Assignment #5 – Bezier surface

Yoke Kai Wen, 2020280598

April 23, 2021

1 Introduction

In this assignment, I modified the code in Tutorial 8_2 to create a Bezier surface with 5x5 control points using TCS to set subdivision level and TES to calculate new vertex coordinates and texture coordinates according to the mathematical equation of the Bezier surface. The smoothness of the surface can be controlled by adjusting the subdivision level using keyboard keys. I also bind an arbitrary texture onto the Bezier surface, and the full and wireframe mode display can be seen in Figure 1.

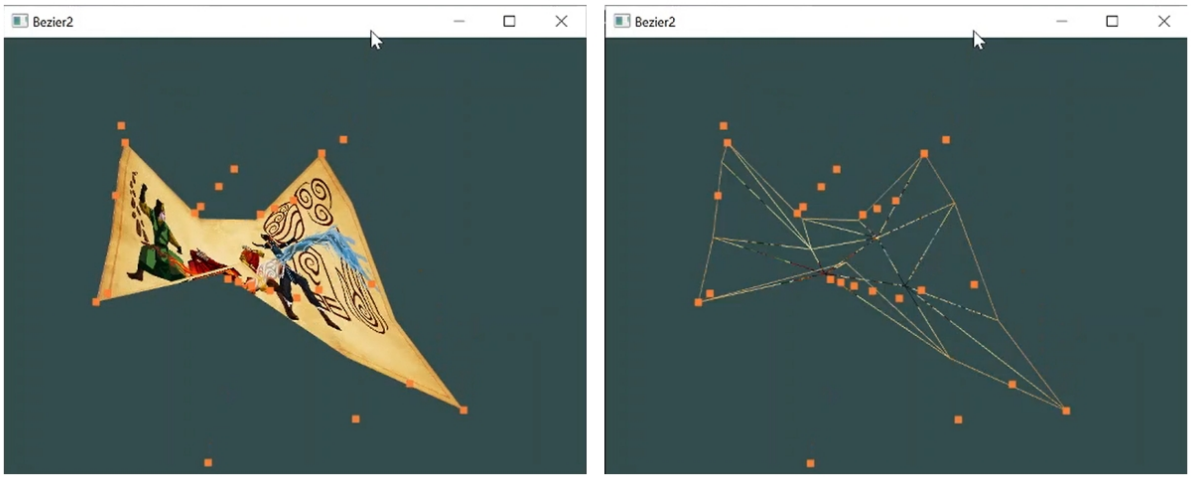

Figure 1: Bezier surface with 'avatars.jpg' texture and wireframe mode display

2 Files and directory structure

1. Main script: main.cpp
2. Shader files in shaders/: main.vert.glsl, main.frag.glsl, main.frag2.glsl, main.tcs.glsl, main.tes.glsl
3. Texture file: avatars.jpg
4. Other header files: include/tut_headers/shader.h, include/tut_headers/camera.h
5. Set include/ and lib/ folders as include and library directories respectively in VS.

3 Usage

Press 'X' to increase smoothness, 'Z' to decrease smoothness, and 'C' to switch from wireframe mode to full display mode.

4 Design

4.1 5x5 control points

I arbitrarily set 25 control points in 5 clusters as can be seen in Figure **??**. The exact coordinates of the control points are as shown below:

```
GLfloat vertices[] = {
     -1.5, 2., -4.,
    -0.5, 1., -4.,
    0.0, 0., -4.,
    0.5, 1., -4.,
    1.5, 2., -4.,

    -1.5, 2., -3.,
    -0.5, 1., -3.,
    0.0, 0., -3.,
    0.5, 1., -3.,
    1.5, 2., -3.,

    -1.5, 1., -2.,
    -0.5, -2., -2.,
    0.0, 0., -2.,
    0.5, 1., -2.,
    1.5, 0., -2.,

    -1.5, 0., -1.,
    -0.5, 1., -1.,
    0.0, 0., -1.,
    0.5, 0., -1.,
    1.5, -1., -1.,

    -1.5, 0., 0.,
    -0.5, 1., 0.,
    0.0, 0., 0.,
    0.5, -1., 0.,
    1.5, -1., 0.
};
```

4.2 Using TCS to set subdivision level

The TCS shader takes in the subdivision level for each of the four outer boundaries of the quad and as well as the inner horizontal and inner vertical subdivision levels as uniforms as seen in the shader code below. In the main code, the same subdivision level is passed to all six outer and inner boundaries and regions.

```
#version 410 core

layout( vertices = 25 ) out;

uniform float uOuter02, uOuter13, uInner0, uInner1;
```

```
void main(){
  gl_out[gl_InvocationID].gl_Position = gl_in[gl_InvocationID].gl_Position;
  // set tessellation levels
  gl_TessLevelOuter[0] = uOuter02;
    gl_TessLevelOuter[1] = uOuter13;
    gl_TessLevelOuter[2] = uOuter02;
    gl_TessLevelOuter[3] = uOuter13;
    gl_TessLevelInner[0] = uInner0;
    gl_TessLevelInner[1] = uInner1;
}
```

4.3 Using TES to calculate new vertex coordinates

The Bezier curve formula is defined as

$$C(u) = \sum_{i=0}^{n} B_{i,n}(u) p_i$$

where $B_{i,n}(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-1}$, and p_i is the set of control points.
To generalise to two dimensions for the Bezier surface, the formula is now

$$S(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_{i,n}(u) B_{j,m}(v) p_{i,j}$$

.
For this assignment, $m = n = 5$, and the Bezier surface calculations are computed as shown below in the TES shader.

```
float u = gl_TessCoord.x;
float v = gl_TessCoord.y;
TexCoord = vec2(u, v);
// Computing the Position, given a u and v
// the basis functions:
float bu0 = (1.-u) * (1.-u) * (1.-u) * (1.-u);
float bu1 = 4. * u * (1.-u) * (1.-u) * (1.-u);
float bu2 = 6. * u * u * (1.-u) * (1.-u);
float bu3 =  4. * u * u * u * (1.-u);
float bu4 = u * u * u * u;

float bv0 = (1.-v) * (1.-v) * (1.-v) * (1.-v);
float bv1 = 4. * v * (1.-v) * (1.-v) * (1.-v);
float bv2 = 6. * v * v * (1.-v) * (1.-v);
float bv3 =  4. * v * v * v * (1.-v);
float bv4 = v * v * v * v;

// finally, we get to compute something:
gl_Position = bu0 * ( bv0*p00 + bv1*p01 + bv2*p02 + bv3*p03 + bv4*p04 ) + bu1 * (
    bv0*p10 + bv1*p11 + bv2*p12 + bv3*p13 + bv4*p14 ) + bu2 * ( bv0*p20 + bv1*p21 +
     bv2*p22 + bv3*p23 + bv4*p24 ) + bu3 * ( bv0*p30 + bv1*p31 + bv2*p32 + bv3*p33
    + bv4*p34 ) + bu4 * ( bv0*p40 + bv1*p41 + bv2*p42 + bv3*p43 + bv4*p44 );
```

4.4 Change smoothness of surface by keyboard

The greater the `level` parameter, the finer the subdivision and the smoother the Bezier surface. Pressing 'X' increase `level` by 1, and pressing 'Z' decreases `level` by 1.

4.5 Switch between wireframe and full mode display by keyboard

Pressing 'C' switches between wireframe and full display mode, as seen in the code fragment below.

```
// Draw bezier surface
switch (drawMode) {
    case 0:
        glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
        break;
    case 1:
        glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
        break;
}
```

4.6 Add texture to Bezier surface

The rendering of texture is controlled by `main.frag.glsl` while `main.frag2.glsl` controls the rendering of the 25 vertices. I chose an arbitrary `avatars.jpg` image and bind it to the surface texture.