SweptSurface_OpenGL

SweptSurface 를 openGL 을 이용하여 구현.

[실행 환경]
Ubuntu 18.04
OpenGL version string: 2.1 Mesa 20.0.8

sudo apt-get install g++
sudo apt-get install freeglut3-dev
sudo apt-get install mesa-utils

[실행 방법]
g++ -o ./mesh ./mesh.cpp -lGL -lglut -lm -lGLU
./mesh

```
230     ifstream ifs;
231     ifs.open("trombone.txt"); rotationsign = -1;
232     // ifs.open("coke_bottle.txt"); rotationsign = 1;
233     // ifs.open("joystick.txt"); rotationsign = 1;
```

231 줄에 위치하는 ifstream 구문이다.
각 줄에 적힌 coke_bottle 이나 trombone , joystick 을 주석처리를 수정하여 실행해볼 수 있다.

[조작 방법]
그냥 마우스로 드래그 시 rotate
shift 키를 누른 채로 드래그 시 translation
ctrl 키를 누른 채로 위로 드래그 시 zoom in, 아래로 드래그 시 zoom out
alt 키를 누른 채로 위로 드래그 시 dolly in, 아래로 드래그 시 dolly out

1. Control points and transformations of the cross sections are provided in the data file. The format is described at the end of this assignment description. Your system should be able to parse the standard format.

Ifstream 을 이용하여 curve_type 을 받아 BSPLINE 인지 아닌지 여부를 판단한다. 그 후 section 의 개수와 control point 의 개수를 받고, for 문을 통해 각 section 에서의 값들을 배열에 저장한다.

첨부 되어있는 coke_bottle.txt 테스트케이스의 경우 주석 처리가 마구 되어 있어 ifstream 을 통해 입력을 수월하게 받을 수 있도록 정해진 data file demands 형식으로 바꾸어 놓았다.

```
string curve_type;
ifs >> curve_type;

bool isBspline = false;
if(curve_type == "BSPLINE")
{
    isBspline = true;
}

int num_section;
ifs >> num_section;

int num_points;
ifs >> num_points;
```

```
dvec2 ctrl[50][50]; // coord of [i th section][j th ctrl point]
double scale[50]; // scale of [i th section]
dvec4 rotate[50];
dvec3 posit[50];

for(int i=0; i<num_section; i++)
{
    for(int j=0; j<num_points; j++)
    {
        ifs >> ctrl[i][j][0];
        ifs >> ctrl[i][j][1];
    }
    ifs >> scale[i];
    ifs >> rotate[i][0];
    ifs >> rotate[i][1];
```

2. Construct each cross section as a closed curve using either B-splines or Catmull-Rom splines. The curve type is provided in the data file.

```
for(int i=0; i<(num_section) * (num_in_section-1) + 1; i++)
{
    for(int j=0; j<num_points; j++)
    {
        for(int k=0; k<num_in_points; k++)
        { // overall 0 ~ (num_points*num_in_points - 1) amounts
            float t = (float)k / float(num_in_points);
            if(isBspline)
            {
                in_points[i][k + j*num_in_points][0] = (1.0-t)*
                in_points[i][k + j*num_in_points][1] = (1.0-t)*

            }
            else
            {
                in_points[i][k + j*num_in_points][0] = (-0.5*t+
                in_points[i][k + j*num_in_points][1] = (-0.5*t+
            }
```

isBspline 의 여부에 따라 Bspline 이라면 cubic Bspline 의 정의 Basis 를 이용하였고, Catmull-Rom spline 의 경우에도 알맞은 basis 를 이용하여 parametrize 하였다.

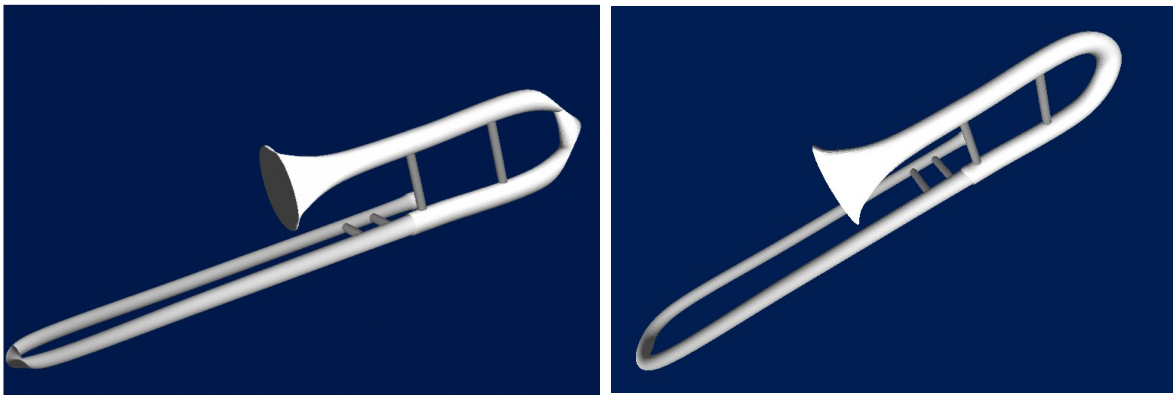
맨 끝점의 경우에는 끝점이 같은 위치에 하나 더 있다는 가정을 하여 interpolation 을 진행하였다.

3. The transformation of each cross section represents the scaling of the cross section, followed by its rotation, followed by its translation. Construct three splines for scaling factors, unit quaternions, and 3D positions.

```
float t = ((float)(i%num_in_section))/((float)num_in_section);
in_scale[i] = (-0.5*t+t*t-0.5*t*t*t)*scale[n-1] + (1.0-2.5*t*t+1.5*t*t*t)*sca
in_rotate[i] = (-0.5*t+t*t-0.5*t*t*t)*rotate[n-1] + (1.0-2.5*t*t+1.5*t*t*t)*r
in_posit[i] = (-0.5*t+t*t-0.5*t*t*t)*posit[n-1] + (1.0-2.5*t*t+1.5*t*t*t)*pos
```

In_scale, in_rotate, in_posit 은 각각 double[], dvec4[], dvec3[]이며, scale 과 rotate, position 각 element 들을 인접한 interpolated 평면 사이에서 그림과 같이 generalized catmull-rom spline interpolation 을 사용하였다. In_rotate 의 경우 unit quaternion 에 대한 언급이 나오는데, 이는 mesh 구현 시 unit quaternion 을 생성하여 좌표를 만드는 데 사용되었다.

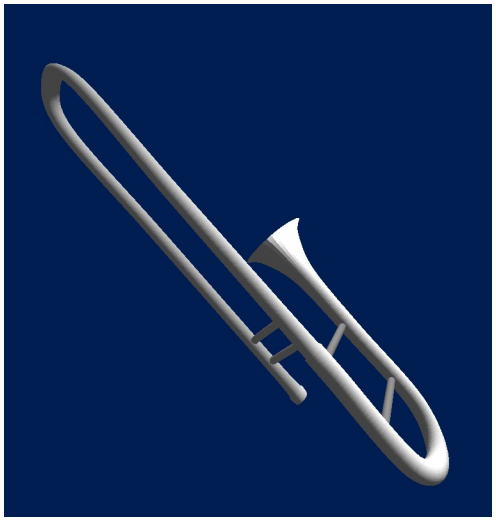
Surface 의 rotation 을 표현할 때, 도넛 모양을 그린다고 하면 회전을 진행할 때 origin 의 진행 방향과 y 축이 일치하는 경우가 있고, 반대되는 경우가 있다. 각각 경우에 따라서 unit quaternion rotation 을 그대로 적용할지, invert 를 적용할지가 달라진다. 주어진 예제 중 trombone 의 경우, 주어진 그대로 rotation 을 사용하게 되면 아래 왼쪽 그림과 같은 문제가 생긴다. 따라서, 이런 경우 static int rotationsign 을 정의하여 처리해주면 아래 오른쪽 그림과 같이 문제가 해결된다.

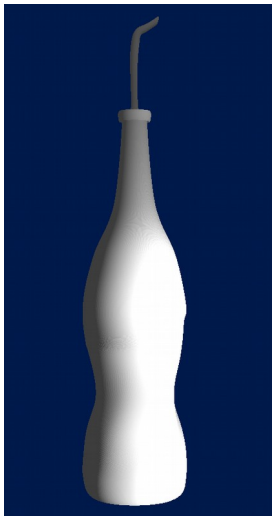


4. Visualize the swept surface as a polygonal mesh. The rendering should be styled to present the shape of the surface clearly.

인접하는 interpolated 평면 위의 interpolated spline 위의 점들이 있다. 해당 점의 좌표를 world coordinate 기준으로 표현하기 위해선, 구해놓은 interpolated rotation 값을 unit quaternion 으로 변환한 뒤, v' = qvq^-1 를 이용하여 (x, 0, z) 벡터를 회전시켜 준다. 그 벡터에 scale 을 곱한 뒤 translation 값과 더해주면 점들의 world coord 기준 좌표를 순서대로 구할 수 있다.

그 점들을 순서대로 인접한 평면끼리 삼각형 두 개로 이루어진 사각형들을 그어 법선 벡터를 설정해 주고 mesh 를 형성하였다. 아래는 두 testcase 의 mesh 구현 결과이다.





5. Allow for the user to rotate the scene so that we can inspect your surfaces at different viewpoints.

기존의 과제 2 에서 구현하였던 camera module 을 적용하였다.

6. Create your own swept surfaces that are aesthetically pleasing.

십자 키와 네 개의 나누어진 키가 있는 조이스틱을 제작하여 보았다.