HW2

Ge Tianyang
27869265

April 13, 2018

Code details and Impletement

0.1 Bezier surface construction

First of all, contruction a Bezier curve using 4 control points. According to the de Casteljau's algorithm, generate a curve. Then if we need to generate a surface, for example, a 5x5 control points mesh:

```
glm::vec3 Pu[5];
  for(int i=0;i<5;i++){
      glm::vec3 curveP[5];
      curveP[0]=controlPoints[i*5];
      curveP[1]=controlPoints[i*5+1];
      curveP[2]=controlPoints[i*5+2];
      curveP[3]=controlPoints[i*5+3];
      curveP[4]=controlPoints[i*5+4];
      Pu[i]=mlEvalBezierCurve(curveP, u);
  }
return mlEvalBezierCurve(Pu, v);
```

Then triangulation the surface. Subdivide on the parameter space, and make it satisfy the Delaunay property. However, we divide the parameter uniformly, it is easy to divide them. Calculate the normal of each points. The last recursion of calculating Bezier curve provides the tangent vector of u,v directions. Cross the vectors and get the normal.

```
for(int k=0;k<101;k++){
        for(int t=0;t<101;t++){
            Point[k][t]=mlbezier.mlEvalBezierPatch(mlbezier.
                controlPoints, k/100.0, t/100.0);
        }
    }
    for(int j=0;j<100;j++){

        P1=Point[0][1+j];
        Norm1=NormalBezier(mlbezier, 0, 0.01+j/100.0);
        P2=Point[0][j];
        Norm2=NormalBezier(mlbezier, 0,j/100.0);
        P3=Point[1][1+j];
        Norm3=NormalBezier(mlbezier, 0.01,0.01+j/100.0);
        texp1[0]=0;
        texp1[1]=0.01+j/100.0;
```

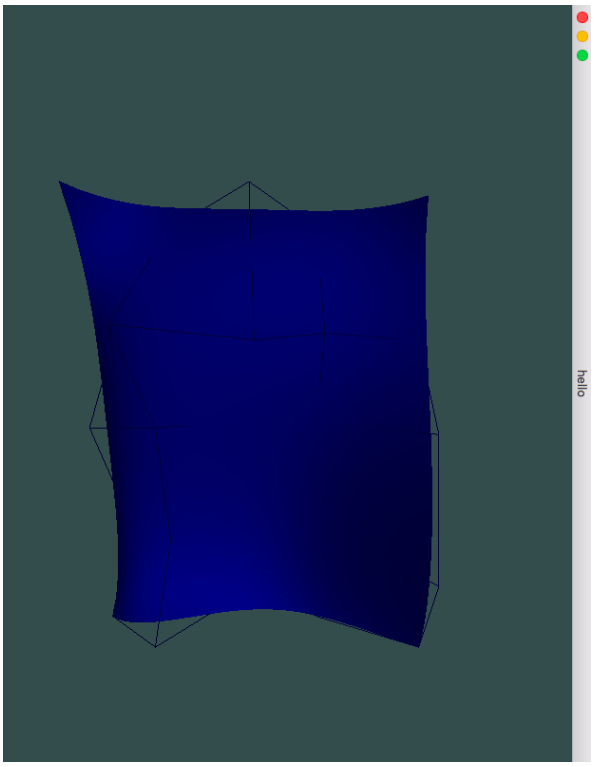
```
        texp2[0]=0;
        texp2[1]=j/100.0;
        texp3[0]=0.01;
        texp3[1]=0.01+j/100.0;
        glBegin(GL_TRIANGLES);
//        glColor3f(0.6, 0.2, 0.3);
        glTexCoord2fv(texp1);
        glNormal3f(Norm1[0], Norm1[1], Norm1[2]);
        glVertex3f(P1[0], P1[1], P1[2]);
        glTexCoord2fv(texp2);
        glNormal3f(Norm2[0], Norm2[1], Norm2[2]);
        glVertex3f(P2[0], P2[1], P2[2]);
        glTexCoord2fv(texp3);
        glNormal3f(Norm3[0], Norm3[1], Norm3[2]);
        glVertex3f(P3[0], P3[1], P3[2]);
        glEnd();

        for(int i=1;i<200;i++)
        {

            texp1[0]=texp2[0];
            texp1[1]=texp2[1];
            P1=P2;
            Norm1=Norm2;
            texp2[0]=texp3[0];
            texp2[1]=texp3[1];
            P2=P3;
            Norm2=Norm3;
            texp3[0]=(i+2-i%2)/200.0;
            texp3[1]=((i+1)%2+j)/100.0;
            P3=Point[(i+2-i%2)/2][((i+1)%2+j)];
            Norm3=NormalBezier(mlbezier, (i+2-i%2)/200.0 ,((i+1)%2+j)
                /100.0);
            glBegin(GL_TRIANGLES);
            glTexCoord2fv(texp1);
            glNormal3f(Norm1[0], Norm1[1], Norm1[2]);
            glVertex3f(P1[0], P1[1], P1[2]);
            glTexCoord2fv(texp2);
            glNormal3f(Norm2[0], Norm2[1], Norm2[2]);
            glVertex3f(P2[0], P2[1], P2[2]);
            glTexCoord2fv(texp3);
            glNormal3f(Norm3[0], Norm3[1], Norm3[2]);
            glVertex3f(P3[0], P3[1], P3[2]);
            glEnd();
        }
```

0.2 Lighting and Texture

Similar with the assignment1, set lighting. Then the image could have a lighting effection. Following the instruction, and then get the following figure. For texture part, init texture and bind it. Then, set texture parameters and filter mode. Mapping texture to the coordinate which could be map to the parameter space.




```
int width, height, nrChannels;
    stbi_set_flip_vertically_on_load(true);
//    unsigned char *data = stbi_load(("resource/textures/container.
    jpg"), &width, &height, &nrChannels, 0);
    unsigned char *data = stbi_load(("/Users/Buddha/container.jpg"), &
        width, &height, &nrChannels, 0);
    printf("char:%d,%d,%d\n",width,height,nrChannels);
    //bind it to texture
    if(data==NULL){
        printf("Error\n");
    }
    glEnable(GL_TEXTURE_2D);
    glGenTextures(1, &texture);
    glBindTexture(GL_TEXTURE_2D, texture);
    // sample: specify texture parameters
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexEnvf(GL_TEXTURE_2D,GL_TEXTURE_ENV_MODE,GL_MODULATE);
    // set the active texture
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB,
        GL_UNSIGNED_BYTE, data);
    stbi_image_free(data);
```

0.3 Mouse and keyboard control

For keyboard input, defined a processInput function to manage any of GLFW's keyboard input:

```
float cameraSpeed = 0.05f; // adjust accordingly
if (glfwGetKey(window, GLFW_KEY_W) == GLFW_PRESS)
    cameraPos += cameraSpeed * cameraFront;
if (glfwGetKey(window, GLFW_KEY_S) == GLFW_PRESS)
    cameraPos -= cameraSpeed * cameraFront;
if (glfwGetKey(window, GLFW_KEY_A) == GLFW_PRESS)
    cameraPos -= glm::normalize(glm::cross(cameraFront, cameraUp)) *
        cameraSpeed;
if (glfwGetKey(window, GLFW_KEY_D) == GLFW_PRESS)
    cameraPos += glm::normalize(glm::cross(cameraFront, cameraUp)) *
        cameraSpeed;
```

Using WASD to control directions.
Keep track of a deltatime variable that stores the time it takes to render the last frame.

```
float deltaTime = 0.0f; // Time between current frame and last frame
float lastFrame = 0.0f; // Time of last frame
float currentFrame = glfwGetTime();
deltaTime = currentFrame - lastFrame;
lastFrame = currentFrame;
```

Calculate the time between the last frame and now,and then we could get camera speed. For mouse input: There are 3 Euler angles: pitch, yaw and roll. We could get the new direction of our view by calculate ptch and yaw.

```
direction.x = cos(glm::radians(pitch)) * cos(glm::radians(yaw));
direction.y = sin(glm::radians(pitch));
direction.z = cos(glm::radians(pitch)) * sin(glm::radians(yaw));
```

We use

```
void mouse_callback(GLFWwindow* window, double xpos, double ypos);
```

to read mouse position and xpos and ypos represent the current mouse positions. And lastX and lastY to record the last position. The following part avoids strange movement of users.

```
if(pitch > 89.0f)
  pitch =  89.0f;
if(pitch < -89.0f)
  pitch = -89.0f;
```

At the last, zoom interface give a feeling of zooming in.

```
void scroll_callback(GLFWwindow* window, double xoffset, double
    yoffset)
{
  if(fov >= 1.0f && fov <= 45.0f)
    fov -= yoffset;
  if(fov <= 1.0f)
    fov = 1.0f;
  if(fov >= 45.0f)
    fov = 45.0f;
}
```