

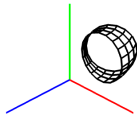
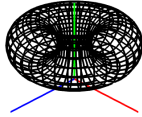
HW01

Building/Drawing a Polygonal Torus Model

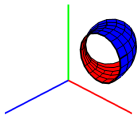
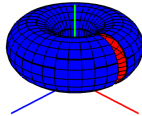
Self-Scoring table

	01	02	03	04
Score	1	1	1	1

-Task1

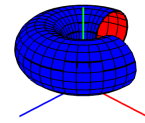
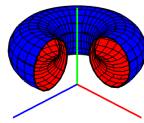
Draw the wireframe only	
	

-Task2

Draw the quads and the wireframe	
	

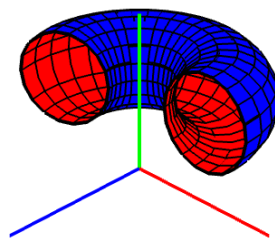
-Task3

Revolving angle control around the y-axis



-Task4

Revolving angle control around the y-axis



How to generate 36X18 points using GLM

```
float angle = 2.0f * M_PI * i / numPoints;
```

처음 그리는 원은 360도를 원을구성하는 점의 개수인 18로 나눈 각을

점의 index가 증가함에 따라 곱해주어 angle을 구했고

```
x = radius * cos(angle) + 0.5f;
y = radius * sin(angle) + 0.5f;
float z = 0;
```

x,y평면에 그려야 하기 때문에 z값은 0으로

x는 반지름에 *cos, y는 반지름*sin 으로 설정한뒤 가시성을위해 중심을 0.5만큼 옮겨주었다.

이후에 그리는 원은

```
for(int j=1;j<36;j++){
    for (int i = 0; i < numPoints; ++i) {
        mat4 M = glm::rotate(glm::mat4(1.0f), theta, axis);
        vec4 point(p[j-1][i].x, p[j-1][i].y, p[j-1][i].z, 1.0f);
        p[j][i]=vec3(M*point);
    }
}
```

이전 p값에 y축을 기준으로하는 회전 행렬을 곱하여 theta(10도)만큼 회전된 점들을 구하고

그 점을 현재 p값에 저장하는 식으로 구하였다.

이때 행렬이 4X4행렬임으로 vec4과의 곱을 이용해 회전한 값을 구한뒤 vec3으로 변환해주었다

회전을 시계방향으로 하는경우 -theta만큼 회전으로 진행되게 하였다.

How to discriminate outside/inside of polygons

```
normal =
glm::normalize(glm::cross(p[i][nextIndex] - p[i][j], p[i + 1][j] - p[i][j]));
if(dot(normal, eye) > 0.0f) {
    if(delta>0){
        glColor3f(1.0f, 0.0f, 0.0f);
    }else{
        glColor3f(0.0f, 0.0f, 1.0f);
    }
} else {
    if(delta>0){
        glColor3f(0.0f, 0.0f, 1.0f);
    }else{
        glColor3f(1.0f, 0.0f, 0.0f);
    }
}
```

안과 바깥의 구별은 해당평면의 normal vector와 view vector의 내적값을 이용해 판단했는데 이때 양의 값이 나오면 view와 normal vector가 같은 방향이라는 뜻이기 때문에 안쪽 즉 빨간색으로 설정하였고 음의 값이 나오면 다른 방향이라는 뜻이기 때문에 파란색으로 설정하였다.

구현상 시계방향으로 회전을 시작하면 외적시 방향이 바뀌기 때문에 normal vector값의 부호가 바뀌고 색이 반전되어 나오게 된다 따라서 우리가 원하는 그림을 얻기 위해 예외처리를 해주었다.