KWANGWOON UNIVERSITY

컴 퓨 터 그 래 픽 스

HW01 Drawing Torus

담당 교수님 : 최민규 교수님

제 출 날 짜 : 2018. 11. 08.

학 과 : 컴퓨터소프트웨어학과

학 번 : 2014707040

이 름 : 유 진 혁

기능 별 구현 방법 설명 및 결과 화면

(1). Draw 36x18 data points using 'GL_POINTS': '1'key

- 구현 방법

```
float cos_10d = cos(10 * 3.14159 / 180.0);
float sin_10d = sin(10 * 3.14159 / 180.0);
// Initial circle on plane xy
for (int j = 0; j < 18; j++)
{
    float cos_th = cos(j * 20 * 3.14159 / 180.0);
    float sin_th = sin(j * 20 * 3.14159 / 180.0);
    p[0][j][0] = cos_th * radius + 3.0;
    p[0][j][1] = sin_th * radius + 3.0;
    p[0][j][2] = 0;
}
// Rotate initial circle around the y-axis
for (int i = 1; i < 36; i++)
    for (int j = 0; j < 18; j++)
    {
        float sin_th = sin(j * 20 * 3.14159 / 180.0);
        p[i][j][0] = p[i - 1][j][0] * cos_10d + p[i - 1][j][2] * sin_10d;
        p[i][j][1] = sin_th * radius + 3.0;
        p[i][j][2] = -p[i - 1][j][0] * sin_10d + p[i - 1][j][2] * cos_10d;
    }
```

InitializePoint 함수 내의 일부 코드이다. 각 점들의 좌표를 배열 p에 저장한다. 먼저, 삼각함수를 이용하여 xy 평면 위의 원의 좌표를 저장한다. 그리고 이 원을 y축 중심으로 10도 회전 변환하여 다음 원의 좌표를 만들고, 이 과정을 반복하여 torus를 이루는 점의 좌표를 배열에 저장한다.

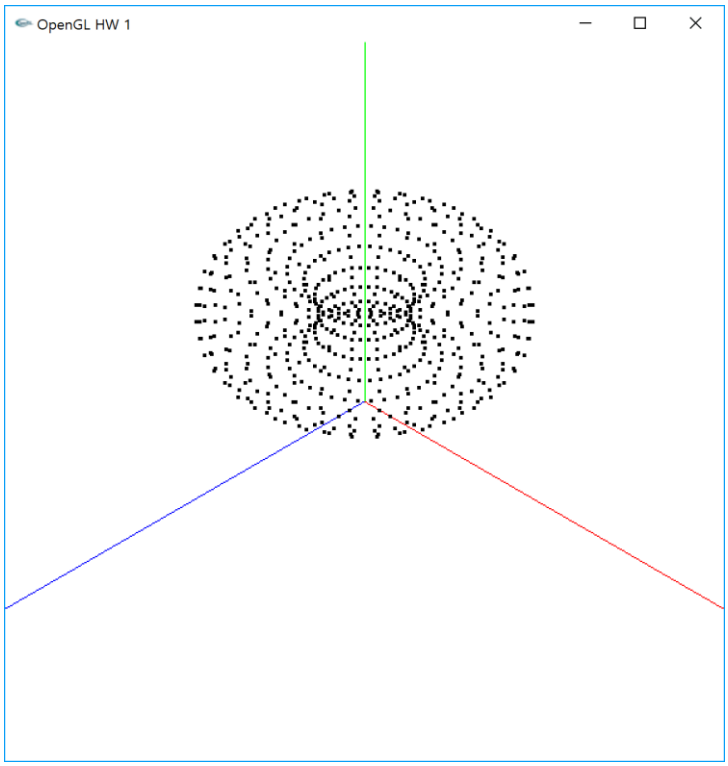
```
void RenderPoints()
{
    // Clear the window with current clearing color
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    gluLookAt(1, 1, 1, 0, 0, 0, 0, 1, 0);

    DrawAxis();

    InitializePoint(1.7);
    glColor3f(0, 0, 0); // black
    glPointSize(3);
    glBegin(GL_POINTS);
    {
        for (int i = 0; i < controlAngleY; i++)
            for (int j = 0; j < controlAngleZ; j++)
            {
                glVertex3fv(p[i][j]);
            }
    }
    glEnd();

    if (toggleNormalOfPolygon)
        DrawNormalOfPolygon();
    if (toggleNormalOfPoint)
        DrawNormalOfPoint();

    glutSwapBuffers();
}
```

RenderPoints는 화면에 점들을 그리는 함수이다. 축을 그리고 torus를 이루는 원의 반지름을 1.7로 하여 점의 좌표를 초기화한다. 그리고 점의 크기를 3으로, 색은 검은색으로 설정 후, GL_POINTS로 점들을 화면에 그린다. 위 코드 하단의 조건문은 normal vector 출력을 위한 조건문이다.

```
case '1':
    glutDisplayFunc(RenderPoints);
    glutPostRedisplay();
    break;
```

키보드에서 숫자 '1' 키를 누르면 RenderPoints 함수를 화면에 그려지는 함수로 등록하고 화면을 갱신한다.

- 결과 화면



(2). Draw the wireframe only: '2' key

- 구현 방법

```
void DrawQuads()
{
    glBegin(GL_QUADS);
    {
        for (int i = 0; i < controlAngleY; i++)
        {
            for (int j = 0; j < controlAngleZ; j++)
            {
                glVertex3fv(p[i][j]);
                glVertex3fv(p[(i + 1) % 36][j]);
                glVertex3fv(p[(i + 1) % 36][(j + 1) % 18]);
                glVertex3fv(p[i][(j + 1) % 18]);
            }
        }
    }
    glEnd();
}
```

DrawQuads는 torus를 이루는 사각형을 GL_QUADS로 그리는 함수이다. 원 하나를 이루는 반복문과 y축 중심으로 회전시키는 반복문을 이중 for문으로 만들어 구현하였다. 배열의 범위를 넘지 않도록 % 연산자를 넣었다.

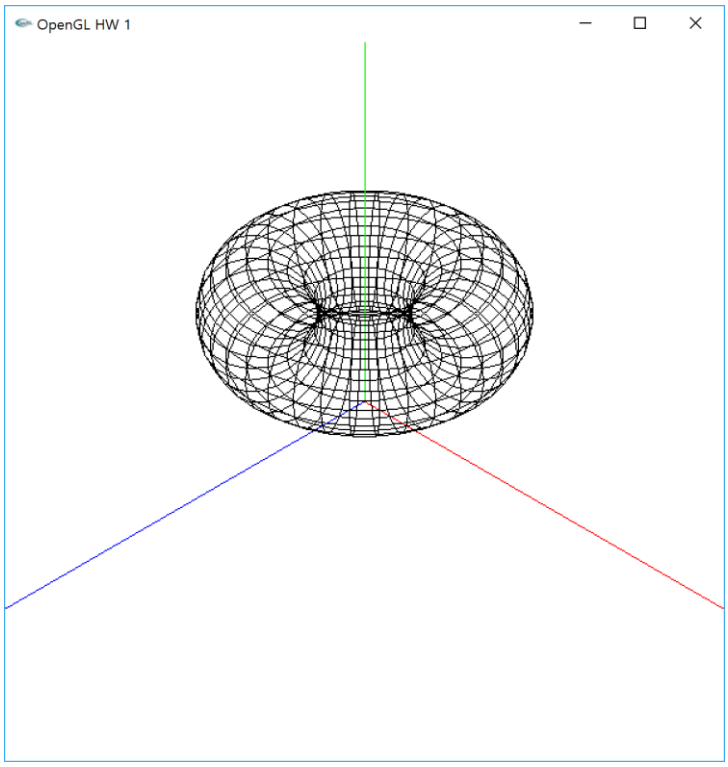
```
void RenderWireframe()
{
    // Clear the window with current clearing color
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    gluLookAt(1, 1, 1, 0, 0, 0, 0, 1, 0);

    DrawAxis();

    InitializePoint(1.7);
    glColor3f(0, 0, 0); // black
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
    DrawQuads();

    if (toggleNormalOfPolygon)
        DrawNormalOfPolygon();
    if (toggleNormalOfPoint)
        DrawNormalOfPoint();

    glutSwapBuffers();
}
```

축을 먼저 그리고 원의 반지름을 1.7로 하여 torus의 점의 좌표를 설정한다. 색은 검은색으로 지정하고 DrawQuads 함수를 호출하여 wireframe을 그린다.

```
case '2':
    glutDisplayFunc(RenderWireframe);
    glutPostRedisplay();
    break;
```

키보드의 '2' 키를 누르면 RenderWireframe 함수를 화면에 그려지는 함수로 등록하고 화면을 갱신한다.

- 결과 화면



(3). Draw the quads only: '3' key

- 구현 방법

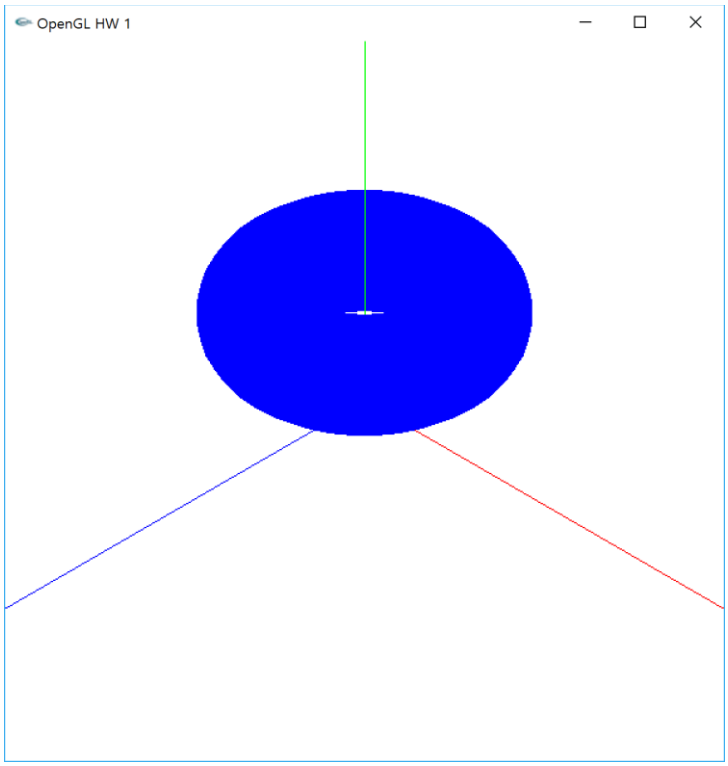
```
void RenderQuads()
{
    // Clear the window with current clearing color
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    gluLookAt(1, 1, 1, 0, 0, 0, 0, 1, 0);

    DrawAxis();

    InitializePoint(1.7);
    glColor3f(0, 0, 1); // blue
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
    DrawQuads();

    if (toggleNormalOfPolygon)
        DrawNormalOfPolygon();
    if (toggleNormalOfPoint)
        DrawNormalOfPoint();

    glutSwapBuffers();
}
```

wireframe을 그리는 것과 같은 방법으로 구현하였다. 다만, 색을 파란색으로 설정하고 사각형의 선을 그리는 것이 아닌 면을 그린다. 그래서 glPolygonMode에서 GL_LINE 대신 GL_FILL이 쓰였다.

```
case '3':
    glutDisplayFunc(RenderQuads);
    glutPostRedisplay();
    break;
```

키보드의 '3' 키를 누르면 RenderQuads 함수를 화면에 그려지는 함수로 등록하고 화면을 갱신한다.

- 결과 화면



(4). Draw the quads and the wireframe: '4' key

- 구현 방법

```
void RenderQuadsAndWireframe()
{
    // Clear the window with current clearing color
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    gluLookAt(1, 1, 1, 0, 0, 0, 0, 1, 0);

    DrawAxis();

    // inside wireframe
    InitializePoint(1.69);
    glColor3f(0, 0, 0); // black
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
    DrawQuads();
```

먼저 축을 그리고, torus 내부와 외부에 그려질 wireframe을 그리기 위해 offsetting 방법을 사용하였다. 원의 반지름을 1.69로 하여 torus의 점의 좌표를 설정한다. 그리고 색을 검은색으로 지정하고 GL_LINE 모드로 내부 wireframe을 그린다.

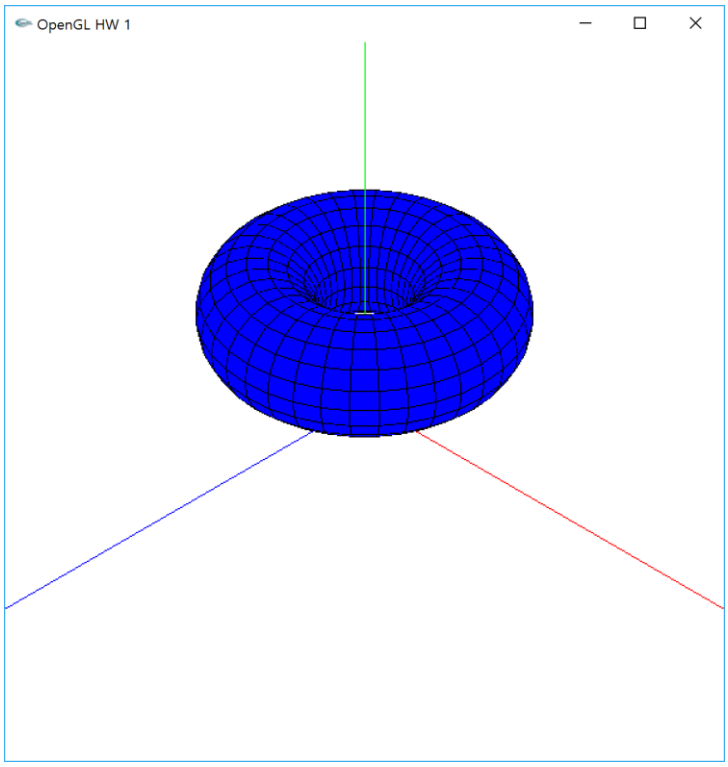
```
    // quad plane
    InitializePoint(1.7);
    glColor3f(0, 0, 1); // blue
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
    DrawQuads();

    if (toggleNormalOfPolygon)
        DrawNormalOfPolygon();
    if (toggleNormalOfPoint)
        DrawNormalOfPoint();

    // outside wireframe
    InitializePoint(1.71);
    glColor3f(0, 0, 0); // black
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
    DrawQuads();

    glutSwapBuffers();
}
```

그리고 torus를 이루는 원의 반지름을 1.7로 하여 좌표를 다시 설정하고 파란색으로 torus의 면을 그린다. 다시, 외부 wireframe을 그리기 위해 원의 반지름을 1.71로 하여 torus의 좌표를 설정해주고, 검은색 외부 wireframe을 그린다.

```
case '4':
    glutDisplayFunc(RenderQuadsAndWireframe);
    glutPostRedisplay();
    break;
```

키보드의 '4' 키를 누르면 RenderQuadsAndWireframe 함수를 화면에 그려지는 함수로 등록하고 화면을 갱신한다.

- 결과 화면



(5). Two-sided constant shading with the wireframe: '5' key

- 구현 방법

```
// Assign normal vectors of planes to array
// Normal vector of p[i][j], p[i+1][j], p[i+1][j+1], p[i][j+1] four-point polygon = normal[i+1][j+1]
for (int i = 0; i < 36; i++)
    for (int j = 0; j < 18; j++)
    {
        Position v1, v2, v3, v4;
        Position n1, n2;
        Vector(p[i][j], p[(i + 1) % 36][j], v1);
        Vector(p[i][j], p[i][(j + 1) % 18], v2);
        CrossProduct(v1, v2, n1);
        Vector(p[(i + 1) % 36][(j + 1) % 18], p[i][(j + 1) % 18], v3);
        Vector(p[(i + 1) % 36][(j + 1) % 18], p[(i + 1) % 36][j], v4);
        CrossProduct(v3, v4, n2);
        normal[(i + 1) % 36][(j + 1) % 18][0] = (n1[0] + n2[0]) / 2.0f;
        normal[(i + 1) % 36][(j + 1) % 18][1] = (n1[1] + n2[1]) / 2.0f;
        normal[(i + 1) % 36][(j + 1) % 18][2] = (n1[2] + n2[2]) / 2.0f;
    }
```

torus 내부와 외부의 색을 다르게 칠하기 위해선 torus를 구성하는 각 사각형 면의 법선 벡터가 필요하다. 따라서, InitializePoint 함수 내에는 위와 같은 코드가 포함되어 있다. 사각형의 네 점을 이용하여 벡터 v1, v2, v3, v4를 만들고 이들끼리 외적하여 두 개의 면의 법선 벡터를 구한다. 이 두 법선 벡터를 평균 내어 전체 사각형 면의 법선 벡터를 구하고, 법선 벡터를 담는 배열 normal에 저장한다.

```
void RenderTwoSidedShading()
{
    // Clear the window with current clearing color
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    gluLookAt(1, 1, 1, 0, 0, 0, 0, 1, 0);

    DrawAxis();

    // inside wireframe
    InitializePoint(1.69);
    glColor3f(0, 0, 0); // black
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
    DrawQuads();
```

함수 RenderTwoSidedShading은 RenderQuadsAndWireframe 함수와 똑같이 먼저 축을 그리고 torus 내부 wireframe을 그린다.

```
// shaded quad plane
InitializePoint(1.7);
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
glBegin(GL_QUADS);
{
    for (int i = 0; i < controlAngleY; i++)
    {
        for (int j = 0; j < controlAngleZ; j++)
        {
            // Dot product
            GLfloat dot = normal[(i + 1) % 36][(j + 1) % 18][0] + normal[(i + 1) % 36][(j + 1) % 18][1] + normal[(i + 1) % 36][(j + 1) % 18][2];
            if (dot >= 0)
                glColor3f(0, 0, 1); // blue
            else
                glColor3f(1, 0, 0); // red
            glVertex3fv(p[i][j]);
            glVertex3fv(p[(i + 1) % 36][j]);
            glVertex3fv(p[(i + 1) % 36][(j + 1) % 18]);
            glVertex3fv(p[i][(j + 1) % 18]);
        }
    }
}
glEnd();
```

torus의 내부와 외부를 다른 색으로 칠하기 위해선 카메라 위치에서 바라볼 때 내부가 보이는지 외부가 보이는지 판단할 필요가 있다. 따라서, 원점에서 카메라 방향의 벡터와 사각형 면의 법선 벡터를 내적 하여 그 값의 부호를 확인한다. 이 때, 카메라의 위치는 (1, 1, 1) 이다. 내적 값이 양수라면 외부가 보이는 것이므로 색을 파란색으로 지정하고, 음수라면 내부가 보이는 것이므로 빨간색으로 지정한다. 그리고 사각형을 그린다.

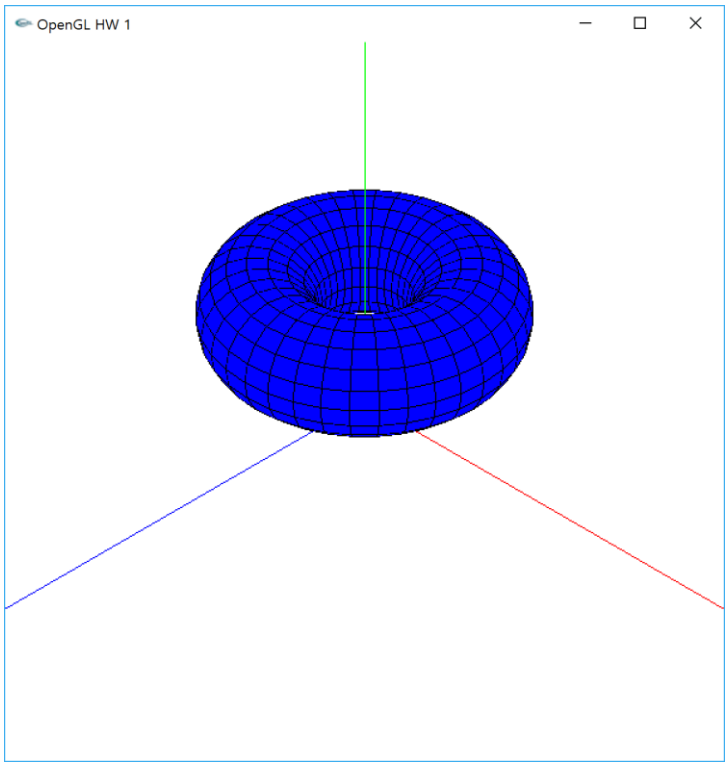
```
    if (toggleNormalOfPolygon)
        DrawNormalOfPolygon();
    if (toggleNormalOfPoint)
        DrawNormalOfPoint();

    // outisde wireframe
    InitializePoint(1.71);
    glColor3f(0, 0, 0); // black
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
    DrawQuads();

    glutSwapBuffers();
}
```

torus 원의 반지름을 1.71로 하여 점의 좌표를 다시 설정하고 외부 wireframe을 그린다.

```
case '5':
    glutDisplayFunc(RenderTwoSidedShading);
    glutPostRedisplay();
    break;
```

키보드의 '5' 키를 누르면 RenderTwoSidedShading 함수를 화면에 그려지는 함수로 등록하고 화면을 갱신한다.

- 결과 화면



torus의 내부가 보이지 않을 땐 RenderQuadsAndWireframe 함수를 그렸을 때와 똑같은 결과 화면이 그려진다.

(6). Around the y-axis with the 'a', 's' keys

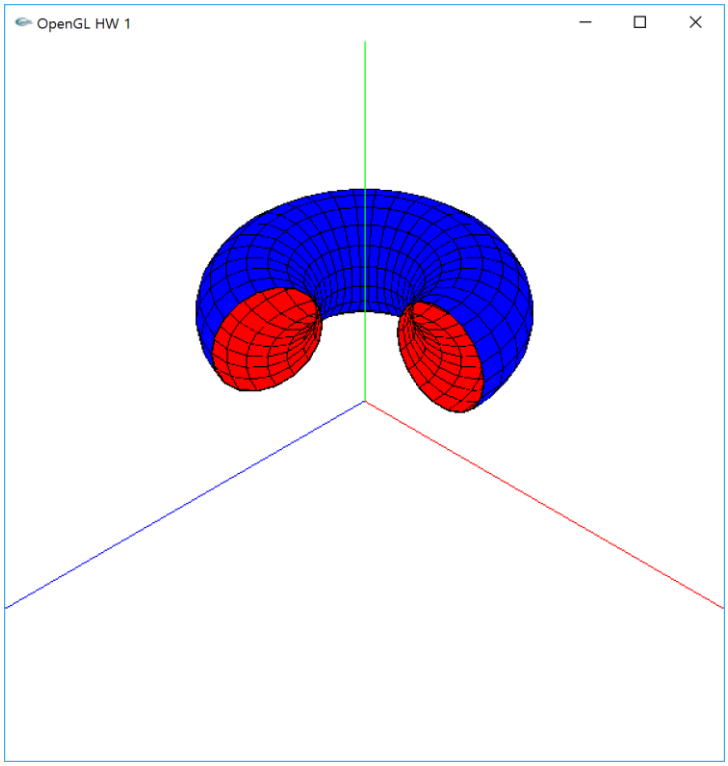
- 구현 방법

```
case 'a':
    if (controlAngleY < 36)
        controlAngleY++;
    glutPostRedisplay();
    break;
case 's':
    if (controlAngleY > 0)
        controlAngleY--;
    glutPostRedisplay();
    break;
```

controlAngleY는 y축 중심으로 회전 변환하며 원을 그릴 때 반복 횟수를 나타내는 변수이다. 즉, y

축 중심으로 회전하는 각도를 의미하고, 초기값으로 36을 갖는다. 키보드의 'a' 키와 's' 키를 누르면 범위 내에서 이 값이 증가하고 감소하며 화면이 갱신된다.

- 결과 화면



위 결과 화면은 's' 키를 눌러 controlAngleY의 값을 줄인 결과이다. 그려지는 횟수가 줄면서 완전한 torus가 아닌 잘린 torus가 그려진다. (5)의 결과에 적용하여 torus 내부가 빨간색으로 칠해져 있다.

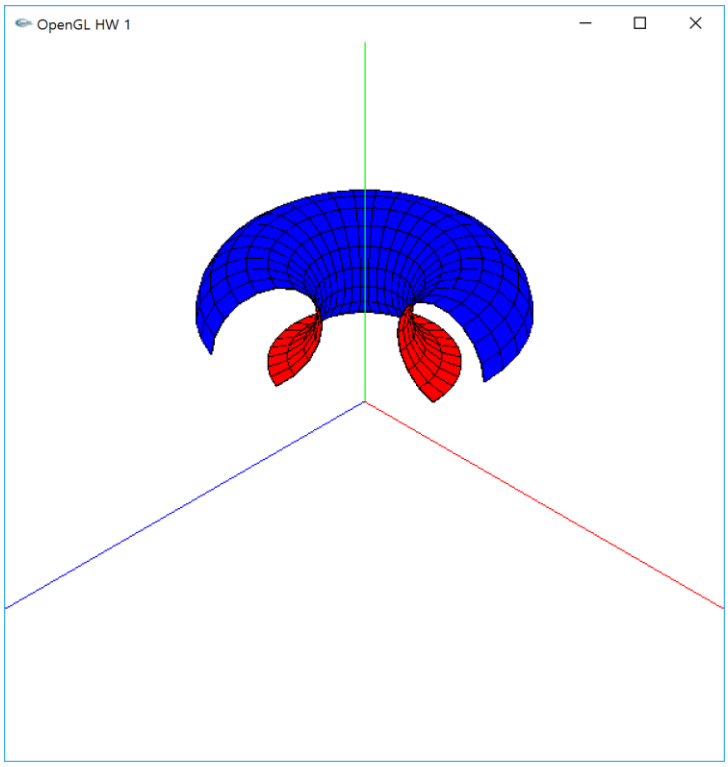
(7). Around the z-axis with the 'j', 'k' keys

- 구현 방법

```
case 'j':
    if (controlAngleZ < 18)
        controlAngleZ++;
    glutPostRedisplay();
    break;
case 'k':
    if (controlAngleZ > 0)
        controlAngleZ--;
    glutPostRedisplay();
    break;
```

변수 controlAngleZ는 torus를 구성하는 원, 즉, y축 중심으로 회전 변환되는 원의 중심각을 의미한다. 초기 값은 18로 완전한 모양의 원을 회전시켜 torus가 그려지고, 이 값이 줄면 잘린 원을 회전시켜 torus가 그려진다. 키보드의 'j' 키와 'k' 키를 누르면 범위 내에서 이 값이 증가하고 감소하며 화면이 갱신된다.

- 결과 화면



(6)의 결과에 'k' 키를 5번 누른 결과이다. 완전한 모양이 아닌 원이 y축 중심으로 회전 변환하여 torus를 만들었다.

(8). Draw the normal vectors of the polygons: toggle with '6' key

- 구현 방법

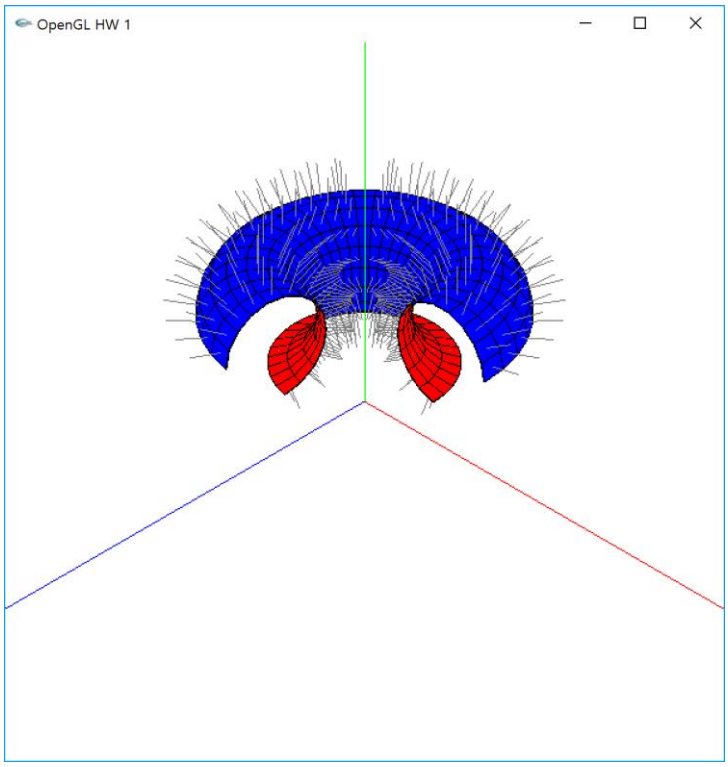
```
void DrawNormalOfPolygon()
{
    glColor3f(0.5, 0.5, 0.5); // gray
    glBegin(GL_LINES);
    {
        for (int i = 0; i < controlAngleY; i++)
        {
            for (int j = 0; j < controlAngleZ; j++)
            {
                Position center;    // the center of a polygon
                center[0] = (p[i][j][0] + p[(i + 1) % 36][j][0] + p[(i + 1) % 36][(j + 1) % 18][0] + p[i][(j + 1) % 18][0]) / 4.0f;
                center[1] = (p[i][j][1] + p[(i + 1) % 36][j][1] + p[(i + 1) % 36][(j + 1) % 18][1] + p[i][(j + 1) % 18][1]) / 4.0f;
                center[2] = (p[i][j][2] + p[(i + 1) % 36][j][2] + p[(i + 1) % 36][(j + 1) % 18][2] + p[i][(j + 1) % 18][2]) / 4.0f;
                glVertex3fv(center);
                glVertex3f(center[0] + normal[(i + 1) % 36][(j + 1) % 18][0], center[1] + normal[(i + 1) % 36][(j + 1) % 18][1], center[2] + normal[(i + 1) % 36][(j + 1) % 18][2]);
            }
        }
    }
    glEnd();
}
```

DrawNormalOfPolygon은 torus를 구성하는 사각형들의 법선 벡터를 그리는 함수이다. 색은 회색으로 지정하고 GL_LINE으로 그린다. 각 사각형의 네 점으로 사각형 중심의 좌표를 구한다. 이 중심 좌표에서부터 법선 벡터만큼 이동한 위치까지 선을 그려 법선 벡터를 표현한다.

```
case '6':
    if (toggleNormalOfPolygon)
        toggleNormalOfPolygon = false;
    else
        toggleNormalOfPolygon = true;
    glutPostRedisplay();
    break;
```

키보드의 '6' 키를 누르면 bool형 변수 toggleNormalOfPolygon의 값이 전환되며 DrawNormalOfPolygon 함수의 호출 여부가 변경된다.

- 결과 화면



(9). Draw the normal vectors of the points: toggle with '7' key

- 구현 방법

```
void DrawNormalOfPoint()
{
    glColor3f(0.5, 0.5, 0.5);
    glBegin(GL_LINES);
    {
        for (int i = 0; i < controlAngleY; i++)
        {
            for (int j = 0; j < controlAngleZ; j++)
            {
                // Calculate a normal vector of a point by averaging normal vectors of polygons surrounding the point
                Position n;
                n[0] = (normal[i][j][0] + normal[(i + 1) % 36][j][0] + normal[i][(j + 1) % 18][0] + normal[(i + 1) % 36][(j + 1) % 18][0]) / 4.0f;
                n[1] = (normal[i][j][1] + normal[(i + 1) % 36][j][1] + normal[i][(j + 1) % 18][1] + normal[(i + 1) % 36][(j + 1) % 18][1]) / 4.0f;
                n[2] = (normal[i][j][2] + normal[(i + 1) % 36][j][2] + normal[i][(j + 1) % 18][2] + normal[(i + 1) % 36][(j + 1) % 18][2]) / 4.0f;
                glVertex3fv(p[i][j]);
                glVertex3f(p[i][j][0] + n[0], p[i][j][1] + n[1], p[i][j][2] + n[2]);
            }
        }
    }
    glEnd();
}
```

DrawNormalOfPoint는 torus의 점에서의 법선 벡터를 그리는 함수이다. DrawNormalOfPolygon과 마찬가지로 GL_LINE을 이용하여 같은 방식으로 그린다. 점의 법선 벡터는 한 점을 포함하는 네 개의 사각형의 법선 벡터를 평균 내어 구한다.

```
case '7':
    if (toggleNormalOfPoint)
        toggleNormalOfPoint = false;
    else
        toggleNormalOfPoint = true;
    glutPostRedisplay();
    break;
```

키보드의 '7' 키를 누르면 bool형 변수 toggleNormalOfPoint의 값이 전환되며 DrawNormalOfPoint 함수의 호출 여부가 변경된다.

- 결과 화면