Subject: Re: Couple of questions
Posted by rylek on Fri, 25 Nov 2005 09:53:25 GMT
View Forum Message <> Reply to Message

Most of the functions by themselves are purely mathematical calculations. It is up to you to choose when to call them and what to do with their results. Of course, as the functions are mostly concerned with 2D & 3D analytic geometry and transforms, they can be used very easily to implement graphics.

The subpackage Geom/Draw is designed to do this. A class named Plotter combines a Draw object with a coordinate transform (defined by a Matrixf) and so it can be used to implement drawing in a logical coordinate space while generating the visual output in another coordinate system bound to the logical coordinate system by an affine transform (a combination of shifting, rotating, scaling and skewing). There are specific classes named PathTool, AreaTool, MarkTool and TextTool which use Plotter to draw the respective objects (defined by their logical coordinates) on a Plotter object.

The subpackage Geom/Ctrl goes even further to define PlotterCtrl, a generic editor object used to display, zoom & pan and edit an arbitrary geometric object (no matter whether vector- or raster-oriented) with a certain logical extent within a simple Ctrl view. The control implements everything concerning scaling and panning, all you have to do to get something working is to derive from the class, override the virtual method Plot(Plotter&) and use it to draw your graphics. You also have to SetExtent to tell the control how big your logical object is.

The PlotterCtrl class also implements a very generic drag & drop mechanism, enabling you to easily define your own custom drag & drop algorithms. In my various commercial applications (cited on our homepage), mainly WinZPV and HydroCheck, this is used to edit charts, to visually define intervals for interpolation, to draw cross sections etc.

A simple example, examples/OpenGL, demonstrates the use of Matrixf3 and Camera to draw 3D vector graphics. This is a very rudimentary 3D terrain viewer, which draws a shaded image of the terrain surface (using OpenGL's Z buffer) and over it the linear mesh of terrain polygon edges (using Draw::DrawLine). This demonstrates correctness and match of the two different computation mechanisms.