计算机图形学作业报告

廖蕾 16340135

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。

答：

- Bezier曲线绘制方程和算法： 这里使用三次Bezier方程去计算的的各个点。所以在实现的过程中，采用了每四个点绘制一次曲线的方法。根据三次Bezier曲线方程，可以得到函数如下：

  ```
  vector<glm::vec3> Bezier(glm::vec3 point0, glm::vec3 point1, glm::vec3
  point3, glm::vec3 point4) {
      vector<glm::vec3> result;
      for (float t = 0; t <= 1; t += 0.001) {
          float f0 = t * t;
          float f1 = f0 * t;
          float f2 = (1 - t);
          float f3 = f2 * (1 - t);
          float f4 = f3 * (1 - t);
          result.push_back(point0 * f4 + 3 * t * f3 * point1 + 3 * f0
  * f2 * point3 + point4 * f1);
      }
      return result;
  }
  ```

  算法的基本思想是，迭代t，然后在每次迭代中计算每个点需要乘的参数，然后计算最后结果，并放入曲线上点集的vector中。

- 鼠标控制输入：
  这里实现了一个`click_callback`的函数，在其中定义了四种点击事件，有左键点击按住、左键点击松开、点击完成、右键点击。
  这样的函数和需要的一些变量如下：

  ```
  int cursorPosX = 0;
  int cursorPosY = 0;
  int control_points_num = 0;
  vector<glm::vec3> controlPoint;
  bool holding = false;
  int closestIndex = 0;
  int linesIndex = 0;
  glm::vec3 standardize(int x, int y) {
  glm::vec3 result = glm::vec3((float(x) / float(WINDOW_WIDTH)*2.0) - 1, -
  ```

````
   ((float(y) / float(WINDOW_HEIGHT) * 2) - 1), 0.0f);
   return result;
````

}bool addFlag = true;

~~~
void click_callback(GLFWwindow* window, int button, int action, int mods) {
        bool isHovered = ImGui::IsWindowHovered(ImGuiHoveredFlags_AnyWindow);
        glm::vec3 clickPos = standardize(cursorPosX, cursorPosY);
        if (button == GLFW_MOUSE_BUTTON_LEFT && action == GLFW_PRESS) {
                holding = true;
        }
        if (button == GLFW_MOUSE_BUTTON_LEFT && action == GLFW_RELEASE) {
                holding = false;
        }
        if (button == GLFW_MOUSE_BUTTON_LEFT && action == GLFW_PRESS &&
!isHovered) {
                controlPoint.push_back(standardize(cursorPosX, cursorPosY));
                control_points_num++;
                if (addFlag) {
                        addFlag = false;
                }
                else {
                        closestIndex++;
                }
        }
        if (button == GLFW_MOUSE_BUTTON_RIGHT && action == GLFW_PRESS &&
control_points_num > 0 && !isHovered) {
                controlPoint.pop_back();
                control_points_num--;
        }
}
```

其中获取```cursorPosX```和```cursorPosY```的函数是通过一个```cursor_pos_callback```
获取的，函数如下：

```C++
void cursor_pos_callback(GLFWwindow* window, double x, double y) {
        bool isHovered = ImGui::IsWindowHovered(ImGuiHoveredFlags_AnyWindow);
        cursorPosX = x;
        cursorPosY = y;
        if (holding & control_points_num >= 4 && !isHovered) {
                glm::vec3 clickPos = standardize(cursorPosX, cursorPosY);
                controlPoint[closestIndex] = clickPos;
        }
}
```

这里使用一个```standardize```函数去标准化我们获取到的x和y坐标，然后获取```glm::vec3```
~~~

~~~
这种类型的点的坐标，函数如下：

```C++
glm::vec3 standardize(int x, int y) {
        glm::vec3 result = glm::vec3((float(x) / float(WINDOW_WIDTH)*2.0) - 1, -
((float(y) / float(WINDOW_HEIGHT) * 2) - 1), 0.0f);
        return result;
}
```

在主函数中，定义这些callback函数就可以用了：
```C++
glfwSetCursorPosCallback(window, cursor_pos_callback);
glfwSetMouseButtonCallback(window, click_callback);
```
~~~

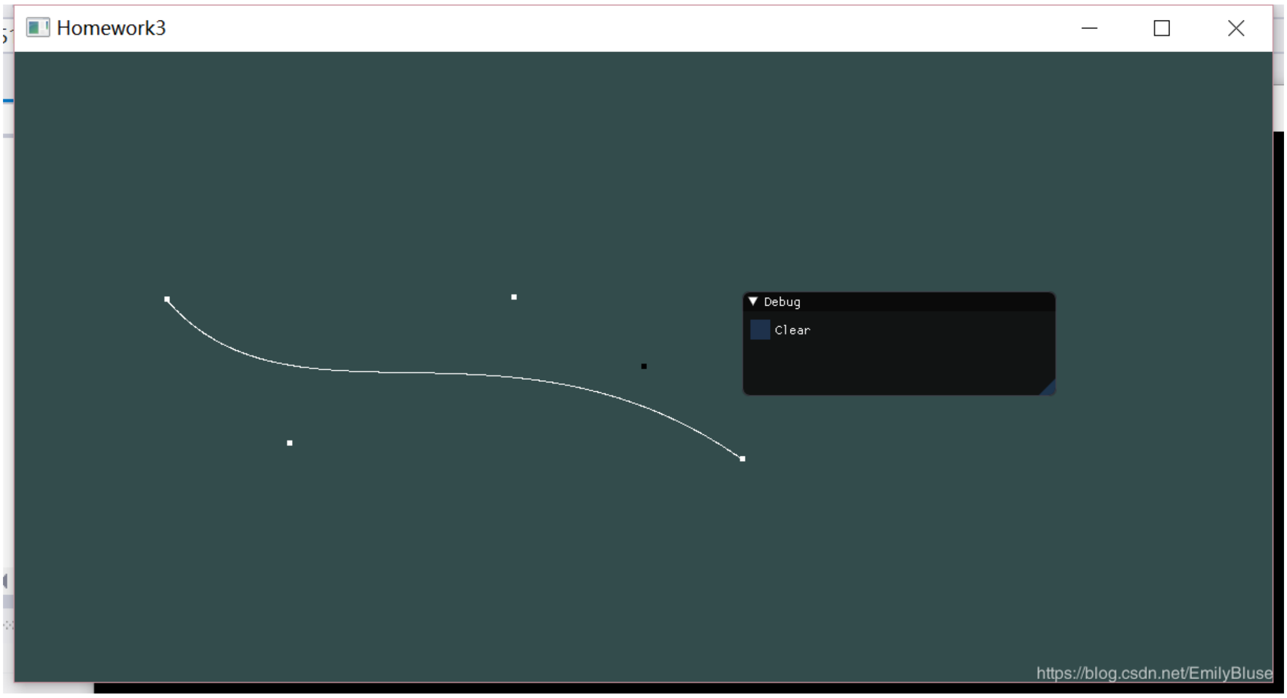
- 渲染结果：
  在渲染的时候，就是开始调用这些函数，然后每次渲染点击的点，每有四个点的时候，渲染生成的曲线就好了，函数如下：

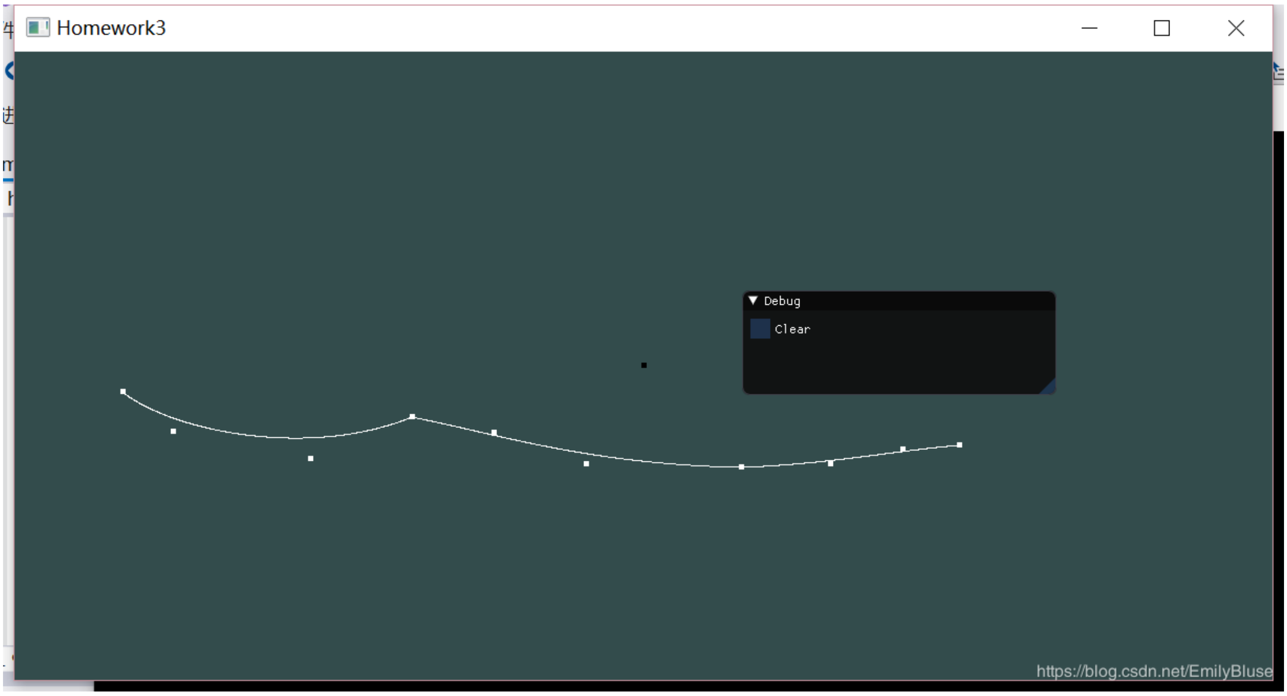
```
while (!glfwWindowShouldClose(window))
{
        // input
        // -----
        processInput(window);
        glfwPollEvents();
        // render
        // ------
        ImGui_ImplGlfwGL3_NewFrame();
        glClearColor(0.2f, 0.3f, 0.3f, 1.0f);
        glClear(GL_COLOR_BUFFER_BIT); // also clear the depth buffer now

        vector<vector<float> > triangle;

        {
            //IMGUI
                ImGui::Checkbox("Clear", &onClear);
                ImGui::SetWindowSize(ImVec2(300, 100));
                //清除全部的点和曲线
                if (onClear) {
                        control_points_num = 0;
                        controlPoint.clear();
                        onClear = false;
                }
                int len = controlPoint.size();
                //渲染曲线部分
                if (len >= 4) {
                        if (len == 4) {
                                curve = Bezier(controlPoint[len - 4],
controlPoint[len - 3], controlPoint[len - 2], controlPoint[len - 1]);
```

```
            }
            if ((len - 4) % 3 == 0) {
                vector<glm::vec3> temppoints =
Bezier(controlPoint[len - 4], controlPoint[len - 3], controlPoint[len - 2],
controlPoint[len - 1]);
                curve.insert(curve.end(),
temppoints.begin(), temppoints.end());
            }
            for (size_t i = 0; i < curve.size(); i++) {
                float point[] = { curve[i].x, curve[i].y,
curve[i].z, color[0], color[1], color[2] };
                glBindVertexArray(VAO);
                glBufferData(GL_ARRAY_BUFFER, sizeof(point),
point, GL_STATIC_DRAW);
                // position
                glVertexAttribPointer(0, 3, GL_FLOAT,
GL_FALSE, 6 * sizeof(float), (void*)0);
                glEnableVertexAttribArray(0);
                //color
                glVertexAttribPointer(1, 3, GL_FLOAT,
GL_FALSE, 6 * sizeof(float), (void*)(3 * sizeof(float)));
                glEnableVertexAttribArray(1);
                glBindBuffer(GL_ARRAY_BUFFER, VBO);
                glBufferData(GL_ARRAY_BUFFER, sizeof(point),
point, GL_STATIC_DRAW);
                glPointSize(1.0f);
                glDrawArrays(GL_POINTS, 0, 1);
            }
        }
        //渲染关键点击点的部分
        for (size_t i = 0; i < controlPoint.size(); i++) {
            float point[] = {
             controlPoint[i].x, controlPoint[i].y,
controlPoint[i].z, color[0], color[1], color[2],
            };
            glBindVertexArray(VAO);
            glBufferData(GL_ARRAY_BUFFER, sizeof(point), point,
GL_STATIC_DRAW);
            // position
            glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 6 *
sizeof(float), (void*)0);
            glEnableVertexAttribArray(0);
            //color
            glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 6 *
sizeof(float), (void*)(3 * sizeof(float)));
            glEnableVertexAttribArray(1);
            glBindBuffer(GL_ARRAY_BUFFER, VBO);
            glBufferData(GL_ARRAY_BUFFER, sizeof(point), point,
GL_STATIC_DRAW);
            glPointSize(5.0f);
            glDrawArrays(GL_POINTS, 0, 4);
        }

    }
```

```
        // glfw: swap buffers and poll IO events (keys pressed/released, mouse moved etc.)
        // -------------------------------------------------------------------------------
        ImGui::Render();
        ImGui_ImplGlfwGL3_RenderDrawData(ImGui::GetDrawData());
        glfwSwapBuffers(window);
        glfwPollEvents();
}
```

这样这部分的代码就写好了，展示的mp4见附件中，这里放出一张实现图片如下：


Homework3
Debug
Clear
https://blog.csdn.net/EmilyBluse


Homework3
Debug
Clear
https://blog.csdn.net/EmilyBluse