计算机图形学 Homework 8

吴宇祺 16340242

Homework

Basic:

1. 用户能通过左键点击添加Bezier曲线的控制点，右键点击则对当前添加的最后一个控制点进行消除
2. 工具根据鼠标绘制的控制点实时更新Bezier曲线。 Hint: 大家可查询捕捉mouse移动和点击的函数方法

Bonus:

3. 可以动态地呈现Bezier曲线的生成过程。

Solution

捕捉鼠标的移动和点击的函数

OpenGL提供了`glfwSetMouseButtonCallback`和`glfwSetCursorPosCallback`作为鼠标点击事件和移动事件的回调函数，前者检测鼠标的动作，后者可以实时获得光标的位置（屏幕坐标）：

```
void cursor_position_callback(GLFWwindow* window, double x, double y)
{
    xpos = x;
    ypos = y;
    return;
}

void mouse_button_callback(GLFWwindow* window, int button, int action, int mods)
{
    if (action == GLFW_PRESS)
    {
        switch (button)
        {
        case GLFW_MOUSE_BUTTON_LEFT:
            cout << "Mouse left button clicked at " << xpos << " " << ypos << endl;
            clickLeft = true;
            break;

        case GLFW_MOUSE_BUTTON_RIGHT:
            cout << "Mouse right button clicked at" << xpos << " " << ypos << endl;
            clickRight = true;
            break;

        default:
            return;
        }
    }
```

```
28
29      return;
30  }
```

在初始化窗口时注册回调函数：

```
1  //GLFWwindow* initWindow(const char* title, int SCR_WIDTH, int SCR_HEIGHT)
2  glfwSetMouseButtonCallback(window, mouse_button_callback);
   glfwSetCursorPosCallback(window, cursor_position_callback);
```

捕捉控制点位置并输出到屏幕

`myBesierCurve`类用于存储

```
1  //myBesierCurve 部分数据成员
2      vector<Point> controlPoints; //控制点的屏幕坐标
3      int nControlPoint;           //控制点数
4
5      Shader* shader;              //着色器
6
7      GLFWwindow* window;          //窗口指针
8
9      float data[3072];            //传入着色器的数据
10     int nTotalPoint;
```

通过以上回调函数，检测鼠标点击并改变系统中全局布尔变量`clickRight`和`clickLeft`间点击信息和位置`xpos`、`ypos`传递给`myBesierCurve`对象。

在render函数中检测这两个变量并作出相应的处理：

```
1  //myBesierCurve::render()
2  if (clickRight == true)
3      delControlPoint();
4
5  if (clickLeft == true)
6      addControlPoint();
7
8  clickLeft = clickRight = false;
9
10 point2data(controlPoints);
11
12 drawPoints(10, glm::vec3(1.0f, 1.0f, 1.0f), true);//点的大小，颜色，是否画线
```

其中，`point2data`函数用于将控制点向量controlPoints 转换为float数组data，期间进行屏幕坐标到标准化坐标的变换：

```cpp
bool myBesierCurve::point2data(vector<Point>& p)
{
    nTotalPoint = p.size();
    for (int i = 0; i < nTotalPoint;i++)
    {
        data[i * 3] = (p[i].x / SCR_WIDTH) * 2 - 1;
        data[i * 3 + 1] = -((p[i].y / SCR_HEIGHT) * 2 - 1);
        data[i * 3 + 2] = 0;
    }

    return true;
}
```

采样并绘制Bezier曲线

伯恩斯坦（Bernstein）基函数的公式为

$$B_{i,n}(t) = \frac{n!}{i!(n-i)!} * (1-t)^{n-i}t^i$$

Bezier曲线的公式为

$$Q(t) = \sum_{i=0}^{n} P_i B_{i,n}(t), t \in [0,1]$$

计算阶乘

计算基函数需要用到大量的阶乘。为了避免重复计算，设置 `factorial` 数组存储目前已经计算过的阶乘，`maxFactorial` 记录当前计算过的最大阶乘：

```cpp
long long int myBesierCurve::computeFactorial(int end)
{
    factorial[0] = 1;
    for (int i = 1; i < end+1; i++)
        factorial[i] = factorial[i - 1] * i;
    maxFactorial = end;
    return factorial[end];
}
long long int myBesierCurve::getFactorial(int index)
{
    return ((index <= maxFactorial)? this->factorial[index]:computeFactorial(index));
}
```

计算Bernstein基函数

```
float myBesierCurve::bernsteinItem(int i, float t)
{
    int n = nControlPoint - 1;
    float tmp = float(getFactorial(n)) / float((getFactorial(i) * getFactorial(n - i)));
    return (tmp * pow(1 - t, n -  i) * pow(t, i));
}
```

计算采样点

```
void myBesierCurve::sampleBesierCurve(float interval, vector<Point>& sample)
{
    sample.clear();
    float x = 0, y = 0;

    for (float t = 0; t <= 1; t += interval)
    {
        x = controlPoints[0].x * bernsteinItem(0, t);
        y = controlPoints[0].y * bernsteinItem(0, t);
        for (int i = 1; i < nControlPoint; i++)
        {
            x += controlPoints[i].x * bernsteinItem(i, t);
            y += controlPoints[i].y * bernsteinItem(i, t);
        }
        sample.push_back(Point(x, y));
    }
}
```

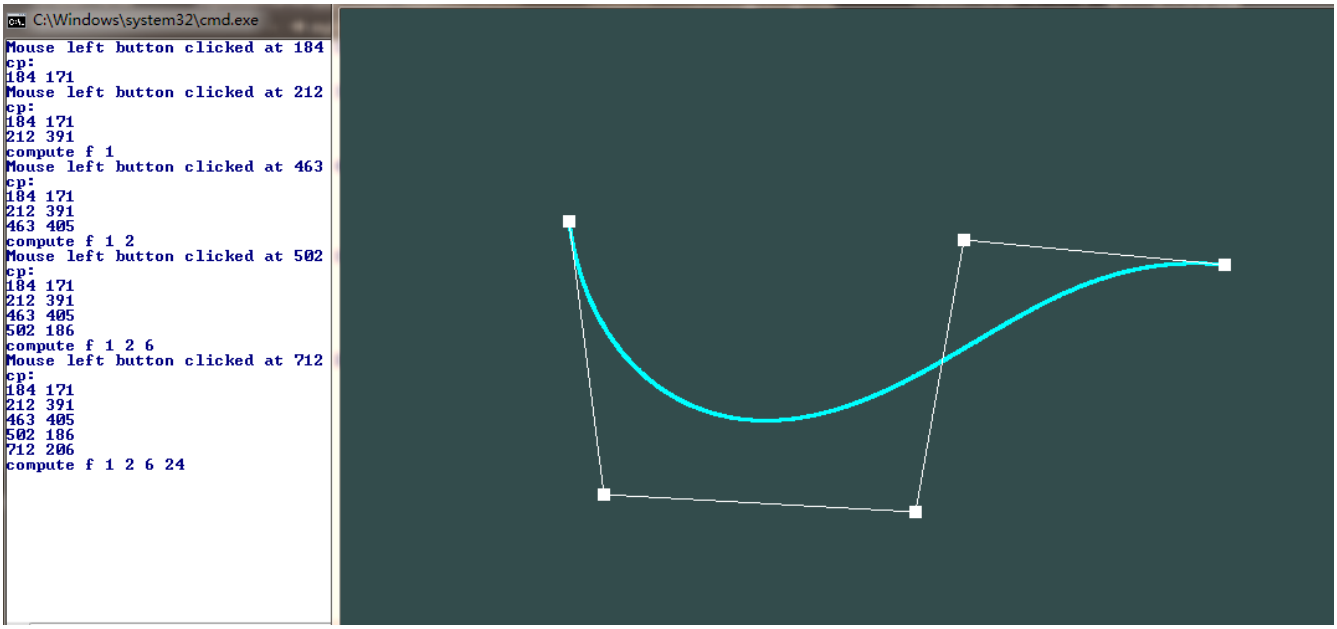
绘制曲线

```
//myBesierCurve::render()
vector<Point> sample;
if (nControlPoint > 1)
{
    sampleBesierCurve(0.001, sample); //采样贝塞尔曲线
    point2data(sample);               //将采样点向量转换为float数组
    drawPoints(3, glm::vec3(0.0f, 1.0f, 1.0f));
}
```

结果

GIF：Basic.gif

图示：


C:\Windows\system32\cmd.exe
Mouse left button clicked at 184
cp:
184 171
Mouse left button clicked at 212
cp:
184 171
212 391
compute f 1
Mouse left button clicked at 463
cp:
184 171
212 391
463 405
compute f 1 2
Mouse left button clicked at 502
cp:
184 171
212 391
463 405
502 186
compute f 1 2 6
Mouse left button clicked at 712
cp:
184 171
212 391
463 405
502 186
712 206
compute f 1 2 6 24