Assignment 5 report

费杰 Fedor Ivachev 2019280373

Bezier surface

1) Use 25 (5 x 5) control points

```
GLfloat vertices[] = {
    -1.5, 2.5, -3.,
    -0.5, 1., -3.,
    0.5, 1., -3.,
    1.0, 1.5, -3.,
    1.5, 2.5, -3.,

    -1.5, 1., -2.,
    -0.5, -2., -2.,
    0.5, 1., -2.,
    1.0, 1., -2.,
    1.5, 0., -2.,

    -1.5, 0., -1.,
    -0.5, 1., -1.,
    0.5, 0., -1.,
    1.0, -0.5, -1.,
    1.5, -1., -1.,

    -1.5, 0., 0.,
```

```
    -0.5, 1., 0.,
    0.5, -1., 0.,
    1.0, -0.5, 0.,
    1.5, 0., 0.,

    -1.5, 0.5, 1.,
    -0.5, 1., 1.,
    0.5, -1., 1.,
    1.0, -1., 1.,
    1.5, 0.5, 1.
};
```

2) Use TCS to set subdivision level

Two new shader stages are added, tessellation control shader:

main.tcs.glsl | main.tes.glsl | main.cpp

```
#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;
}

```

3) Use TES to calculate new vertex coordinates and texture coordinates according to the mathematical equation of Bezier surface

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

Using this formula for Bezier surface, we get the following code:

(main.tcs.glsl)

```
float bu0 = (1.-u) * (1.-u) * (1.-u) * (1.-u);
    float bu1 = 4. * u * (1.-u) * (1.-u) * (1.-u);
    float bu2 = 3. * 2. * 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 = 3. * 2. * v * v * (1.-v) * (1.-v);
    float bv3 = 4. * v * v * v * (1.-v);
    float bv4 = v * v * v * v;
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) Change smoothness of the surface by keyboard

Here level value is responsible for the smoothness. To change the value, press Z and X.

```
if (keys[GLFW_KEY_Z] && level > 1){
    if (level <= 20.0f)
        level -= deltaTime * 5.0f;
    else
        level -= deltaTime * 10.0f;
    level = level <= 1.0f ? 1.0f : level;
    std::cout << "\rLevel: " << level << "    ";
}
if (keys[GLFW_KEY_X] && level < 40){
    if (level < 20.0f)
        level += deltaTime * 5.0f;
    else
        level += deltaTime * 10.0f;
    level = level >= 40.0f ? 40.0f : level;
    std::cout << "\rLevel: " << level << "    ";
```

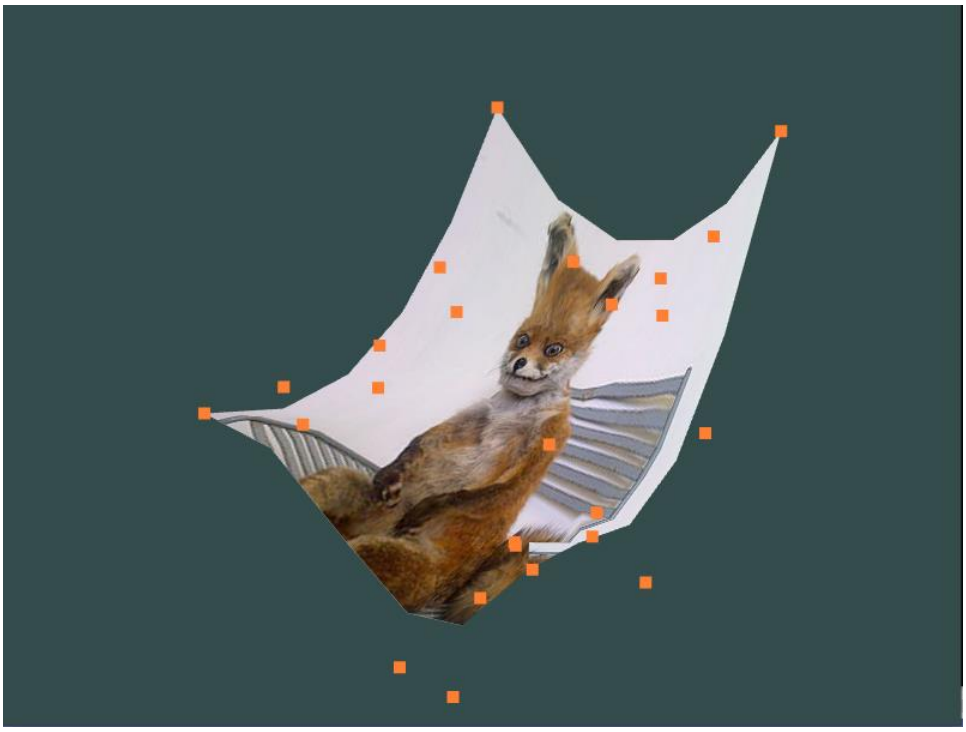
```
// Activate shader
ourShader.Use();
glUniform1f(glGetUniformLocation(ourShader.Program, "uOuter02"), level);
glUniform1f(glGetUniformLocation(ourShader.Program, "uOuter13"), level);
glUniform1f(glGetUniformLocation(ourShader.Program, "uInner0"), level);
glUniform1f(glGetUniformLocation(ourShader.Program, "uInner1"), level);
```

5) Support wireframe mode display.

To change to wireframe mode and back to original, press C. Then the GL_LINE instead of GL_FILL will be used in the drawing.

```
case 0:
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
    break;
case 1:
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
    break;
```

6) Add texture to Bezier surface. Choose the texture by yourself.



Screenshot of the running program.

Here control points are drawn by the different shader, for drawing the texture the fragment shader:

main.frag2.glsl | main.frag.glsl | main.tcs.glsl | main.tes.glsl

```
#version 410 core

in vec2 TexCoord;

out vec4 color;

uniform sampler2D ourTexture;

void main()
{
    color = texture(ourTexture, TexCoord);
    //color = vec4(1.0f, 0.5f, 0.2f, 1.0f);
}

```