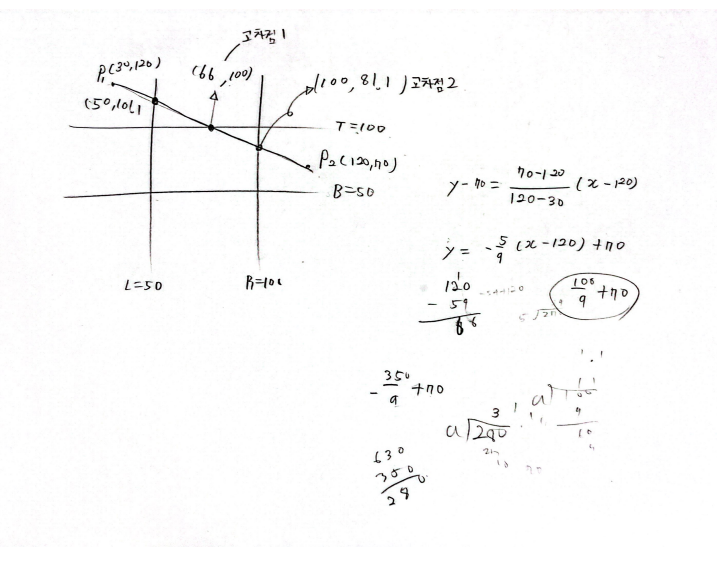
(1) 알고리즘 코드를 이용하여 계산하면 p1의 아웃코드는 9. 즉 1001이고 p2의 아웃코드는 2, 즉0010이 나오게 된다. 코드 실행 결과는 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

(2) 1의 선분과 viewing area의 모든 교차점을 손으로 구해보면 다음과 같이 나온다.

즉 2개의 교차점이 존재한다.



(3) 2의 결과를 콘솔에 출력하면 다음과 같은 결과가 나온다.

코드를 조금 변형하여 출력문을 whlie문 안에 넣어주었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

4.

(1)

선분의 기울기가 1보다 크면 DDA알고리즘에서 x의 크기를 1씩 증가시키는 것이 아니라 y의 크기를 1씩 증가시켜야 한다. 또한 기울기(m)는 y증가량/x증가량인데, 우리는 여기서 y증가량을 1로 하였으므로 x증가량은 1/m이 된다.

(2)

1을 기반으로 기울기가 1보다 클 경우 DDA알고리즘을 수정해보았다. 수정한 부분은

다음 사진과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

우선 x를 int로 정의하지 않고 float로 정의하였다. 그 이유는 기울기가 1보다 작을 때 x는 정수이지만 클 경우에는 소수형이기 때문에 y와 x 둘 다 float형으로 선언을 해 주었고 if-else문으로 1보다 큰 경우와 작은 경우를 나누었다.

작은 경우는 변화가 DDA알고리즘을 그대로 사용하였고 큰 경우는 1)에서 설명한 대로 y의 크기를 1씩 늘려주고 이럴 경우엔 x증가량이 1/m이 되기 때문에 x에 1/m씩 더해주었다. 또한 반올림을 해주어야 하므로 round(x)를 이용하였다.

(3)

2를 통해 수정한 코드를 기반으로 DDA(100,100,200,300)을 실행하면 다음과 같은 결과가 나온다.

