

## Code Samples: Digital Straight Line in C#

Copyright © by W. Kovalevski and V. Miszalok, last update: 09-12-2006

This code has been developed with Visual C#2.0.

It furnishes the crack code of the best digital straight line segment DSS in the sense that the resulting DSS always meanders in the closest possible distance around the original analog line segment ASS. At any vertex the two possible following vertices `candidate1` and `candidate2` are tried using the "Hesse Normalized Standard Form" of the line that furnishes both point-to-line distances `d1` and `d2`. The algorithm yields best accuracy at moderate computational costs.

Create a new Windows-project "straight". Delete the files `Form1.Designer.cs` and `Program.cs` from the project. Clear any prefabricated content from `Form1.cs` and replace it by the following code, where the important lines are red:

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Collections;
using System.Text;

public class Form1 : Form
{ static void Main() { Application.Run( new Form1() ); }
  const int nButtons = 3;
  const int CRno = 16; //no of columns/rows
  int CRsize; //width = height of a column/row
  Button[] button = new Button[nButtons];
  Pen pen = new Pen( Color.Black, 1 );
  Brush brush = new SolidBrush( Color.Red );
  Panel panel = new Panel();
  Graphics g;
  PointF p0 = new PointF(), p1 = new PointF(), m = new PointF();
  Random r = new Random();
  dLineClass dLine;

  public class dLineClass
  { public PointF p0f = new PointF();
    public PointF plf = new PointF();
    public Point p0i = new Point();
    public Point pli = new Point();
    public float[] d;
    public int idx, idy, adx, ady;
    public float a, b, c, norm;
    public StringBuilder eswn;

    public dLineClass( PointF start, PointF end )
    { p0f = start; p0i = Point.Round( p0f );
      plf = end; pli = Point.Round( plf );
      eswn = new StringBuilder( "(" + (p0i.X).ToString() + "/" +
                                (p0i.Y).ToString() + ")" );

      idx = pli.X-p0i.X; adx = Math.Abs( idx );
      idy = pli.Y-p0i.Y; ady = Math.Abs( idy );
      //Standard Form for the Equation of a Line: a*x + b*y + c = 0;
      a = plf.Y - p0f.Y;
      b = -plf.X + p0f.X;
      c = - p0f.X * ( plf.Y - p0f.Y ) + p0f.Y * ( plf.X - p0f.X );
      float norm = (float)Math.Sqrt( a*a + b*b );
      if ( c > 0 ) norm = -norm;
      a /= norm; b /= norm; c /= norm; //Hesse Normalized Standard Coefficients
    }
  }
}
```

```

public void cracks()
{ int x_increment = Math.Sign( idx );
  int y_increment = Math.Sign( idy );
  d = new float[adx+ady+1]; //adx+ady+1 = no of 0-cells = no of points
  d[0] = a * p0i.X + b * p0i.Y + c; //1st point-to-line-distance
  Point p = new Point(); p = p0i; //set running point to 1st point
  if ( x_increment * y_increment == 0 ) //special cases Horiz/Vertical
  {   if ( x_increment > 0 ) eswn.Append( 'e', adx+ady );
      else if ( y_increment > 0 ) eswn.Append( 's', adx+ady );
      else if ( x_increment < 0 ) eswn.Append( 'w', adx+ady );
      else if ( y_increment < 0 ) eswn.Append( 'n', adx+ady );
      for ( int i=1; i < d.Length; i++ ) //point-to-line-distances
        d[i] = a*(p.X+x_increment) + b*(p.Y+y_increment) + c; return;
  }
  Point candidatel = new Point(); //1. alternative
  Point candidate2 = new Point(); //2. alternative
  //compare point-to-line-distances d1 with d2 and follow the closer one
  for ( int i=1; i <= adx+ady; i++ )
  { candidatel.X = p.X+x_increment; candidatel.Y = p.Y;
    candidate2.X = p.X; candidate2.Y = p.Y+y_increment;
    float d1 = a*candidatel.X + b*candidatel.Y + c; //point-to-line-distance
    float d2 = a*candidate2.X + b*candidate2.Y + c; //point-to-line-distance
    if ( Math.Abs( d1 ) <= Math.Abs( d2 ) )
    { p = candidatel; d[i] = d1;
      if ( x_increment > 0 ) eswn.Append( 'e' );
      else eswn.Append( 'w' );
    }
    else
    { p = candidate2; d[i] = d2;
      if ( y_increment > 0 ) eswn.Append( 's' );
      else eswn.Append( 'n' );
    }
  }
} // end of for
} // end of DLineClass.cracks()
} // end of DLineClass

public Form1()
{ BackColor = Color.White;
  Text = "Digital Straight Line";
  for ( int i=0; i < nButtons; i++ )
  { button[i] = new Button(); Controls.Add( button[i] );
    button[i].Click += new EventHandler( button_handler );
    button[i].BackColor = Color.Gray;
  }
  button[0].Text = "Line";
  button[1].Text = "Clear";
  button[2].Text = "Rotate Clockwise";
  Controls.Add( panel ); //Put a drawing space on Form1
  panel.Paint += new PaintEventHandler( PanelOnPaint );
  Size = new Size( 800, 600 ); // calls OnResize()
}

protected override void OnResize( EventArgs e )
{ Int32 w = ClientRectangle.Width / 5;
  Int32 h = ClientRectangle.Height / nButtons;
  Int32 top = 1;
  for ( int i=0; i < nButtons; i++ )
  { Controls[i].Top = top; top += h;
    Controls[i].Left = 2;
    Controls[i].Width = w;
    Controls[i].Height = h - 2;
  }
  panel.Location = new Point( w+4, 2 );
  panel.Size = new Size( ClientRectangle.Width-panel.Location.X,
    ClientRectangle.Height-4 );
  //width = height of a column/row
  if ( panel.Width <= panel.Height ) CRsize = panel.Width /CRno;
  else CRsize = panel.Height/CRno;
  m.X = m.Y = CRno/2; //mid point of the grid
  panel.Invalidate();
}

```

```

protected void PanelOnPaint( object o, PaintEventArgs e )
{ if ( g != null ) g.Dispose();
  g = e.Graphics;
  g.Clear( SystemColors.Window );
  pen.Color = Color.Black;
  int d = CRsize; //draw grid lines
  for ( int x=0; x <= CRno; x++ ) g.DrawLine( pen, x*d, 0, x*d, CRno*d );
  for ( int y=0; y <= CRno; y++ ) g.DrawLine( pen, 0, y*d, CRno*d, y*d );
  if ( p0.Equals( p1 ) ) return;
  pen.EndCap = System.Drawing.Drawing2D.LineCap.ArrowAnchor;
  pen.Color = Color.Red; pen.Width = 5;
  Point p = new Point(); p = dLine.p0i; //current 0-cell
  //loop through the directions of eswn-string and draw the red arrows
  //and print the distances d[i] right below any 0-cell
  //start at 1st direction z=e/s/w/n inside eswn-string (x0,y0)z.....
  int first = dLine.eswn.ToString().IndexOf('') + 1;
  for ( int i = first; i < dLine.eswn.ToString().Length; i++ )
  { String si = String.Format( " {0:F2}", dLine.d[i-first] );
    g.DrawString( si, Font, brush, p.X*d, p.Y*d );
    switch ( dLine.eswn[i] )
    { case 'e': g.DrawLine( pen, p.X*d, p.Y*d, (++p.X)*d, p.Y*d ); break;
      case 's': g.DrawLine( pen, p.X*d, p.Y*d, p.X*d, (++p.Y)*d ); break;
      case 'w': g.DrawLine( pen, p.X*d, p.Y*d, (--p.X)*d, p.Y*d ); break;
      case 'n': g.DrawLine( pen, p.X*d, p.Y*d, p.X*d, (--p.Y)*d ); break;
    }
  }
  //draw the last distance d[d.Length-1] right below the last 0-cell
  String slast = String.Format( " {0:F2}", dLine.d[dLine.d.Length-1] );
  g.DrawString( slast, Font, brush, p.X*d, p.Y*d );
  //draw the analog straight line p0 -> p1
  pen.Color = Color.Green; pen.Width = 1;
  g.DrawLine( pen, dLine.p0f.X*d, dLine.p0f.Y*d, dLine.plf.X*d, dLine.plf.Y*d );
  pen.EndCap = System.Drawing.Drawing2D.LineCap.NoAnchor;
  //Draw the crack code into the lower left corner
  g.DrawString( dLine.eswn.ToString(), Font, brush, 0, panel.Height-4*Font.Height );
  //Print the Hesse Normalized Standard Form of line(p0, p1)
  String s = "H(x,y) = " + String.Format( " {0:F}*x" , dLine.a );
  if ( dLine.b > 0f ) s += String.Format( " + {0:F}*y" , dLine.b );
  else s += String.Format( " {0:F}*y" , dLine.b );
  if ( dLine.c > 0f ) s += String.Format( " + {0:F} = 0", dLine.c );
  else s += String.Format( " {0:F} = 0", dLine.c );
  g.DrawString( s, Font, brush, 0, panel.Height - 3*Font.Height );
}

protected void button_handler( object sender, System.EventArgs e )
{ switch( ((Button)sender).Text )
  { case "Line":
    //random point somewhere inside the upper left quadrant
    float x = m.X * (float)r.NextDouble();
    float y = m.Y * (float)r.NextDouble();
    //m = symmetry point in the mid of the panel, see code at end of OnResize()
    p0.X = m.X-x; p0.Y = m.Y-y; //start in the upper left quadrant
    p1.X = m.X+x; p1.Y = m.Y+y; //mirror in the lower right quadrant
    dLine = new dLineClass( p0, p1 ); dLine.cracks(); break;
  case "Clear":
    p1 = p0; break;
  case "Rotate Clockwise":
    if ( p1.Equals( p0 ) ) return;
    double arcus = 5 * Math.PI/180.0; // rotation step = 5 degrees
    float cosinus = (float)Math.Cos( arcus );
    float sinus = (float)Math.Sin( arcus );
    p0.X = (dLine.p0f.X-m.X)*cosinus - (dLine.p0f.Y-m.Y)* sinus + m.X;
    p0.Y = (dLine.p0f.X-m.X)* sinus + (dLine.p0f.Y-m.Y)*cosinus + m.Y;
    p1.X = (dLine.plf.X-m.X)*cosinus - (dLine.plf.Y-m.Y)* sinus + m.X;
    p1.Y = (dLine.plf.X-m.X)* sinus + (dLine.plf.Y-m.Y)*cosinus + m.Y;
    dLine = new dLineClass( p0, p1 ); dLine.cracks(); break;
  }
  panel.Invalidate();
}
}

```

#### Recommended experiments:

- 1) Resize to extreme window formats: broad+thin or narrow+high. You will obtain smaller pixels.
- 2) Observe the deformations of the digital line caused by rotation.
- 3) Observe the length of the crack code while rotating.