Assignment 2:
Assignment 2 : Geometric Modeling

NAME: YIN HAIRUI
STUDENT NUMBER: 2020533028
EMAIL: YINHR@SHANGHAITECH.EDU.CN

1 INTRODUCTION

The followings are all jobs that have been doen, containing basic task and the bonus of NURBS surface!

- implementation of basic iterative de Casteljau Bézier vertex evaluation
- Construction of Bézier Surface with normal evaluation
- Rendering a Bézier Surface
- Generate tea object
- Finish NURBS surface construction

2 IMPLEMENTATION DETAILS

2.1 iterative de Casteljau Algorithm

First we need to finish the part in "bezier.cpp" of "Vertex BezierCurve::evaluate". The main idea of the realization comes from a iterative equation:

$$P_{i,j} = (1-u)P_{i-1,j} + uP_{i-1,j+1}$$

where $i = 1, 2, ..., n$, $j = 0, 1, ..., n-i$
I define a assistant function called deCasteljau() outside evaluate() to help iterative loops.

2.2 Construction of Bézier Surface

The main idea of create a surface is to firstly, construct m intermediate Bézier curves based on the corresponding n control points and evaluate m points at parameter v, and then, construct another Bézier curve based on the m points and evaluate the point at u.
In the process, we can find one vertex with position and its tangent in u direction. Since Bézier Surface is a sum of vertices weights, the position is ensured.
However, to find the normal of the point in surface, we have to repeat evaluate process in another diretion. With the result two vertices, do glm::cross() to find the normal.
The result is a Vertex.

2.3 Rendering a Bézier Surface

The target of this mission is to store vertex information into the class Object, load VAO, VBO and EBO, and use member function to draw corresponding graph.
I modify the function parameters in Object::initialize() to pass necessary vertices and indices to build VAO, VBO and EBO. The vertices is in vector<Vertex> form and indices is a index vertex.

After finishing these basic function, I define a self made 3×3 control points in main.cpp, and use Bézier Surface function to generate sample points. Then after dealing with the data, I pass vertices and indices into Object to draw my Bézier Surface.
The shader used is called easyshader, with color of $vec4(1.0f, 1.0f, 1.0f, 1.0f)$.

2.4 Generate tea object

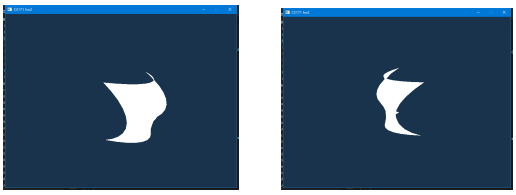
The tea object is stored in "assets/tea.bzs". With the surface number, vertex number and range of the surface, we can load it into vectors and create the corresponding vertices vector and indices vector. Since the data is continuous (different surface share the same edge), it ensures L0-th order continuity.
After generating data, we can load it into Object and draw the graph in the screen, with the same procedure as previous mission.

2.5 NURBS surface

de Boor algorithm also works for NURBS surface, with a little adjustment. NURBS requires one more dimension, which is the weight of control points, ranging from $[0, 1]$. In the code, I create a new deBoor function with parameters in vec4 and return value in vec4. The new control point sending into the function is $(x \times w, y \times w, z \times w, w)$, and we should divide the weight after deBoor result to transform it back to 3D.
To see the effect of NURBS clearly, I use the same data as surface part, with some adjustments in weight of particular points. To render the normal control point, we can set weight to 1 for every point.

3 RESULTS

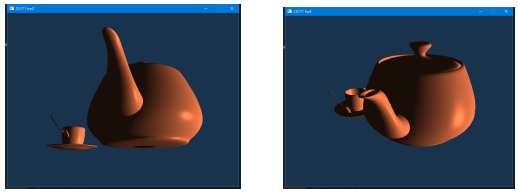
The results are pictures following.
As we can see, there is only a point in SURBS changed its weight to 0.1 and that causes a big difference in the curve that point controls.



(a) Left (b) Left

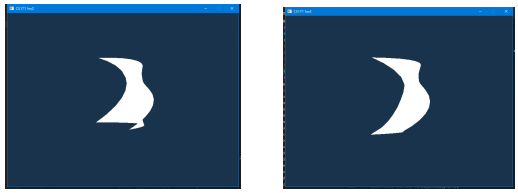
Fig. 1. Bézier Surface

student number: 2020533028
email: yinhr@shanghaitech.edu.cn



(a) Bottom (b) Head

Fig. 2. Teapot and spone



(a) Original Surface (b) Changed weight surface

Fig. 3. NURBS