科目	学号	姓名	邮箱
计算机图形学	16340294	张星	dukecheung@foxmail.com

Basic

题目一

实验要求

> 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。

实验思路

- 捕获鼠标点击事件：通过查阅资料，通过设置回调函数即可：

  ```
  glfwSetMouseButtonCallback(window, mouseCallback);
  ```

  其中函数参数为：action为点击，button区分左右键即可。vertices为坐标数组，pointNum为画点/线的数目，点击左键，添加坐标，点击右键，pointNum减一，不用修改vertices数组，因为下次添加时会自动覆盖掉。

  ```
  void mouseCallback(GLFWwindow* window, int button, int action, int mods) {
      if (action == GLFW_PRESS) {
          switch (button) {
          case GLFW_MOUSE_BUTTON_LEFT:
              vertices[pointNum * 2] = currentX / (float)(width/2);
              vertices[pointNum * 2 + 1] = currentY / (float)(height / 2);
              pointNum++;
              break;
          case GLFW_MOUSE_BUTTON_RIGHT:
              if (pointNum > 0) {
                  pointNum--;
              }
              break;
          default:
              break;
          }
      }
  }
  ```

- 上面的函数只能捕获点击，但是在作业中需要获取屏幕坐标，所以需要光标的位置，通过以下函数设置：

  ```
  glfwSetCursorPosCallback(window, cursorCallback);
  ```

  其中 *cursorCallback* 函数获取的坐标是以窗口左上为坐标原点，但屏幕坐标的原点则是在正中央，所以需要做一些变换：

```
void cursorCallback(GLFWwindow* window, double x, double y) {
    currentX = x - width / 2;
    currentY = -y + height / 2;
}
```

- 添加控制点：使用GL_POINTS即可，但是需要设置点的大小，显得明显一些。

```
void renderPoint(unsigned int VAO) {
    glBindVertexArray(VAO);
    glPointSize(10.0f);
    glDrawArrays(GL_POINTS, 0, pointNum);
}
```

- 绘制直线，将每个控制点连接起来，需要注意的是，使用了GL_LINE_STRIP参数，与GL_LINE参数不同的是，前者会将每段线段连接起来，符合我们的期望。

```
void renderLine(unsigned int VAO) {
    glBindVertexArray(VAO);
    glDrawArrays(GL_LINE_STRIP, 0, pointNum);
}
```

- 绘制Bezier曲线：通过老师上课讲解，绘制曲线的简单原理就是不停地划分，将多边形最后按照t参数，每条边上取一个点，连接起来，直至最后连接成一条线，再取该线段上的t点，即从左端点占长度t的一个点，然后迭代t，将前后连个t点连接起来，如此直至t=1，就可以绘制Bezier曲线了。

  所以在此次实验中，我使用了一个包含了两个定点数据的数组，每次使用它来绘制线段，直至t为1，这样就绘制了Bezier曲线。首先，将左端点赋值给该数组：

```
qVertices[0] = vertices[0];
qVertices[1] = vertices[1];
```

  然后开始进行t迭代，这里可以任意指定t递增的速度，越小曲线越细致，但也不能太小，人眼无法分辨出，并且浪费开销。

```
for (float t = 0.0f; t < 1.0f; t += 0.02f)
```

  然后根据公式计算右端点的位置：*binomialCoef* 是我自己定义的一个计算二项式参数的函数，需要注意的是，老师所给公式中的n是顶点数目-1，而此时的pointNum则是顶点数目，所以需要-1，一开始我没有注意到这一点，绘制出来的曲线一直不对，所以这是一个重点。

```
for (int i = 0; i < pointNum; i++) {
    qVertices[2] += vertices[i * 2] * binomialCoef(pointNum - 1, i)*pow(t, i)*pow((1 -
t), pointNum - 1 - i);
    qVertices[3] += vertices[i * 2 + 1] * binomialCoef(pointNum - 1, i)*pow(t,
i)*pow((1 - t), pointNum - 1 - i);
}
```

  每次绘制完一段之后，更新数组，将右端点赋值给左端点，然后开始下一次循环：

```
qVertices[0] = qVertices[2];
qVertices[1] = qVertices[3];
qVertices[2] = 0.0f;
qVertices[3] = 0.0f;
```

实验结果

详见result.gif。

Bonus

题目一

实验要求

> 1. 可以动态地呈现Bezier曲线的生成过程。

实验思路

上面使用了二项式，而这道题目则是需要写出中间过程，是一个树形的结构，先使用最外面的顶点，计算出下一层顶点的坐标，然后根据计算出的这一层顶点，算出下一层顶点，直至顶点数目为二即可，每一层比上层顶点数目少一。按道理需要计算到最后一个顶点才可，但是我们已经用了二项式来计算出了曲线顶点坐标，所以只需算出中间过程即可。顶点计算原理如下：

$$P = (1-t)P_0 + tP_1$$

我使用了数组来存储点的数目，每层循环绘制点的数目-1，即可达到效果：

```
for (int i = 0; i < 200; i++) {
    bonusVertices[i] = vertices[i];
}
for (float t = 0.0f; t < 1.0f; t += 0.02f) {
    for (int i = pointNum; i > 2; i--) {
        for (int j = 0; j < i - 1; j++) {
            bonusVertices[j * 2] = bonusVertices[j * 2] * (1 - t) + bonusVertices[(j + 1) *
2] * t;
            bonusVertices[j * 2 + 1] = bonusVertices[j * 2 + 1] * (1 - t) +
bonusVertices[(j + 1) * 2 + 1] * t;
        }
        glBindVertexArray(qVAO);
        glDrawArrays(GL_LINE_STRIP, 0, i);
    }
}
```

实验结果

详见result.gif。