计算机图形学 作业8

徐达烽

16340260

效果见Bezier Curve.mp4视频。

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。

实现思路

1. 实现一个鼠标点击的回调函数：

```
glfwSetMouseButtonCallback(window, mouse_button_callback);
...
void mouse_button_callback(GLFWwindow* window, int button, int action, int mods) {
    if (whetherControlColor) {
        return;
    }
    if (GLFW_MOUSE_BUTTON_LEFT == button && GLFW_PRESS == action) {
        double xpos, ypos;
        glfwGetCursorPos(window, &xpos, &ypos);
        curve.setPoint(xpos / display_w, ypos / display_h);
        whetherDrop = true;
    }
    else if (GLFW_MOUSE_BUTTON_RIGHT == button && GLFW_PRESS == action) {
        curve.deletePoint();
    }
}
```

其中curve是封装好的Curve对象，setPoint方法加入一个点，deletePoint方法删除一个点。

1. 坐标变换，将屏幕坐标变换到openGL中的坐标系统：

```
glm::vec2 Curve::coorTransform(const double & x, const double & y) {
    return glm::vec2(2 * x - 1.0, -2 * y + 1);
}
```

1. 生成曲线：使用Bezier曲线的公式生成曲线上的1000个点：

```
void Curve::calculateCurve() {
    if (points.size() <= 1) return;
    for (unsigned int i = 0; i < T_NUMBER; ++i) {
        GLfloat t = i * T_GAP;
        curve[i] = glm::vec2(0, 0);
        float co = 1;
        int n = points.size() - 1;
        for (int j = 0; j <= n; j++) {
            if (j > 0) {
                co *= n - j + 1;
                co /= j;
            }
            curve[i] += co * pow(t, j) * pow(1-t,n-j) * points[j];
        }
    }
}
```

其中变量co是二项式系数。

1. 绘图，实现一个drawPoints函数，表示将一个数组中的点按给定的size和color绘制出来.

```
void Curve::drawPoints(const vector<glm::vec2> &points, GLfloat psize, glm::vec3 color, bool line = false) {
    glPointSize(psize);
    this->setColorVec3(color);
    GLfloat *vertices = new GLfloat[points.size() * 2];
    GLuint *indices = new GLuint[points.size()];
    for (unsigned int i = 0; i < points.size(); ++i) {
        vertices[i * 2] = points[i].x;
        vertices[i * 2 + 1] = points[i].y;
        indices[i] = i;
    }
    glBufferData(GL_ARRAY_BUFFER, sizeof(GLfloat) * 2 * points.size(), vertices, GL_STREAM_DRAW);
    glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(GLuint)*points.size(),
indices, GL_STREAM_DRAW);
    glDrawElements(GL_POINTS, points.size(), GL_UNSIGNED_INT, 0);
    delete[] indices;
    if (points.size() >= 2 && line) {
        glPointSize(1.0f);
        this->setColorVec3(glm::vec3(1.0f));
        indices = new GLuint[(points.size() - 1) * 2];
        for (int i = 0; i < points.size() - 1; i++) {
            indices[i * 2] = i;
            indices[i * 2 + 1] = i + 1;
        }
        glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(GLuint)*2*(points.size()-1), indices, GL_STREAM_DRAW);
        glDrawElements(GL_LINES, 2 * (points.size() - 1), GL_UNSIGNED_INT, 0);
        delete[] indices;
    }
    delete[] vertices;
}
```

函数的参数line布尔值为真，则将这个数组的点连成的折现画出来，使用EBO就不需要重新产生一份VBO了。

Bonus

可以动态地呈现Bezier曲线的生成过程。

实现思路

实现函数DrawProcess(float t), 参数t是随时间变化的参数，取值为[0,1]

```
void Curve::DrawProcess(float t) {
    vector<glm::vec2> points = this->points;
    while (points.size() > 1) {
        vector<glm::vec2> next_points;
        for (size_t i = 0; i + 1 < points.size(); i++) {
            next_points.push_back((1 - t)*points[i] + t * points[i + 1]);
        }
        glm::vec3 color(0.8f, 0.9f, 0.3f);
```

```
            if (next_points.size() == 1) color = curveColor;
            drawPoints(next_points, 10.0f, color, true);
            points = next_points;
        }
    }
```