

Curve approximation using Bezier curves

Author: Nils Haeck

Date: 21 July 2003

Last updated: 26 Sep 2004

This document describes how to use the Delphi source code that implements line smoothing (or curve approximation) using connected Bezier segments. There are also versions for Java and C++, but they are not described here in detail.

The package contains these files:

main.pas
main.dfm
bezier.dpr

Main unit and project file. These files serve as an example on how to use BezierApprox.pas.

Note you must install RxLib and use Delphi 5 in order to compile this example.

BezierApprox.pas

This is the source file that implements bezier approximation

BezierApprox.pas

Interface:

```
{ unit BezierApprox

Project: Bezier
Date: 2002-05-10

Changes:
21-07-2003: Changed classes to records for compat with other languages

This unit handles approximation of curves using Bezier segments

Copyright (c) 2002, Nils Haeck M.Sc., SimDesign

The source code is partly based on the description of Bezier curves as
found in:
- "Computer Graphics" by Hearn/Baker, Prentice/Hall International Editions
  1994, ISBN 0-13-159690-X, pp. 327-333

The Newton Rapson method is a general numerical method, a description can
be found here:
- "The VNR Concise Encyclopedia of Mathematics", VNR, 1977,
  ISBN 0-442-22646-2, pp. 639

}
unit BezierApprox;

interface

uses
  Windows, Classes, Graphics;

type
```

```

// TBezierSegment is a structure that holds info for a Bezier segment.
// This is the start and end points, and the 2 control points.
TBezierSegment = record
  P1X, P1Y,          // Start point
  P2X, P2Y: integer; // End point
  C1X, C1Y,          // Control points - as doubles for enhanced precision
  C2X, C2Y: double;  // during calculations
end;

// TBezierList is a list of bezier segments. Always use PBezierList and use
// GetMem or ReallocMem to allocate memory for it
PBezierList = ^TBezierList;
TBezierList = array[0.. (MaxInt div SizeOf(TBezierSegment)) - 1] of TBezierSegment;

// TPointList is a list of TPoint segments.
PPointList = ^TPointList;
TPointList = array[0.. (MaxInt div SizeOf(TPoint)) - 1] of TPoint;

// Convert a list of bezier segments to a polyline type used by Windows' PolyBezier
// command. The array Points must be initialized with the correct number of elements
// upon calling!
// Use this formula to calculate the number of points:
// PointCount = 3 * BezierSegmentCount + 1
procedure BezierListToPolyline(BezierList: PBezierList; BezierCount: integer; var Points:
array of TPoint);

// This is the core routine of the bezier approximation. Approximate a point list with
// a set of Bezier curves. The curves are fitted with a maximum error on each pixel of
// Precision. The error is only calculated in the points, not in interpolated positions.
procedure BezierApproximation(PointList: PPointList; PointCount: integer; var BezierList:
PBezierList; var BezierCount: integer; Smoothness, Precision: double);

// Draw the bezier line segments, and start/end/control points
procedure DrawBezierControls(Canvas: TCanvas; BezierList: PBezierList; BezierCount:
integer; CtlColor, PtsColor: TColor);

// This procedure is created to test the Bezier representation and theory. For faster
// methods please use the PolyBezier routine of TCanvas. DrawBezierPoly will draw
// a list of Bezier segments onto Canvas with current Pen settings
procedure DrawBezierPoly(CanvasHandle: HDC; BezierList: PBezierList; BezierCount:
integer);

```

Structures

TBezierSegment:

This structure holds a single bezier segment (start and end points, control points)

TPoint:

The Windows structure that holds X and Y coordinates of a single point (integer values).

Procedure calls:

```

procedure BezierApproximation(PointList: PPointList; PointCount:
integer; var BezierList: PBezierList; var BezierCount: integer;
Smoothness, Precision: double);

```

This is the core routine of the Bezier approximation. It approximates a point list with a set of Bezier curves. The curves are fitted with a maximum error on each pixel of Precision and with the Smoothness setting.

Inputs:

PointList:

Pointer to the list that contains TPoint structures that hold the X and Y coordinates of the user freeform.

PointCount:

The number of points in PointList

Smoothness:

A number between 0 and 100 that indicates how “hard” or “smooth” the curves will be. A good default setting is somewhere between 50 and 80.

Precision:

A number that defines how precise the curve will match the user freeform, given in pixels. Usually a setting between 1 and 5 will do.

Outputs:

BezierList:

Pointer to the list that will hold the Bezier curve segments. Note that this list will be allocated in procedure BezierApproximation. The list must be cleared at program termination using a call to FreeMem().

BezierCount:

The number of Bezier segments in BezierList.

Example:

This basic Delphi example shows how to approximate the zigzag line with a Bezier curve. It does not serve any real purpose except being an example:

```
var
  PointList: PPointList;
  BezierList: PBezierList;
  APolyLine: array of TPoint;
begin
  // Create a point list
  GetMem(PointList, SizeOf(TPoint) * 3); // for 3 points
  // And this one should be nil; will be allocated by procedure
  BezierList := nil;

  // Set the points 0, 1 and 2
  PointList[0].X := 0.0;
  PointList[0].Y := 0.0;
  PointList[1].X := 1.0;
  PointList[1].Y := 1.0;
  PointList[2].X := 2.0;
  PointList[2].Y := 0.0;
```

```
// Now approximate them
BezierApproximation(PointList, 3, BezierList, BezierCount,
    50.0, 1.0);

// Draw them with GDI, so first create a polyline with right length
SetLength(APolyline, 3 * BezierCount + 1);

// Convert Bezier list to Polyline
BezierListToPolyline(BezierList, BezierCount, APolyline);

// Use windows GDI to draw it
Canvas.PolyBezier(APolyLine);

// Finally, free the lists we created
FreeMem(PointList);
FreeMem(BezierList);

end;
```

Source code purchase

Source code can be purchased at the cost of US\$99 through our website
www.simdesign.nl
(follow links Delphi Corner > Components > Line smoothing package)