R/ E/ P/ O/ R/ T

Self-scoring table :

	report	video	add	remove	Drag	Insert	Total
Score	1/1	1/1	1/1	1/1	1/1	1/1	6/6

제출일: 20. 05. 22

과제명: Programming Assignment #1

과목 : 컴퓨터애니메이션

학과 : 소프트웨어 학부

학번 : 2015726076

이름 : 김현구

1. Mouse Click

먼저 마우스 클릭을 인식하기 위해 마우스의 좌표를 가져와서 캔버스와 대응시켜준다.

```
354   void mouse(GLFWwindow* window, int button, int action, int mods)
355   {
356       if (button == GLFW_MOUSE_BUTTON_LEFT)
357       {
358           if (action == GLFW_PRESS)
359           {
360               //find cursor
361               double xpos, ypos;
362               glfwGetCursorPos(window, &xpos, &ypos);
363               ee_goal = Vector3f(xpos, ypos, 0);
364
365               float aspect = (float)screenW / screenH;
366               ee_goal[0] = 2.0 * (ee_goal[0] / screenW - 0.5) * aspect;
367               ee_goal[1] = -2.0 * (ee_goal[1] / screenH - 0.5);
```

glfwGetCursorPos()함수를 통해 마우스의 좌표를 가져온 후, 캔버스의 비율에 맞추어 변환해준다.

2. Add

점은 A 키를 누르고 마우스를 클릭했을 때 그 위치에 생긴다.

```
179   void keyboard(GLFWwindow* window, int key, int scancode, int action, int mods)
180   {
181       if (action == GLFW_PRESS || action == GLFW_REPEAT)
182       {
183           switch (key)
184           {
185               //Quit
186           case GLFW_KEY_Q:
187           case GLFW_KEY_ESCAPE: glfwSetWindowShouldClose(window, GL_TRUE); break;
188           case GLFW_KEY_A: condition = 1; cout << "Add Point" << endl; break;
189           case GLFW_KEY_R: condition = 2; cout << "select/remove" << endl; break;
190           case GLFW_KEY_D: condition = 3; cout << "select/drag" << endl; break;
191           case GLFW_KEY_I: condition = 4; cout << "select/insert" << endl; break;
192           }
193       }
194   }
```

```
//a key
case 1:
{
    p.push_back(ee_goal);
    break;
}
```

점의 좌표는 vector<Vector3f> p 에 저장된다.

3. Draw Curve

점이 2 개 이상일 때부터 곡선을 그린다.

```
196    void setPoint()
197    {
198        if (p.size() > 1)
199        {
200            buildLinearSystem();
201            solveLinearSystem();
202        }
203    }
```

```
if (p.size() > 1)
    drawNaturalCubicSpline();
```

Spline Curve 의 Natural 조건으로 삼차식을 구하고

```
65     // 2n
66     for (int i = 0; i < p.size() - 1; i++, row += 2)
67     {
68         A(row, 4 * i + 0) = 1;
69
70         b(row, 0) = p[i][0]; //x
71         b(row, 1) = p[i][1]; //y
72         b(row, 2) = p[i][2]; //z
73
74         A(row + 1, 4 * i + 0) = 1;
75         A(row + 1, 4 * i + 1) = 1;
76         A(row + 1, 4 * i + 2) = 1;
77         A(row + 1, 4 * i + 3) = 1;
78
79         b(row + 1, 0) = p[i + 1][0];
80         b(row + 1, 1) = p[i + 1][1];
81         b(row + 1, 2) = p[i + 1][2];
82     }
```

```
84     // n-1 tangential
85     for (int i = 0; i < p.size() - 2; i++, row++)
86     {
87         A(row, 4 * i + 1) = 1;
88         A(row, 4 * i + 2) = 2;
89         A(row, 4 * i + 3) = 3;
90         A(row, 4 * i + 5) = -1;
91
92         b(row, 0) = 0;
93         b(row, 1) = 0;
94         b(row, 2) = 0;
95     }
```

```
97     // n-1 second-derivative
98     for (int i = 0; i < p.size() - 2; i++, row++)
99     {
100        A(row, 4 * i + 2) = 2;
101        A(row, 4 * i + 3) = 6;
102        A(row, 4 * i + 6) = -2;
103
104        b(row, 0) = 0;
105        b(row, 1) = 0;
106        b(row, 2) = 0;
107    }
```

```
109    // 2 for natural boundary condition
110    {
111        A(row, 2) = 2;
112
113        b(row, 0) = 0;
114        b(row, 1) = 0;
115        b(row, 2) = 0;
116
117        row++;
118
119        A(row, 4 * (p.size() - 2) + 2) = 2;
120        A(row, 4 * (p.size() - 2) + 3) = 6;
121
122        b(row, 0) = 0;
123        b(row, 1) = 0;
124        b(row, 2) = 0;
125
126        row++;
127    }
```

구간 사이를 여러 개의 segment 로 나누어 곡선을 표현한다.

```
135  void drawNaturalCubicSpline()
136  {
137      int N_SUB_SEGMENTS = 40;
138
139      //segment curve
140      glLineWidth(1.5*dpiScaling);
141      glColor3f(0, 0, 0);
142      for (int i = 0; i < p.size(); i++)
143      {
144          glBegin(GL_LINE_STRIP);
145          for (int j = 0; j < N_SUB_SEGMENTS; j++)
146          {
147              float t = (float)j / (N_SUB_SEGMENTS - 1);
148
149              float x = c(4 * i, 0) + (c(4 * i + 1, 0) + (c(4 * i + 2, 0) + c(4 * i + 3, 0)*t)*t)*t;
150              float y = c(4 * i, 1) + (c(4 * i + 1, 1) + (c(4 * i + 2, 1) + c(4 * i + 3, 1)*t)*t)*t;
151              float z = c(4 * i, 2) + (c(4 * i + 1, 2) + (c(4 * i + 2, 2) + c(4 * i + 3, 2)*t)*t)*t;
152
153              glVertex3f(x, y, z);
154          }
155          glEnd();
156      }
157
158      //point
159      glPointSize(10);
160      glColor3f(1, 0, 0);
161      glBegin(GL_POINTS);
162      for (int i = 0; i < p.size(); i++)
163          glVertex3f(p[i][0], p[i][1], 0);
164      glEnd();
165  }
```

4. remove

```
//r key
case 2:
{
    int idx = selectPoint(ee_goal);
    p.erase(p.begin() + idx);
    break;
}
```

selectPoint()함수로 커서와 가장 가까운 점을 선택합니다.

```
205    int selectPoint(Vector3f vec)
206    {
207        int index;
208        float min = 100;
209
210        //find nearest point
211        for (int i = 0; i < p.size(); i++)
212        {
213            float x, y;
214            x = (p[i][0] - vec[0]) * (p[i][0] - vec[0]);
215            y = (p[i][1] - vec[1]) * (p[i][1] - vec[1]);
216            if (min > x + y)
217            {
218                min = x + y;
219                index = i;
220            }
221        }
222        return index;
223    }
```

벡터의 x, y 좌표를 제곱하여 더한 것의 최소를 구해 해당하는 점의 index 를 리턴하고, p 에서 제거한다.

5. Drag

mouse 함수에 action = GLFW_RELEASE 를 추가하고 bool 변수 isDragged 를 통해 마우스가 클릭되고 있는 것을 감지한다.

```
385             case 3:
386             {
387                 isDragged = true;
388                 selectedPoint = selectPoint(ee_goal);
389                 break;
390             }
391             //i key
392             case 4:
393 >           {…
396             }
397
398             }
399         }
400         if (action == GLFW_RELEASE)
401             isDragged = false;
```

selectPoint()함수로 가장 가까운 점을 선택해 드래그를 할 수 있다.

6. Insert

커서와 가장 가까운 segment 직선, 직선이 없다면 가장 가까운 segment 포인트에 점을 추가한다.

```
//i key
case 4:
{
    selectCurve();
    break;
}
```

Segment 의 Point 를 x, y 변수를 통해 가져온다.

```
240   void selectCurve()
241   {
242       Vector3f index;
243       int idxPoint;
244       float min = 100;
245       float length;
246
247       //find Point
248       for (int i = 0; i < p.size() - 1; i++)
249       {
250           Vector3f beforePoint;
251
252           for (int j = 0; j < 40; j++)
253           {
254               float t = (float)j / 39;
255
256               float x = c(4 * i, 0) + (c(4 * i + 1, 0) + (c(4 * i + 2, 0) + c(4 * i + 3, 0)*t)*t)*t;
257               float y = c(4 * i, 1) + (c(4 * i + 1, 1) + (c(4 * i + 2, 1) + c(4 * i + 3, 1)*t)*t)*t;
```

```
259                   if (j != 0)
260                   {
261                       // vector : before Point to next Point
262                       float vectorX = x - beforePoint[0];
263                       float vectorY = y - beforePoint[1];
264
265                       float line = sqrt(vectorX * vectorX + vectorY * vectorY);
266
267                       // vector : before Point to cursor Point
268                       float cursorX = ee_goal[0] - beforePoint[0];
269                       float cursorY = ee_goal[1] - beforePoint[1];
270
271                       // vector : before Point to cursor Point
272                       float nowX = ee_goal[0] - x;
273                       float nowY = ee_goal[1] - y;
274
275                       float dot = vectorX * cursorX + vectorY * cursorY;
```

Segment 의 처음인 j=0 일때는 x, y 를 beforePoint 벡터변수에 저장하고, j=1 일 때부터 beforePoint, nowPoint, CursorPoint 를 이용해 가까운 곳을 찾기 시작한다.

```
277     //find dot Proj(a,b) = dot1/(line*line)*vector
278     float pointX, pointY;
279     //beforePoint = nowPoint, line = 0
280     if (line == 0)
281     {
282         pointX = beforePoint[0];
283         pointY = beforePoint[1];
284     }
285     else
286     {
287         pointX = beforePoint[0] + dot / (line * line) * vectorX;
288         pointY = beforePoint[1] + dot / (line * line) * vectorY;
289     }
```

정사영 공식을 이용해 beforePoint 와 nowPoint 를 연결하는 직선위의 정사영벡터의 좌표를 구한다.

Data Point 가 가까울 때, beforePoint 와 nowPoint 가 같은 좌표가 되는 경우가 생겨 발생되는 오류를 예외처리 해준다.

```
291     //range check
292     float cos=0;
293     if (beforePoint[0] < pointX && pointX < x)
294     {
295         if (beforePoint[1] < pointY && pointY < y)
296             cos = 1;
297     }
298     else if (x < pointX && pointX < beforePoint[0])
299     {
300         if (y < pointY && pointY < beforePoint[1])
301             cos = 1;
302     }
```

구해진 정사영 벡터의 좌표가 beforePoint 와 nowPoint 사이에 있는지 검사하여 사이에 있다면 cos 에 1 을 저장한다.

```
304         //point is inline
305         if (cos == 1)
306         {
307             // proj : dot(vector,cursor) / length(vector)
308             float proj = fabs(dot / line);
309 
310             if (isnan(proj) != 0) //vector Point == proj Point H, When line = 0
311                 proj = 0;
312 
313             // length : cursor^2 - proj^2
314             length = sqrt(cursorX * cursorX + cursorY * cursorY - proj * proj);
315         }
```

정사영 벡터가 선분안에 있다면 피타고라스 정리로 선분과의 길이를 구한다.

```
316         else
317         {
318             beforePoint = Vector3f(x, y, 0);
319 
320             float beforeLength = sqrt(cursorX * cursorX + cursorY * cursorY);
321             float nowLength = sqrt(nowX * nowX + nowY * nowY);
322             //select close Point
323             if (beforeLength <= nowLength)
324             {
325                 length = nowLength;
326                 pointX = x;
327                 pointY = y;
328             }
329             else
330             {
331                 length = beforeLength;
332                 pointX = beforePoint[0];
333                 pointY = beforePoint[1];
334             }
335 
336         }
```

선분 밖에 있다면 더 가까운 점을 선택하고 그 점과의 거리를 구한다.

```
if (min > length)
{
    min = length;

    index = Vector3f(pointX, pointY, 0);
    idxPoint = i;
}
```

반복하면서 가장 가까운 곳을 찾는다.

```
//insert Point
p.insert(p.begin() + idxPoint + 1, index);
```

7. 실행화면