2023학년도 1학기

컴퓨터애니메이션

컴퓨터 애니메이션 보고서

Self-Scoring Table

	Report	video	Add	Remove	Drag	Insert
Score	1	1	1	1	1	1

정보융합학부 2018204058 김민교

1 - Mouse click and mouse move

```
80      // 버튼을 누를 때 마다, 마우스 클릭의 월드 좌표를 알아낸다.
81      void mouseButton(GLFWwindow* window, int button, int action, int mods)
82      {
83          double xs, ys;
84          if (action == GLFW_PRESS && button == GLFW_MOUSE_BUTTON_LEFT)
85          {
86              // Mouse cursor position in the screen coordinate
87              glfwGetCursorPos(window, &xs, &ys);
88
89              // Mouse cursor position in the framebuffer coordinate
90              // 이거를 unproject하면 world space에서 x,y,z를 가질 수 있다.
91              // z는 상관하지 않는다. 어차피 x,y평면에 있기 때문
92              xs = xs * dpiScaling;
93              ys = ys * dpiScaling;
94
95              if (interactMode == NONE)
96                  return;
97
98              GLdouble x_ws, y_ws, z_ws;
99              unProject(xs, ys, &x_ws, &y_ws, &z_ws);
100             if (preX == x_ws && preY == y_ws) {
101                 return;
102             }
103
104             if (interactMode == ADD) addDataPoint(x_ws, y_ws);
105             else if (interactMode == ERASE) eraseDataPoint(x_ws, y_ws);
106             else if (interactMode == DRAG) selectedDataPoint(x_ws,y_ws);
107             else if (interactMode == INSERT) insertDataPoint(x_ws,y_ws);
108
109             preX = x_ws; preY = y_ws;
110         }
111
112     }
```

mouse가 클릭한 screen 좌표를 world 좌표로 바꾼다. 그리고 interactMode가 ADD, ERASE, DRAG, INSERT냐에 따라 각각의 함수를 호출한다.

```
61      void mouseMove(GLFWwindow* window, double xs, double ys)
62      {
63          // Drag 모드가 아니거나 드래그 하고있지 않으면 return
64          if (glfwGetMouseButton(window, GLFW_MOUSE_BUTTON_LEFT) == GLFW_RELEASE || interactMode != DRAG) return;
65
66          if (!IsSelectedPoint) // 선택된게 없어서 움직일 게 없다.
67              return;
68
69          // Mouse cursor position in the framebuffer coordinate
70          double x = xs * dpiScaling;
71          double y = ys * dpiScaling;
72
73          // 마우스 좌표 unProject.
74          GLdouble x_ws, y_ws, z_ws;
75          unProject(x, y, &x_ws, &y_ws, &z_ws);
76          dragDataPoint(x_ws, y_ws);
77
78      }
```

mouse move는 interactMode가 Drag일 때만 작용한다. 선택된 dataPoint가 없으면 함수를 종료한다.

2 - 자료구조

[control point]

```
46      //(x,y,z) of data points
47      int N = 0; // curve segement
48      vector<vector<GLdouble>> points; // control points
```

저번 과제에서 Iterator를 썼던게 불편하여, 인덱싱이 편한 벡터로 control points를 관리했다.

[sample point[

```
48      // Samples
49      vector<vector<float>> samples;
```

sample point는 control polygon을 그리기 위한 control point들이 담긴다.

3 - Selected point, selected edge

[selected point]

selected point는 data point와 마우스 클릭 좌표의 거리가 0.0002 이내면 selected되었다고 판단했다.

```
130     for (int i=0; i<N; i++)
131     {
132         float dist = (points[i][0] - x_ws) * (points[i][0] - x_ws) + (points[i][1] - y_ws) * (points[i][1] - y_ws);
133
134         if (dist <= 0.0002)
135         {
136             if (minDist > dist) {
137                 minDist = dist;
138                 minIndex = i;
139             }
140         }
141     }
```

control point가 담긴 벡터를 순회한다. 그러면서 마우스 클릭 좌표와 control point 좌표와의 거리를 구하고, 이 거리가 0.0002 이내면 일단 minDist라고 저장한다. 또한 현재 리스트의 노드를 가리키는 iter도 저장한다. minDist가 있어야 하는 이유는 가장 가까운 control point를 선택해야하기 때문이다.

```
142     if (minDist == 1) // 0.002이내에 점이 없다
143     {
144         IsSelectedPoint = false;
145         return false;
146     }
147
148     selectedIndex = minIndex;
149     IsSelectedPoint = true;
150
151     return true;
152 }
```

벡터 순회가 끝나고, minDist가 1이라면 선택된 control point가 없다는 뜻이다. 그렇지 않다면 선택된 control point를 가리키는 인덱스를 selectedIndex에 저장하고, 선택된 control point가 있다는 뜻인 IsSelectedPoint값도 true로 설정한다. 이 값은 후에 drag 할 때 쓰인다.

[selected edge]

selected edge는 연속적인 2개의 control point가 이루는 직선에서 마우스 클릭 좌표까지의 거리, 마우스 클릭 좌표와 2개의 control point까지의 각각의 거리 , 총 3개의 거리를 구해서 이 거리 중 최소값이 0.0004 이내면 edge를 선택했다고 판단했다.

```
159     //sample point를 전부 순회... => 어차피 4개
160     for (int i = 0; i < 3; i++)
161     {
162         // samples[i]와 samples[i+1]의 직선의 방정식을 구한다.
163         // 기울기
164         float m = (samples[i + 1][1] - samples[i][1]) / (samples[i + 1][0] - samples[i][0]);
165         // y절편
166         float c = -m * samples[i][0] + samples[i][1];
```

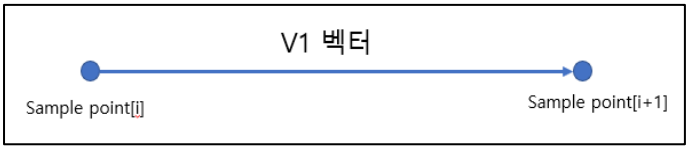
먼저 control point [i]와 control point [i+1]를 지나는 (1)직선의 방정식을 구한다.

```
168         // 점에서 직선에 내린 수선의 발을 구한다.
169         // 수직을 지나는 직선의 기울기 : -1/m
170         // y = -1/m + b <- 마우스 좌표를 넣어서 b를 득템.
171         // 그 후 연립 방정식으로?
172         float cPrime = (1 / m) * x_ws + y_ws;
173
174         // x랑 y는 수선의 발이다.
175         float x = (cPrime - c) / (m + (1 / m));
176         float y = m * x + c;
```

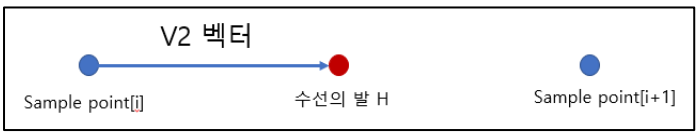
그 직선의 방정식에 수직하고 마우스 좌표를 지나는 (2)직선의 방정식을 구한다. (1)직선의 방정식과 (2) 직선의 방정식을 연립방정식 형태로 풀어서 수선의 발을 구한다.

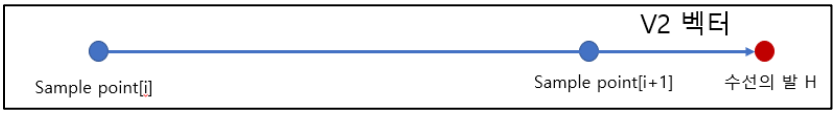
수선의 발을 구했지만, 이 수선의 발이 control point[i], control point[i+1] 선분 내부에 있는지 확인해야 한다. 이를 확인하기 위해서 벡터의 내적과 벡터의 길이를 사용했다.

```
236         float tempX1 = x-samples[i][0]; float tempY1 = y-samples[i][1];
237         float tempX2 = samples[i + 1][0]-samples[i][0]; float tempY2 = samples[i + 1][1]-samples[i][1];
238
239         Vector2f v1(tempX1,tempY1); // samples[i]번째 점에서 수선의 발로 가는 벡터.
240         Vector2f v2(tempX2,tempY2); // samples[i]에서 samples[i+1]로 가는 벡터
```



Sample point[i]에서 Sample point[i+1]로 가는 V1 벡터를 구한다.





control point[i]에서 수선의 발까지 가는 벡터는 총 3가지로 나뉜다. 수선의 발이 선분 내부에 있을 때, 선분 밖에 있는데 방향은 같을 때, 선분 밖에 있는데 방향이 다를 때이다.

1) 선분 밖에 있는데 방향이 다른 경우는, v1과 v2의 내적이 0보다 작을 때이다. 이때는 마우스 좌표와 control point [i]의 거리로 selected 되었는지 판단한다.

2) 선분 밖에 있는데 방향이 같은 경우는, v2의 길이가 v1의 길이보다 클 때이다. 이때는 마우스 좌표와 control point [i+1]의 거리로 selected 되었는지 판단한다.

3) 선분 안에 있다면 수선의 발과 마우스 클릭 좌표의 거리로 selected 되었는지 판단한다.

```
246         if (v1.dot(v2) > 0 && (v1.dot(v1) <= v2.dot(v2)))// 방향이 같고 길이가 v1이 더 작다면 수선의 발이 선분 내부에 있다.
247         {
248             dist = (x - x_ws) * (x - x_ws) + (y - y_ws) * (y - y_ws);// 거리는 점과 직선 사이의 거리. or 수선의 발과 마우스 좌표까지의 거리
249             tempInsert[0] = x;
250             tempInsert[1] = y;
251         }
252         else {
253             if (v1.dot(v2) < 0)
254             {
255                 dist = (samples[i][0] - x_ws) * (samples[i][0] - x_ws) + (samples[i][1] - y_ws) * (samples[i][1] - y_ws); // 거리는 sample point와의 거리.
256                 tempInsert[0] = samples[i][0];
257                 tempInsert[1] = samples[i][1];
258             }
259             else
260             {
261                 dist = (samples[i + 1][0] - x_ws) * (samples[i + 1][0] - x_ws) + (samples[i + 1][1] - y_ws) * (samples[i + 1][1] - y_ws);
262                 tempInsert[0] = samples[i+1][0];
263                 tempInsert[1] = samples[i+1][1];
264             }
265         }
```

위의 내용을 코드로 구현했다.

```
267         if (dist <= 0.0004)
268         {
269             if (minDist > dist) {
270                 dataPointInserted[0] = tempInsert[0];
271                 dataPointInserted[1] = tempInsert[1];
272                 minDist = dist;
273                 minIndex= i;
274             }
275         }
276     }
277
278     if (minDist == 1) // 0.002이내에 점이 없다
279     {
280         IsSelectedEdge = false;
281         return false;
282     }
283
284     IsSelectedEdge = true;
285     selectedEdgeIndex = minIndex;
286
287
288     return true;
289
290 }
```

dist가 0.0004이내면 삽입할 datapoint를 의미하는 dataPointInserted 변수에 좌표를 넣는다. control point들을 전부 순회하고, 선택된 edge가 몇 번째 control point 에서 출발하는지 나타내는 seletedEdgeIndex값도 저장한다.

4 - Add

```
75  void addDataPoint(GLdouble x_ws, GLdouble y_ws)
76  {
77      vector<GLdouble> v = { x_ws, y_ws, 0 };
78      points.push_back(v);
79      N += 1;
80  }
```

.

screnn space상의 좌표를 world space상의 좌표로 바꾼 값을 벡터 points에 추가한다. 그리고 controlPoint의 개수인 N도 +1 해준다.

5 - Remove

```
82  void eraseDataPoint(GLdouble x_ws, GLdouble y_ws)
83  {
84      if (selectedDataPoint(x_ws, y_ws)) {
85          points.erase(points.begin() + selectedIndex); // 가까운 것 삭제.
86          N -= 1;
87      }
88  }
```

controlPoint가 선택되었는지 판단하고, 선택되었다면 삭제한다. N도 감소시킨다.

6 - Drag

```
90  void dragDataPoint(GLdouble x_ws, GLdouble y_ws)
91  {
92      points[selectedIndex][0] = x_ws;
93      points[selectedIndex][1] = y_ws;
94  }
```

dragDataPoint함수는 mouse move에서 호출된다. 마우스 좌표의 값으로 위치를 갱신한다.

7 - Insert

```
96  void insertDataPoint(GLdouble x_ws, GLdouble y_ws)
97  {
98      if (selectedEdge(x_ws, y_ws))
99      {
100         // 현재 edge가 몇번 째 control point의 edge인지 알아낸다. -> 삽입하기 위함.
101         //0번째 일 때, 0~1사이에 껴 넣야한다.
102         int edgeIndex;
103
104         // edge Index는 iSegment에서 현재 선택된 edge가 몇번째 인지로 알아낸다.
105         edgeIndex = selectedEdgeIndex+ iSegment+1;
106
107         vector<GLdouble> v;
108         v.push_back(dataPointInserted[0]); v.push_back(dataPointInserted[1]); v.push_back(0);
109
110         // Repetition만큼 순서를 앞으로 땡긴다.
111         points.insert(points.begin()+(edgeIndex-REPETITION), v);
112         N += 1;
113     }
114 }
```

selectedEdgeIndex + iSegment를 하면 이 엣지가 몇 번째 control Point에서 출발하는지 알 수 있다. 그리고 control Point뒤에 삽입하기 때문에 +1을 해준다.

또 중요한 것은 삽입할 때, Repetition 만큼 순서를 앞으로 땡겨서 저장해야 한다. selecteEdgeIndex는 Repetition도 고려한 control point의 엣지 인덱스이기 때문이다.