Report for Assignment 1

Adaptive tessellation of Bezier Surfaces

Jiajia Fan ---2016/10/18
fanj@shanghaitech.edu.cn

Structure

The program is designed to create a Bezier surface and traverse it with OpenGL. And to produce the surface, the De casteljau's algorithm is used to compute the coordinate of points lying on it with CMUTriangle to do triangulation. The chart below shows classes I developed in this program.

Class Name	Purpose
bezier	Produce the points and faces of Bezier surfaces
Shader	Control shader program
triangle	Do triangulation (provided by CMUTriangle)

The core class for creating the surface we want is bezier.

Detail in Core Class

bezier

structure of vertexes in Bezier:

```
typedef struct { GLfloat x, y, z; }MyVertex;
```

Variant

```
private:
    // math:C(n,i)
    int* binomial_coefficient_;
    // number of control points in a row/column (row == colomn)
    int control_length_;
    // step to sample u and v
    float sample_step_;

    std::vector<MyVertex>* control_points_;
    std::vector<MyVertex>* bezier_points_;
    int* triangulation_edges_;
    int edges_num_;
    int* ctrl_edges_;
    int ctrl_edges_num_;
```

Functions

```
void initCtrlPoints(); // initialize control points (control_length*control_length)
void initBinomialCoefficient(); // get n! (n <= control_length)
void initSamplePoints(); // initialize sample points with sample_step
// p(u, v) = sum(sum(B(u)_n_i * B(v)_m_j * k(i, j)))
// B(t)_n_i = C(n, i) * u^i * (1-u)^(n-i)
float calculateBernstein(float t,int i); // calculate B(t)_n_i
MyVertex calculateBezierPointUV(float u_in, float v_in); // calculate B(u)_n_i * B(v)_m_j * k(i, j)
void calculateBezierPoints(); // calculate p(u, v)
void calculateEdges(std::vector<MyVertex>* points_in_, bool isCtrl); // get point indices of faces

// operations for MyVertex
    // calculate multiplication of a vertex and a number
void scalarMultVertex(MyVertex& v_in, float s, MyVertex* v_out);
```

```
std::vector<MyVertex>* getBezierPoints(){ return bezier_points_; };
std::vector<MyVertex>* getControlPoints(){ return control_points_; };
int* getBezierEdges(){ return triangulation_edges_; };
int* getCtrlEdges(){ return ctrl_edges_; };
int getEdgesNum(){ return edges_num_; };
int getCtrlEdgesNum(){ return ctrl_edges_num_; };
```

Core Codes

```
int num_ = bezier_points_->size();
for (int i = 0; i < num_; i++)
{
    MyVertex temp = calculateBezierPointUV(bezier_points_->at(i).x, bezier_points_->at(i).y);
    bezier_points_->at(i) = temp;
}
```

```
MyVertex bezier::calculateBezierPointUV(float u_in, float v_in){
    MyVertex bezier_point_uv = { 0.0f, 0.0f, 0.0f };
    for (int i = 0; i < control_length_; i++)
    {
        for (int j = 0; j < control_length_; j++)
        {
            float ratio_bernstein = calculateBernstein(u_in, i)*calculateBernstein(v_in, j);
            scalarMultVertex(control_points_->at(i*(control_length_) + j), ratio_bernstein, &bezier_point_uv);
        }
    }
    return bezier_point_uv;
}
```

```
float bezier::calculateBernstein(float t, int i){
    float b_out = 1.0f;
    float t_conv = 1.0f - t;
    // B(t)^i = C(n,i)*t^i*(1-t)^(n-i)
    b_out *= (float)binomial_coefficient_[i];
    b_out *= pow(t, i);
    b_out *= pow(t_conv, control_length_-1-i);

    return b_out;
}
```

Algorithms

The process of producing a Bezier surface can be divided into 3 parts listing below:

1. Initialization of control points

   Control points are initialized into grid of n*n, lying on a plane. Then we can use CMUTriangle to connect the points. Next these points' coordinates can be determined by users.

2. Sampling and creating connections between points

   Since the control points are determined, we can do the sampling with properties of those control points. Before that, triangulation can be done by CMUTriangle.

3. Calculate coordinates of sample points

   The equation following is used to calculate those sample points.

$$\mathbf{p}(u,v)=\sum_{i}^{n}\sum_{j}^{m}B_i^n(u)\,B_j^m(v)\,\mathbf{k}_{i,j}$$

Operating Instructions

Keyboard

↑	↓	←	→	Space
Translate up	Translate down	Translate left	Translate right	Rotation On/Off

Mouse

Scroll up	Scroll down
Zoom in	Zoom out

Example Images