

## Code Samples: 3D Digital Straight Line in C#

Copyright © by W. Kovalevski and V. Miszalok, last update: 22-01-2007

This code has been developed with Visual C#2.0. It requires the presence of the DirectX SDK. See: [www.miszalok.de/C\\_3DCis/Index\\_of\\_Course.htm](http://www.miszalok.de/C_3DCis/Index_of_Course.htm)

It furnishes the crack code of the best 3D digital straight line segment DSS in the sense that the resulting DSS always meanders in the closest possible distance around the original 3D analog line segment ASS. At any vertex the three possible following vertices  $t_1$ ,  $t_2$  and  $t_3$  are tried using the "Hesse Normalized Standard Form" of the line that furnishes three point-to-line distances  $d_1$ ,  $d_2$  and  $d_2$ . The algorithm yields best accuracy at moderate computational costs.

Create a new Windows-project "straight3D". 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.

Inside the Solution Explorer-window click the plus-sign in front of `straight3D`. A tree opens. Look for the branch "References". **Right**-click References and **left**-click Add Reference... An Add Reference dialog box opens. Scroll down to the component name: `Microsoft.DirectX Version 1.0.2902.0`.

Highlight this reference by a left-click and (holding the Strg-key pressed) two more references:

`Microsoft.DirectX.Direct3D Version 1.0.2902.0` and

`Microsoft.DirectX.Direct3DX Version 1.0.2902.0` or `1.0.2903.0` or `1.0.2904.0`.

Quit the Add Reference dialog box with OK.

Check if three references:

`Microsoft.DirectX` and

`Microsoft.DirectX.Direct3D` and

`Microsoft.DirectX.Direct3DX` are now visible inside the Solution Explorer window underneath `straight3D` → References.

Copy the following code into the empty `Form1.cs`- file:

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Text;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;

public class Form1 : Form
{
    static void Main() { Application.Run( new Form1() ); }
    static Device device = null;
    static float fAngle = 0f; //grid animation velocity
    static int GridSize = 1; //can be any no. between 1 and 10
    static float GridStep = 2f / GridSize;
    static Mesh meshGridCyl, meshCrackCyl, meshArrowCyl, meshSphere;
    static Vector3 p0 = new Vector3(), p1 = new Vector3();
    static Panel panel = new Panel();
    static StringBuilder eswnbf = new StringBuilder();
    static System.Drawing.Font font = new System.Drawing.Font( "Arial", 16 );
    int distance = 0;
    Random r = new Random();
    Timer myTimer = new Timer();
    const int nButtons = 8, nTrackBars = 2;
    Button [] button = new Button [ nButtons ];
    TrackBar[] trackbar = new TrackBar[nTrackBars];
    Label [] label = new Label [nTrackBars];
}
```

```

public Form1()
{
    BackColor = Color.White;
    Text = "3D Digital Straight Line Segment DSS";
    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 = "Diagonal Line";
    button[1].Text = "Clear";
    button[2].Text = "Rotate X+"; button[3].Text = "Rotate X-";
    button[4].Text = "Rotate Y+"; button[5].Text = "Rotate Y-";
    button[6].Text = "Rotate Z+"; button[7].Text = "Rotate Z-";
    for ( int i=0; i < nTrackBars; i++ )
    {
        trackbar[i] = new TrackBar(); Controls.Add( trackbar[i] );
        label [i] = new Label(); Controls.Add( label[i] );
        trackbar[i].AutoSize = false;
        trackbar[i].TickStyle = TickStyle.None;
        trackbar[i].ValueChanged += new EventHandler( trackbar_handler );
        label [i].TextAlign = ContentAlignment.TopCenter;
    }
    trackbar[0].Name = label[0].Text = "GridSize";
    trackbar[1].Name = label[1].Text = "Distance";
    trackbar[0].Minimum = 1; trackbar[0].Maximum = 10;
    trackbar[1].Minimum = 0; trackbar[1].Maximum = 10;
    trackbar[0].Value = GridSize; trackbar[1].Value = 0;
    Controls.Add( panel ); //Put a drawing space on Form1
    myTimer.Tick += new EventHandler( OnTimer );
    myTimer.Interval = 1;
    Size = new Size( 800, 600 ); // calls OnResize()
}

protected override void OnResize( System.EventArgs e )
{
    myTimer.Stop(); // stop the timer during initialization
    int w = ClientRectangle.Width / 5;
    int h = ClientRectangle.Height / (Controls.Count-1);
    Int32 top = 1;
    for ( int i=0; i < Controls.Count-1; 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 );
    try
    {
        PresentParameters presentParams = new PresentParameters();
        presentParams.Windowed = true; //no full screen display
        presentParams.SwapEffect = SwapEffect.Discard; //no swap buffer
        presentParams.EnableAutoDepthStencil = true; //with depth buffer
        presentParams.AutoDepthStencilFormat = DepthFormat.D16; //16 bit depth
        if ( device != null ) device.Dispose(); //free the old canvas if any
        device = new Device( 0, DeviceType.Hardware, panel,
            CreateFlags.SoftwareVertexProcessing, presentParams );
        Material mtrl = new Material();
        mtrl.Diffuse = mtrl.Ambient = Color.White;
        device.Material = mtrl;
        device.Lights[0].Type = LightType.Directional;
        device.Lights[0].Diffuse = System.Drawing.Color.DarkTurquoise;
        device.Lights[0].Direction = new Vector3( 1, 1, 5 );
        device.Lights[0].Enabled = true; //turn it on
        device.RenderState.Ambient = System.Drawing.Color.FromArgb( 0x202020 );
        device.Transform.View = Matrix.LookAtRH(
            new Vector3( 0f, 0f, -4f - distance ),
            new Vector3( 0f, 0f, 0f ),
            new Vector3( 0f, -1f, 0f ) );
        device.Transform.Projection = Matrix.PerspectiveFovRH( (float)Math.PI/4, 1f, 1f, 20f );
        device.RenderState.CullMode = Cull.None;
        device.RenderState.Lighting = true;
        if ( meshGridCyl != null ) meshGridCyl.Dispose();
        if ( meshCrackCyl != null ) meshGridCyl.Dispose();
        if ( meshArrowCyl != null ) meshGridCyl.Dispose();
        if ( meshSphere != null ) meshSphere.Dispose();
        float a = 0.1f / GridSize;
        meshGridCyl = Mesh.Cylinder( device, a, a, 2f, 10, 2 );
        meshCrackCyl = Mesh.Cylinder( device, a*1.1f, a*1.1f, GridStep, 10, 2 );
        meshArrowCyl = Mesh.Cylinder( device, a*2f, 0, GridStep/5f, 20, 2 );
        meshSphere = Mesh.Sphere ( device, 1.5f*a, 20, 20 );
        myTimer.Start();
    }
    catch ( DirectXException ) { MessageBox.Show( "Could not initialize Direct3D." ); return; }
}

```

```

protected void button_handler( object sender, System.EventArgs e )
{
    myTimer.Stop();
    float cosinus = (float)Math.Cos( 5*Math.PI/180.0 );
    float sinus = (float)Math.Sin( 5*Math.PI/180.0 );
    if ( p0.Equals( pl ) ) sender = button[0];
    float help;
    switch( ((Button)sender).Text )
    {
        case "Diagonal Line":
            p0.X = -1f; pl.X = +1f;
            p0.Y = -1f; pl.Y = +1f;
            p0.Z = -1f; pl.Z = +1f;
            cracks(); break;
        case "Clear":
            eswnbf.Length = distance = 0; break;
        case "Rotate X+":
        case "Rotate X-":
            if ( ((Button)sender).Text == "Rotate X-" ) sinus *= -1f;
            help = p0.Y * cosinus - p0.Z * sinus;
            p0.Z = p0.Y * sinus + p0.Z * cosinus; p0.Y = help;
            help = pl.Y * cosinus - pl.Z * sinus;
            pl.Z = pl.Y * sinus + pl.Z * cosinus; pl.Y = help;
            cracks(); break;
        case "Rotate Y+":
        case "Rotate Y-":
            if ( ((Button)sender).Text == "Rotate Y-" ) sinus *= -1f;
            help = p0.Z * sinus + p0.X * cosinus;
            p0.Z = p0.Z * cosinus - p0.X * sinus; p0.X = help;
            help = pl.Z * sinus + pl.X * cosinus;
            pl.Z = pl.Z * cosinus - pl.X * sinus; pl.X = help;
            cracks(); break;
        case "Rotate Z+":
        case "Rotate Z-":
            if ( ((Button)sender).Text == "Rotate Z-" ) sinus *= -1f;
            help = p0.X * cosinus - p0.Y * sinus;
            p0.Y = p0.X * sinus + p0.Y * cosinus; p0.X = help;
            help = pl.X * cosinus - pl.Y * sinus;
            pl.Y = pl.X * sinus + pl.Y * cosinus; pl.X = help;
            cracks(); break;
    }
    myTimer.Start();
}

protected void trackbar_handler( object sender, System.EventArgs e )
{
    myTimer.Stop();
    switch( ((TrackBar)sender).Name )
    {
        case "GridSize":
            GridSize = trackbar[0].Value; GridStep = 2f / GridSize;
            eswnbf.Length = 0; OnResize( null );
            label[0].Text = "GridSize = " + trackbar[0].Value.ToString(); break;
        case "Distance":
            distance = trackbar[1].Value; OnResize( null );
            label[1].Text = "Distance = " + trackbar[1].Value.ToString(); break;
    } // end of switch
    myTimer.Start();
}

```

```

protected static void OnTimer( Object myObject, EventArgs myEventArgs )
{ if (device == null) return;
  device.Clear( ClearFlags.Target | ClearFlags.ZBuffer, Color.Blue, 1f, 0 );
  Graphics g = panel.CreateGraphics();
  g.DrawString( eswnbf.ToString(), font, Brushes.Black, 0, 0 );
  Material mtrl = new Material();
  Matrix anim = Matrix.RotationY( fAngle +=0.002f ) * Matrix.RotationX( (float)Math.PI/10 );
  float x, y, z;
  device.BeginScene();
  mtrl.Diffuse = mtrl.Ambient = Color.White; device.Material = mtrl;
  Vector3 t = new Vector3();
  for ( t.Y = -1f; t.Y < 1.00001f; t.Y += GridStep ) //z-rods
    for ( t.X = -1f; t.X < 1.00001f; t.X += GridStep )
      { device.Transform.World = Matrix.Translation( t ) * anim;
        meshGridCyl.DrawSubset( 0 );
      }
  Matrix turn = Matrix.RotationY( (float)Math.PI/2f ); //horiz. rods
  t.X = 0f;
  for ( t.Z = -1f; t.Z < 1.00001f; t.Z += GridStep )
    for ( t.Y = -1f; t.Y < 1.00001f; t.Y += GridStep )
      { device.Transform.World = turn * Matrix.Translation( t ) * anim;
        meshGridCyl.DrawSubset( 0 );
      }
  turn = Matrix.RotationX( (float)Math.PI/2f ); //vertical rods
  t.Y = 0f;
  for ( t.Z = -1f; t.Z < 1.00001f; t.Z += GridStep )
    for ( t.X = -1f; t.X < 1.00001f; t.X += GridStep )
      { device.Transform.World = turn * Matrix.Translation( t ) * anim;
        meshGridCyl.DrawSubset( 0 );
      }
  mtrl.Diffuse = mtrl.Ambient = Color.Green; device.Material = mtrl;
  device.Transform.World = Matrix.Translation(new Vector3(-1,-1,-1))*anim; meshSphere.DrawSubset(0);
  mtrl.Diffuse = mtrl.Ambient = Color.White; device.Material = mtrl;
  device.Transform.World = Matrix.Translation(new Vector3(-1,-1, 1))*anim; meshSphere.DrawSubset(0);
  device.Transform.World = Matrix.Translation(new Vector3(-1, 1,-1))*anim; meshSphere.DrawSubset(0);
  device.Transform.World = Matrix.Translation(new Vector3(-1, 1, 1))*anim; meshSphere.DrawSubset(0);
  device.Transform.World = Matrix.Translation(new Vector3( 1,-1,-1))*anim; meshSphere.DrawSubset(0);
  device.Transform.World = Matrix.Translation(new Vector3( 1,-1, 1))*anim; meshSphere.DrawSubset(0);
  device.Transform.World = Matrix.Translation(new Vector3( 1, 1,-1))*anim; meshSphere.DrawSubset(0);
  device.Transform.World = Matrix.Translation(new Vector3( 1, 1, 1))*anim; meshSphere.DrawSubset(0);
  if ( eswnbf.Length == 0 ) { device.EndScene(); device.Present(); return; }

  mtrl.Diffuse = mtrl.Ambient = Color.Green; device.Material = mtrl;
  device.Transform.World = Matrix.Translation( p0 ) * anim;
  meshSphere.DrawSubset( 0 );
  mtrl.Diffuse = mtrl.Ambient = Color.Black; device.Material = mtrl;
  device.Transform.World = Matrix.Translation( p1 ) * anim;
  meshSphere.DrawSubset( 0 );
  device.Transform.World = anim;
  Vector3[] pp = { p0, p1 };
  device.DrawUserPrimitives(PrimitiveType.LineList, 1, pp );
  mtrl.Diffuse = mtrl.Ambient = Color.Blue; device.Material = mtrl;
  Matrix translation = Matrix.Identity;
  x = translation.M41 = Convert.ToInt32( (1f+p0.X)/GridStep )*GridStep-1f;
  y = translation.M42 = Convert.ToInt32( (1f+p0.Y)/GridStep )*GridStep-1f;
  z = translation.M43 = Convert.ToInt32( (1f+p0.Z)/GridStep )*GridStep-1f;
}

```

```

int first = eswnbf.ToString().IndexOf('') + 1;
for ( int i = first; i < eswnbf.ToString().Length; i++ )
{ switch ( eswnbf[i] )
  { case 'e': turn = Matrix.RotationY( (float)Math.PI/2f );
    translation.M41 = x + GridStep/2;
    device.Transform.World = turn * translation * anim;
    x+=GridStep; break;
    case 's': turn = Matrix.RotationX( -(float)Math.PI/2f );
    translation.M42 = y + GridStep/2;
    device.Transform.World = turn * translation * anim;
    y+=GridStep; break;
    case 'w': turn = Matrix.RotationY( -(float)Math.PI/2f );
    x-=GridStep;
    translation.M41 = x + GridStep/2;
    device.Transform.World = turn * translation * anim;
    break;
    case 'n': turn = Matrix.RotationX( (float)Math.PI/2f );
    y-=GridStep;
    translation.M42 = y + GridStep/2;
    device.Transform.World = turn * translation * anim;
    break;
    case 'b': translation.M43 = z + GridStep/2;
    device.Transform.World = translation * anim;
    z+=GridStep; break;
    case 'f': turn = Matrix.RotationX( -(float)Math.PI );
    z-=GridStep;
    translation.M43 = z + GridStep/2;
    device.Transform.World = turn * translation * anim;
    break;
    default: Console.Beep(); break;
  } // end of switch
  meshCrackCyl.DrawSubset( 0 );
  meshArrowCyl.DrawSubset( 0 );
  translation.M41 = x; translation.M42 = y; translation.M43 = z;
} // end of for */
device.EndScene();
device.Present();
}

public void cracks()
{ if (p0.X < -1f) p0.X=-1f; if (p0.Y < -1f) p0.Y=-1f; if (p0.Z < -1f) p0.Z=-1f;
  if (p1.X < -1f) p1.X=-1f; if (p1.Y < -1f) p1.Y=-1f; if (p1.Z < -1f) p1.Z=-1f;
  if (p0.X > 1f) p0.X= 1f; if (p0.Y > 1f) p0.Y= 1f; if (p0.Z > 1f) p0.Z= 1f;
  if (p1.X > 1f) p1.X= 1f; if (p1.Y > 1f) p1.Y= 1f; if (p1.Z > 1f) p1.Z= 1f;
  int x0=Convert.ToInt32((1f+p0.X)/GridStep), x1=Convert.ToInt32((1f+p1.X)/GridStep);
  int y0=Convert.ToInt32((1f+p0.Y)/GridStep), y1=Convert.ToInt32((1f+p1.Y)/GridStep);
  int z0=Convert.ToInt32((1f+p0.Z)/GridStep), z1=Convert.ToInt32((1f+p1.Z)/GridStep);
  float x_increment = Math.Sign(x1-x0) * GridStep;
  float y_increment = Math.Sign(y1-y0) * GridStep;
  float z_increment = Math.Sign(z1-z0) * GridStep;

  String s = "(" + x0.ToString() + "/" + y0.ToString() + "/" + z0.ToString() + ")";
  eswnbf.Length = 0; eswnbf.Append( s );
  Vector3 p = new Vector3( x0*GridStep-1f, y0*GridStep-1f, z0*GridStep-1f );

  float dx = p1.X - p0.X, dy = p1.Y - p0.Y, dz = p1.Z - p0.Z, divisor = dx*dx+dy*dy+dz*dz;
  float t1x, t1y, t1z, t2x, t2y, t2z, t3x, t3y, t3z, ddx, ddy, ddz, d1, d2, d3;
  int nx = Math.Abs(x1-x0), ny = Math.Abs(y1-y0), nz = Math.Abs(z1-z0);
  do
  { ddx = p.X - p0.X; ddy = p.Y - p0.Y; ddz = p.Z - p0.Z;
    t1x = dy*(ddz ) - dz*(ddy );
    t1y = dx*(ddz ) - dz*(ddx+x_increment);
    t1z = dx*(ddy ) - dy*(ddx+x_increment);
    t2x = dy*(ddz ) - dz*(ddy+y_increment);
    t2y = dx*(ddz ) - dz*(ddx );
    t2z = dx*(ddy+y_increment) - dy*(ddx );
    t3x = dy*(ddz+z_increment) - dz*(ddy );
    t3y = dx*(ddz+z_increment) - dz*(ddx );
    t3z = dx*(ddy ) - dy*(ddx );
    if (nx > 0) d1 = (float)Math.Sqrt( (t1x*t1x+t1y*t1y+t1z*t1z)/divisor ); else d1=100f;
    if (ny > 0) d2 = (float)Math.Sqrt( (t2x*t2x+t2y*t2y+t2z*t2z)/divisor ); else d2=100f;
    if (nz > 0) d3 = (float)Math.Sqrt( (t3x*t3x+t3y*t3y+t3z*t3z)/divisor ); else d3=100f;
    char c = '?';
    if (d1<=d2&&d1<=d3&&nx>0) {p.X+=x_increment; nx--; if (x_increment > 0) c='e'; else c='w';}
    else if (d2<=d1&&d2<=d3&&ny>0) {p.Y+=y_increment; ny--; if (y_increment > 0) c='s'; else c='n';}
    else if (d3<=d1&&d3<=d2&&nz>0) {p.Z+=z_increment; nz--; if (z_increment > 0) c='b'; else c='f';}
    eswnbf.Append( c );
  } while ( nx+ny+nz > 0 );
} // end of crack()
}

```