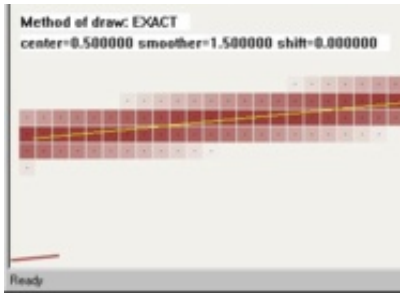
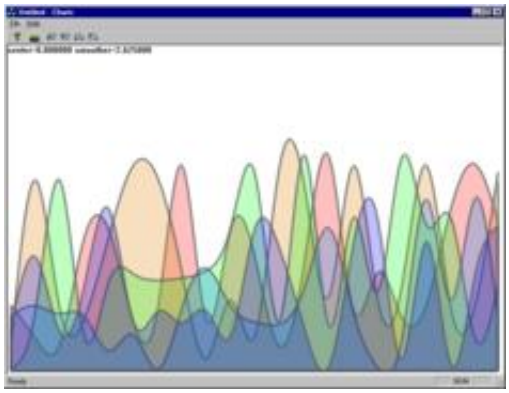
Screenshots

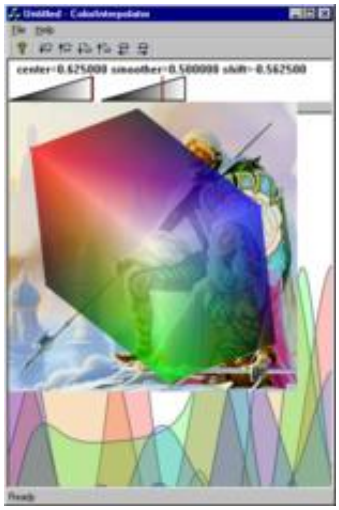
HISTORIC This is what **Anti-Grain Geometry** started with - Rotating and scaling pixel images with high performance high quality Anti-Aliasing.



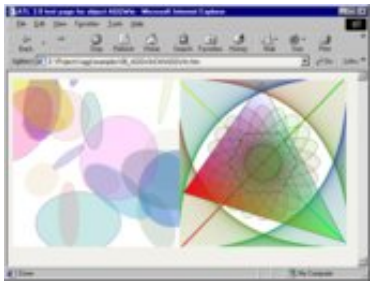
HISTORIC AGG-1. Demonstrating or the principle of **Anti-Aliasing** when drawing line segments. Here the subpixel accuracy of 1/16 of a pixel is used.



HISTORIC AGG-1. Using Anti-Aliased lines with subpixel accuracy and half-transparent polygons for drawing charts.

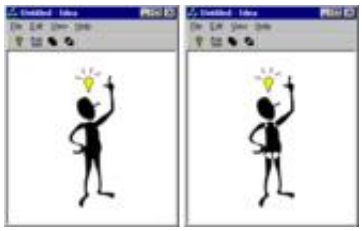


HISTORIC AGG-1. Rendering half-transparent triangles with color interpolation and Anti-Aliasing. Can be used for Gouraud shading.

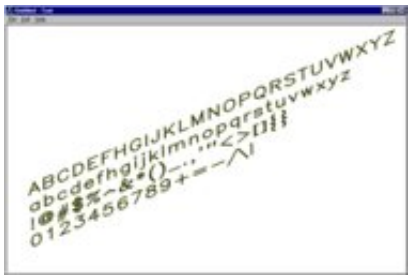


HISTORIC AGG-1. An Active-X control with a set of drawing functions that can be used in JavaScript or VB Script.

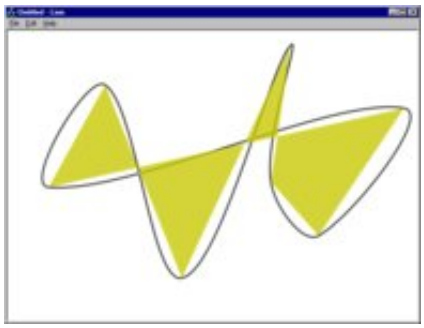
HISTORIC Demonstration of the filling rules (non-zero and even-odd) when two polygons with different orientations are being rendered at the same time.



More about filling rules. The polygon was taken from the book "Dynamic HTML in Action" by Eric Schurman. The left polygon is rendered using non-zero filling rule while the right one uses even-odd. Such a funny looking picture on the right is not my invention - I used the original polygons from the book :-)



Anti-Grain Geometry can render vector text. It actually is just an example of rendering polygonal figures. The font is encoded in a propriatory format, but it does not prevent us from rendering any standard fonts.



Demonstration of a simple method of smoothing polygons with Bezier curves. Polygons can be non-convex and even self-intersecting. See also Interpolation with Bezier Curves. **Anti-Grain Geometry** has converters that allow you to use it very easily.

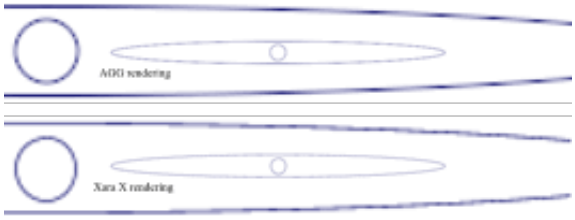
Popular in **SVG** lion rendered with **Anti-Grain Geometry**.

The same lion but rendered many times with transparency.

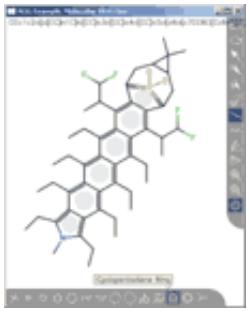
One more variant of the lion looking like a piece of surrealistic art.

A simple SVG Viewer renders the tiger. **Anti-Grain Geometry** does render it really very fast.

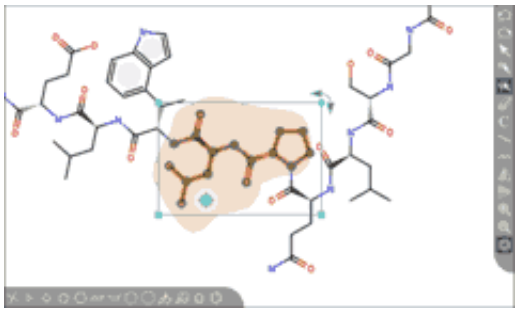
Another example rendered with the SVG Viewer. Here is a well known problem of adjacent Anti-Aliased edges. It appears when rendering adjacent polygons with anti-aliasing and looks like thin "web" upon the image. In **Anti-Grain Geometry** this problem has been successfully solved.



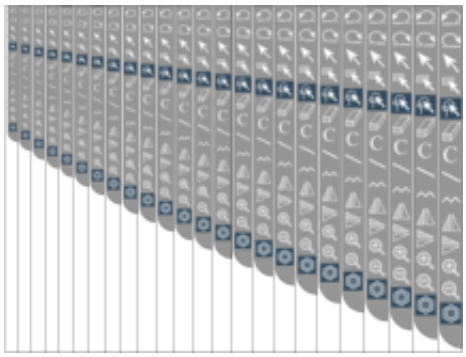
Comparison of the quality of rendering engines in **Anti-Grain Geometry** and a popular vector graphic editor **Xara X**. Adobe Photoshop, Adobe SVG Viewer, and many other renderers produce similar to **Xara X** results.



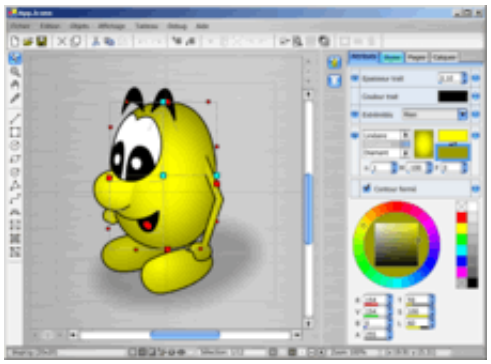
Chemcal sketcher that I work on at my primary job.



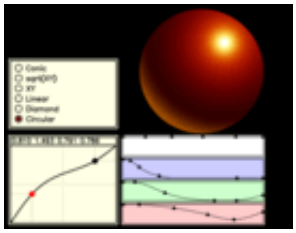
One more screenshot of the sketcher.



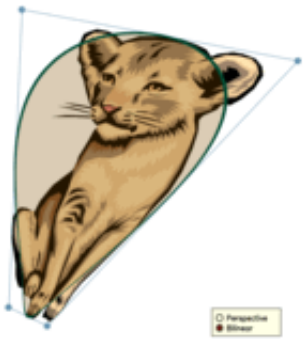
This “accordion” demonstrates the capability to display scalable toolbars. In the chemical sketcher the toolbar buttons are represented in a simple vector format and being rendered on the fly. Using pure vector graphics to represent all the GUI elements, including toolbar buttons allows you to get rid of the dependency on the screen resolution.



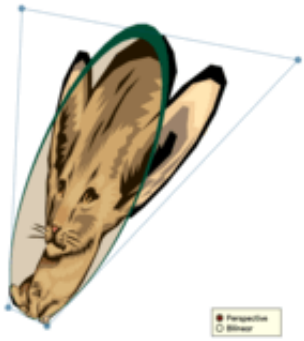
A vector graphic editor of new generation by Pierre Arnaud and Daniel Roux (http://www.opac.ch). The work is in progress. Pierre uses the .NET platform, but all the GUI elements, including menus and buttons are implemented on the basis of **Anti-Grain Geometry**, so that, all the rendering infrastructure is multi-platform. There are no pixel maps at all! All the pictograms are rendered with **Anti-Grain Geometry** from their vectorial representation.



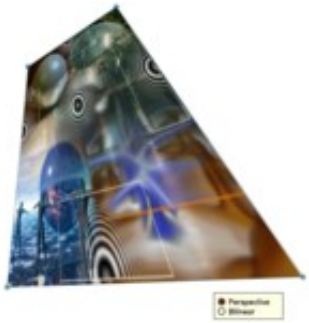
This “sphere” is rendered with color gradients only. Initially there was an idea to compensate so called Mach Bands effect. To do so I added a gradient profile functor. Then the concept was extended to set a color profile. As a result you can render simple geometrical objects in 2D looking like 3D ones.



2D Bilinear transformations in a quadrangle



2D Perspective transformations in a quadrangle



Perspective transformations can be applied to an image.

Bilinear transformations of the image.

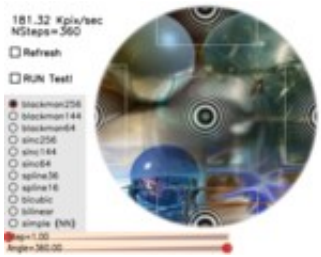
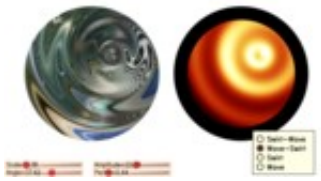


Image rotation with different interpolation filters. This image has been consecutively rotated 360 times with 1 degree step. The image doesn't look too much spoiled because of using the Sinc-Blackman 16x16 filter (i.e. 256 source pixels per one destination pixel). However, the speed is about 180 thousand destination pixels per second on a typical P4 2.0 GHz. For the comparison there is a result of the bilinear filter and the Spline16 (modified bicubic filter). Here is the Source, not rotated image.



You can apply custom distortions to images and gradients. In case of images all the interpolation fiters are also applicable.

TO BE CONTINUED...