Hardware Accelerated Canvas

A X11 Canvas layer that supports alpha blending, anti–aliased text and hardware acceleration as well as highly optimized software.

By Carsten Haitzler (The Rasterman)
raster@valinux.com
raster@rasterman.com
VA Linux Systems

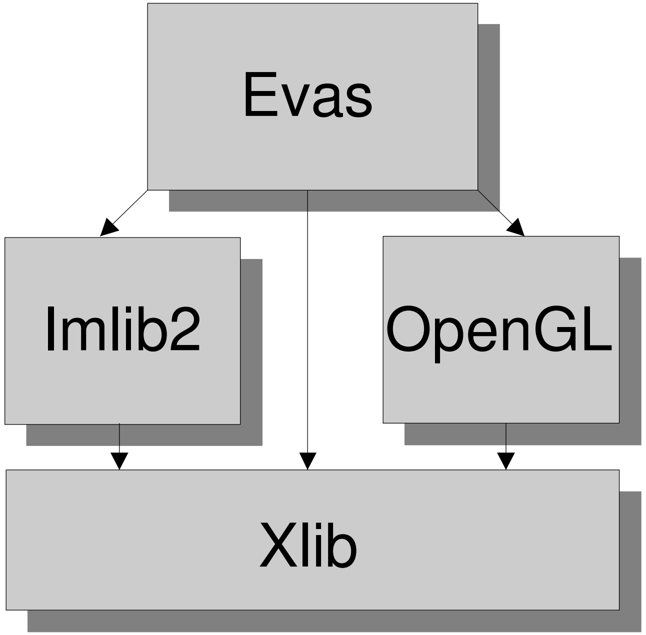
YAC!

There are many Canvas implementations already in existence. The Tk Canvas, GNOME Canvas, Motif Canvas, Java AWT Canvas and a whole host more.

Evas is Yet Another Canvas, and follows many of the same principles as other canvases – except it already has been written to support anti–aliased text, alpha blending and also pipe the rendering via hardware acceleration when desired, and supports the X11 API directly so any X11 application can use it.

Evas builds on top of Imlib2 which does a lot of the dirty work for the software rendering engine and all of the image loading.

Evas' Building Blocks



How do canvases work?

- Object based – application creates objects (Images, Rectangles, Polygons, Lines, Text Strings etc.) and the Canvas manages them.
- Application handles objects on a high level (move, resize, change parameters for object, raise, lower, show, hide etc.).
- Canvas handles the mechanics of doing redraws.
- Canvas is capable (if written well) to optimize this so the manipulation of objects uses as little processing as possible – as well as the redraws.
- As a result the Canvas limits drawing elements to only those supported as objects by the Canvas.

How does Evas work?

Evas never renders anything until you tell it to. This means moving and resizing objects and showing, hiding them etc. takes almost no processing power. It is a state machine internally and state changes are almost zero overhead. Rendering is all the overhead. Example of using evas:

```
[Create Canvas]
[Create Object]
[Move Object somewhere]
[Resize Object]
[Show Object]
[Render Evas]
```

Why do it this way?

This means Evas can defer all rendering till a render call is explicitly called for that Canvas (which in effect "flushes" the changes you've made).

This means that a program can do lots of seemingly useless manipulations to objects and use almost no processing time because Evas minimizes processing to do this. All the smart work is done at render time when it figures out what to and not to render – if indeed anything changed at all.

Rendering Optimizations

Evas calculates rendering optimizations at render time.

It calculates a list of rectangles that "changed" in the canvas. It granulates this to a list of tiles that changed (tiles being 32x32 pixels). Tiles are concatenated into larger rectangles that group contiguous blocks of tiles that need updating to reduce the rectangle count for updating.

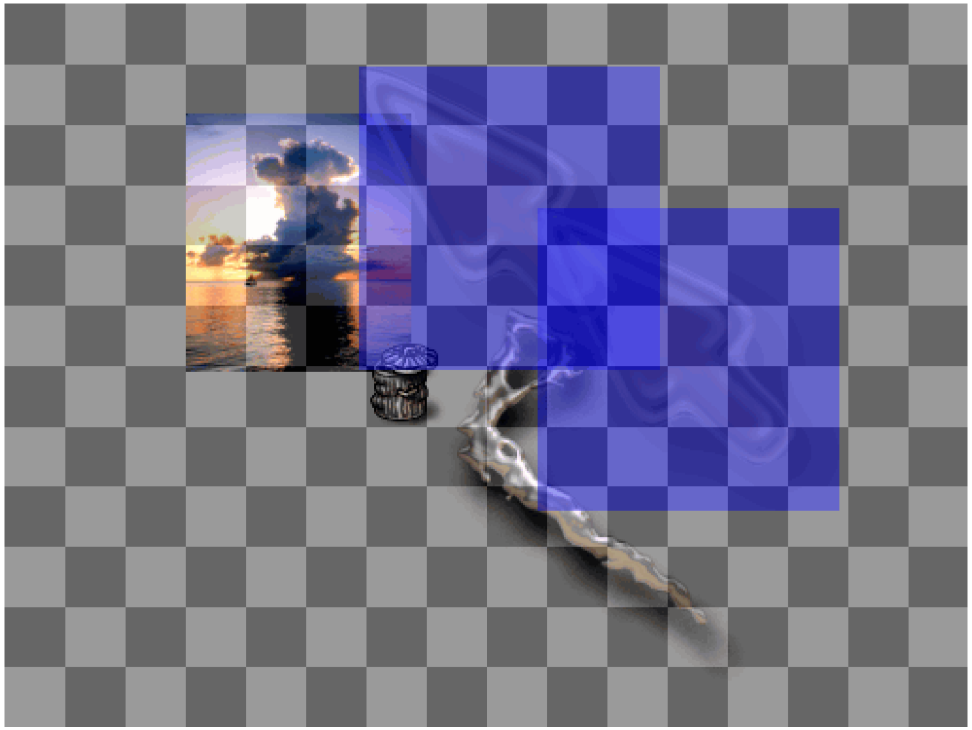
Each "macro" is then rectangle re−rendered off−screen then copied/converted to the destination.

This minimizes the amount of the screen to be redrawn AND also minimizes granularity of the rectangle count (a problem with the shape extension and clipping in X in general).

Example of "Smart" Rendering

Objects occupy their bounding rectangles

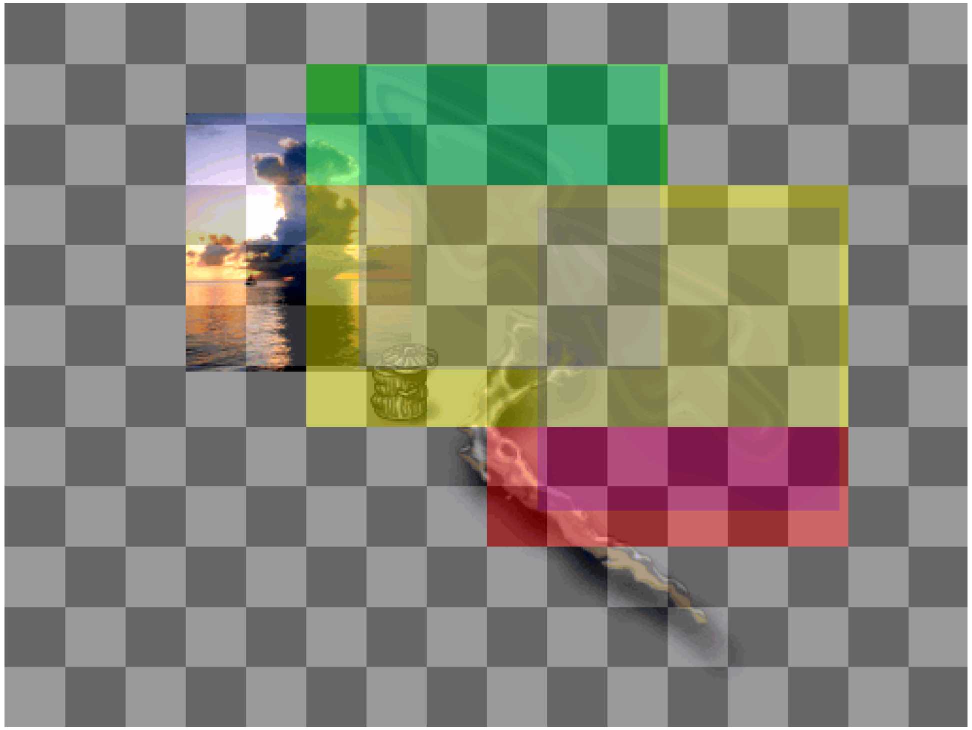
Delta is mapped to a grid of 32x32 pixel tiles



Application moves arrow object

Delta of the change is the bounding rectangles of the 2 previous and current location

Delta is expanded to occupy the tiles and optimized to a minimum number of updates



Update rectangles have no overdraw and are a superset of the delta change

Update rectangles are re–rendered to update canvas contents

Result is canvas with arrow moved with minimum redrawing.

What can you use Evas for?

Diagrams, viewports into graphical data, drawing programs, games, widget sets, user interfaces. Almost anything that does 2D graphics can be done with a canvas – and Evas is capable of that as well as optimizing the rendering and doing all the blending, scaling and anti–aliasing for you. It provides ultra–fast software rendering as well as all the abstraction to letting the hardware do it for you. It can render to in–memory images for saving to output files and much more. It saves lots of time in writing graphics code and makes it trivial.

Evas sources

http://www.enlightenment.org/

http://download.sourceforge.net/enlightenment/evas−0.0.2.tar.gz

http://download.sourceforge.net/enlightenment/imlib2−1.0.0.tar.gz

And now the demo & questions

And now to prove that this is NOT vaporware...

(I wish my laptop had an OpenGL accelerated graphics chipset to show that – trust me on performance reaching 970 frames per second with hardware acceleration – or try it yourself).