```
 1  /*
 2   * Author  : Dubas Loïc
 3   * Class   : I.FA-P3B
 4   * School  : CFPT-I
 5   * Date    : June 2018
 6   * Descr.  : Drawing hand, circle and line functions
 7   * Version : 1.0
 8   * Ext. dll: LeapCSharp.NET4.5
 9   */
10
11  using System;
12  using System.Collections.Generic;
13  using System.Linq;
14  using System.Text;
15  using System.Threading.Tasks;
16  // References to add
17  using System.Drawing;
18  using System.Windows.Forms;
19  using Leap;
20
21  namespace fingers_cloner
22  {
23      class Paint
24      {
25          #region Initialization
26          // Fixed circle size
27          const int CIRCLESIZE = 50;
28
29          // Palm fixed location in the panel
30          Vector palmPanelPos;
31
32          // Dimensions of the panel
33          private int _panelWidth;
34          private int _panelHeight;
35          public int PanelWidth { get => _panelWidth; set => _panelWidth =
              value; }
36          public int PanelHeight { get => _panelHeight; set => _panelHeight =
              value; }
37
38          // hand to draw
39          private MyHand _hand;
40          private List<Vector> _fingersPanelPos;
41          private List<Vector> _modelePanelPos;
42          public MyHand Hand { get => _hand; set => _hand = value; }
43          public List<Vector> FingersPanelPos { get => _fingersPanelPos; set =>
              _fingersPanelPos = value; }
44          public List<Vector> ModelePanelPos { get => _modelePanelPos; set =>
              _modelePanelPos = value; }
45          #endregion
46
47          /// <summary>
48          /// Paint constructor
49          /// </summary>
50          /// <param name="panelWidth">Panel width</param>
51          /// <param name="panelHeight">Panel height</param>
52          public Paint() { }
```

```
53
54          /// <summary>
55          /// get the panel size
56          /// </summary>
57          /// <param name="panelWidth">Panel width</param>
58          /// <param name="panelHeight">Panel height</param>
59          public void GetPanelSize(int panelWidth, int panelHeight)
60          {
61              this.PanelWidth = panelWidth;
62              this.PanelHeight = panelHeight;
63
64              palmPanelPos = new Vector((PanelWidth / 2), 0, (PanelHeight -
                  CIRCLESIZE));
65          }
66
67          #region drawing black
68          /// <summary>
69          /// Draw a hand
70          /// </summary>
71          /// <param name="e">paint event</param>
72          /// <param name="hand">hand to paint</param>
73          public void paintHand(PaintEventArgs e, MyHand hand)
74          {
75              this.Hand = hand;
76              FingersPanelPos = normToPalmPanelPos();
77
78              this.DrawEllipseRectangle(e, Convert.ToInt32(palmPanelPos.x),
                  Convert.ToInt32(palmPanelPos.z));
79              for (int i = 0; i < FingersPanelPos.Count; i++)
80              {
81                  this.DrawEllipseRectangle(e, Convert.ToInt32(FingersPanelPos
                      [i].x), Convert.ToInt32(FingersPanelPos[i].z));
82                  this.DrawLinePoint(e, Convert.ToInt32(FingersPanelPos[i].x),
                      Convert.ToInt32(FingersPanelPos[i].z));
83              }
84          }
85
86          /// <summary>
87          /// Draw a circle at a certain location
88          /// </summary>
89          /// <param name="e">Paint event</param>
90          /// <param name="x">Horizonzal coordinate of finger/palm</param>
91          /// <param name="z">Vertical coordinate of finger/palm</param>
92          private void DrawEllipseRectangle(PaintEventArgs e, int x, int z)
93          {
94              // Create pen.
95              Pen Pen = new Pen(Color.Black, 3);
96
97              // Create rectangle for ellipse.
98              Rectangle rect = new Rectangle(x - (CIRCLESIZE / 2), z -
                  (CIRCLESIZE / 2), CIRCLESIZE, CIRCLESIZE);
99
100             // Draw ellipse to screen.
101             e.Graphics.DrawEllipse(Pen, rect);
102         }
103
```

```
104         /// <summary>
105         /// Draw a line beteween two points (center of palm to finger)
106         /// </summary>
107         /// <param name="e">Paint event</param>
108         /// <param name="x">Horizontal coordinate of finger</param>
109         /// <param name="z">Vertical coordinate of finger</param>
110         private void DrawLinePoint(PaintEventArgs e, int x, int z)
111         {
112             // Create pen.
113             Pen Pen = new Pen(Color.Black, 3);
114 
115             // Create points that define line.
116             Point point1 = new Point(Convert.ToInt32(palmPanelPos.x),
                  Convert.ToInt32(palmPanelPos.z));
117             Point point2 = new Point(x, z);
118 
119             // Draw line to screen.
120             e.Graphics.DrawLine(Pen, point1, point2);
121         }
122         #endregion
123 
124         #region drawing colors
125         /// <summary>
126         /// draw user's hand in color
127         /// </summary>
128         /// <param name="e">paint event</param>
129         /// <param name="hand">hand of user</param>
130         /// <param name="colors">list of colors of user's finger</param>
131         public void paintHandColor(PaintEventArgs e, MyHand hand, List<Color>
              colors)
132         {
133             this.Hand = hand;
134             FingersPanelPos = normToPalmPanelPos();
135 
136             this.DrawEllipseRectangle(e, Convert.ToInt32(palmPanelPos.x),
                  Convert.ToInt32(palmPanelPos.z));
137             for (int i = 0; i < FingersPanelPos.Count; i++)
138             {
139                 this.DrawEllipseRectangleColor(e, Convert.ToInt32
                      (FingersPanelPos[i].x), Convert.ToInt32(FingersPanelPos
                      [i].z), colors[i]);
140                 this.DrawLinePointColor(e, Convert.ToInt32(FingersPanelPos
                      [i].x), Convert.ToInt32(FingersPanelPos[i].z), colors[i]);
141             }
142         }
143 
144         /// <summary>
145         /// Draw a circle at a certain
146         /// </summary>
147         /// <param name="e">Paint event</param>
148         /// <param name="x">Horizonzal coordinate of finger/palm</param>
149         /// <param name="z">Vertical coordinate of finger/palm</param>
150         /// <param name="penColor">color of the finger</param>
151         private void DrawEllipseRectangleColor(PaintEventArgs e, int x, int z,
               Color penColor)
152         {
```

```
153             // Create pen.
154             Pen Pen = new Pen(penColor, 3);
155 
156             // Create rectangle for ellipse.
157             Rectangle rect = new Rectangle(x - (CIRCLESIZE / 2), z -
                  (CIRCLESIZE / 2), CIRCLESIZE, CIRCLESIZE);
158 
159             // Draw ellipse to screen.
160             e.Graphics.DrawEllipse(Pen, rect);
161         }
162 
163         /// <summary>
164         /// Draw a line beteween two points (center of palm to finger)
165         /// </summary>
166         /// <param name="e">Paint event</param>
167         /// <param name="x">Horizontal coordinate of finger</param>
168         /// <param name="z">Vertical coordinate of finger</param>
169         /// <param name="penColor">color of the finger</param>
170         private void DrawLinePointColor(PaintEventArgs e, int x, int z, Color
              penColor)
171         {
172             // Create pen.
173             Pen Pen = new Pen(penColor, 3);
174 
175             // Create points that define line.
176             Point point1 = new Point(Convert.ToInt32(palmPanelPos.x),
                  Convert.ToInt32(palmPanelPos.z));
177             Point point2 = new Point(x, z);
178 
179             // Draw line to screen.
180             e.Graphics.DrawLine(Pen, point1, point2);
181         }
182         #endregion
183 
184         #region transform norm to panel position
185         /// <summary>
186         /// Calculate the position on the panel with the normalized vector
187         /// </summary>
188         /// <returns>A list of vector with the finger's position to the palm</
              returns>
189         public List<Vector> normToPalmPanelPos()
190         {
191             float scaleFactor = PanelHeight + CIRCLESIZE;
192             List<Vector> fingersPanelPos = new List<Vector>();
193             Vector originToPalm = new Vector(Hand.PalmNormPos.x, 0,
                  Hand.PalmNormPos.z);
194             List<Vector> originToFingers = new List<Vector>();
195 
196             for (int i = 0; i < Hand.FingersNormPos.Count; i++)
197             {
198                 originToFingers.Add(new Vector(Hand.FingersNormPos[i].x, 0,
                      Hand.FingersNormPos[i].z));
199 
200                 fingersPanelPos.Add(new Vector((-originToPalm +
                      originToFingers[i]) * scaleFactor + palmPanelPos));
201             }
```

```
202
203             return fingersPanelPos;
204         }
205
206         /// <summary>
207         /// Calculate panel position of the modele hand
208         /// </summary>
209         /// <param name="modele">the current modele</param>
210         /// <returns>A list of positions</returns>
211         public List<Vector> normToPalmPanelModelePos(MyHand modele)
212         {
213             float scaleFactor = PanelHeight + CIRCLESIZE;
214             List<Vector> modelePanelPos = new List<Vector>();
215             Vector originToPalm = new Vector(modele.PalmNormPos.x, 0,
                  modele.PalmNormPos.z);
216             List<Vector> originToFingers = new List<Vector>();
217
218             for (int i = 0; i < modele.FingersNormPos.Count; i++)
219             {
220                 originToFingers.Add(new Vector(modele.FingersNormPos[i].x, 0,
                      modele.FingersNormPos[i].z));
221
222                 modelePanelPos.Add(new Vector((-originToPalm + originToFingers
                      [i]) * scaleFactor + palmPanelPos));
223             }
224
225             ModelePanelPos = modelePanelPos;
226
227             return modelePanelPos;
228         }
229         #endregion
230     }
231 }
232
```