GEGL
Generic Graphics Library

Frank Sweetser
WPI Senior Network Engineer
<fs@wpi.edu>

What Is GEGL?

- GEneric Graphics Library
- Originally intended as new GIMP core
- Lets you do stuff to pictures
- XML based image creation and filtering
  - Image(s) + XML => new image

What Makes It So Special?

A GEGL document is an XML encoded directed acyclic graph, with images represented as nodes and operations as edges

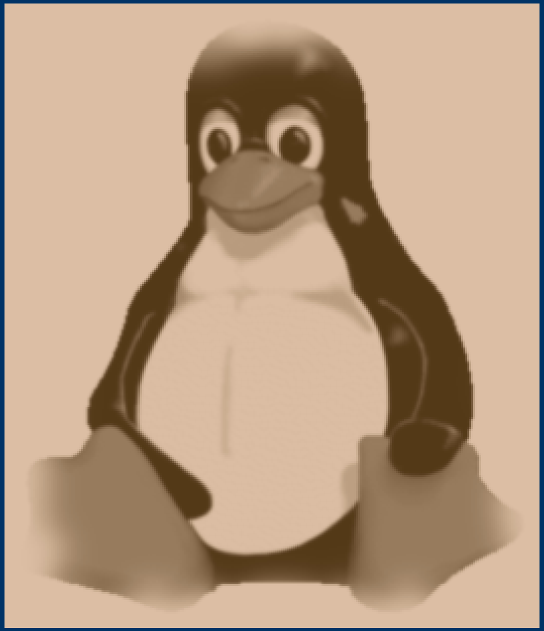
So... why should you care?

Traditional Image Editing

Original

Traditional Image Editing

Original



Old Photo

Traditional Image Editing

Original

Old Photo

Coffee Stains

Three Months Later...

- New “Mosaic” filter released!
- Client hears about it, wants it used
- Options:

Three Months Later...

- New “Mosaic” filter released!
- Client hears about it, wants it used
- Options:
  - Undo coffee stain, undo photo, apply mosaic, reapply photo and coffee stain with old values (you kept notes about what values you used, right?)

Three Months Later...

- New “Mosaic” filter released!
- Client hears about it, wants it used
- Options:
  - Undo coffee stain, undo photo, apply mosaic, reapply photo and coffee stain with old values (you kept notes about what values you used, right?)
  - Take original image (if you have it), apply mosaic, re-do everything else (notes again!)

Three Months Later...

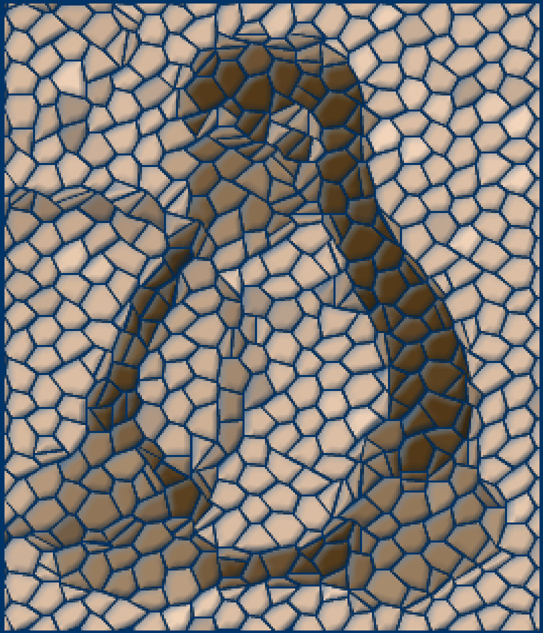
- New “Mosaic” filter released!
- Client hears about it, wants it used
- Options:
  - Undo coffee stain, undo photo, apply mosaic, reapply photo and coffee stain with old values (you kept notes about what values you used, right?)
  - Take original image (if you have it), apply mosaic, re-do everything else (notes again!)
  - Apply mosaic filter and hope...

Adding Mosaic

Coffee Stains

Adding Mosaic

Coffee Stains



Mosaic

Not Best Results...

- Order of operations matters
  - Background image was tiled
  - Coffee stains were tiled
  - Tile lines sharp, missing old photo effects
- What we really want is a transcript of exactly how the image was created:
  - Complete
  - Repeatable
  - Editable

Results vs Workflow

- Traditional image workflow saves the final result
- GEGL saves entire workflow
  - All original images, unmodified
  - All operations performed
  - Non-destructive editing
  - Final output

Adding Mosaic With GEGL

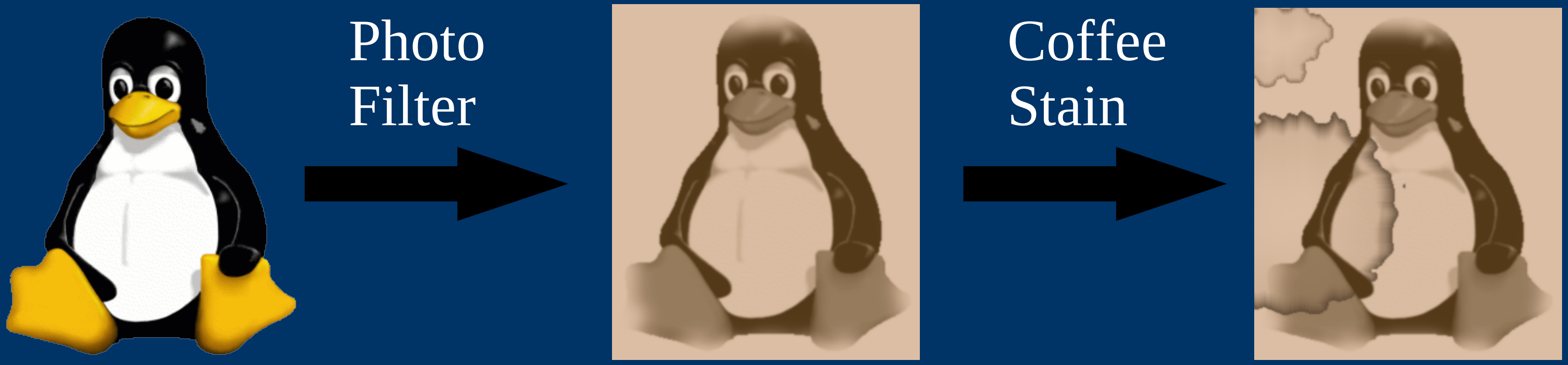
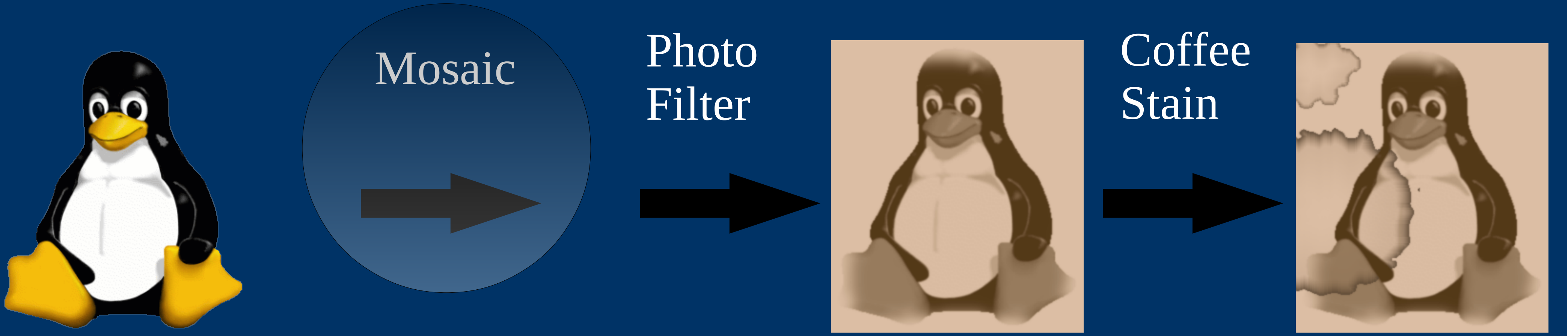


Photo Filter

Coffee Stain

Adding Mosaic With GEGL

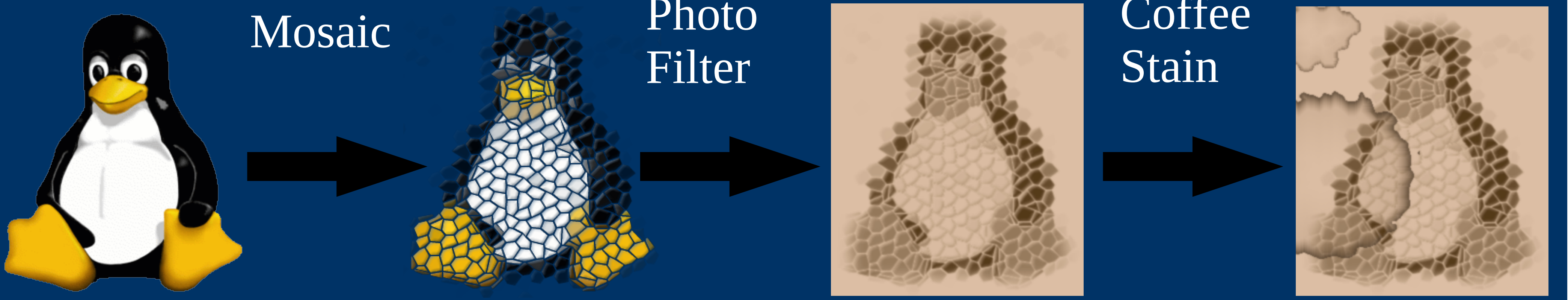


Mosaic

Photo Filter

Coffee Stain

Adding Mosaic With GEGL

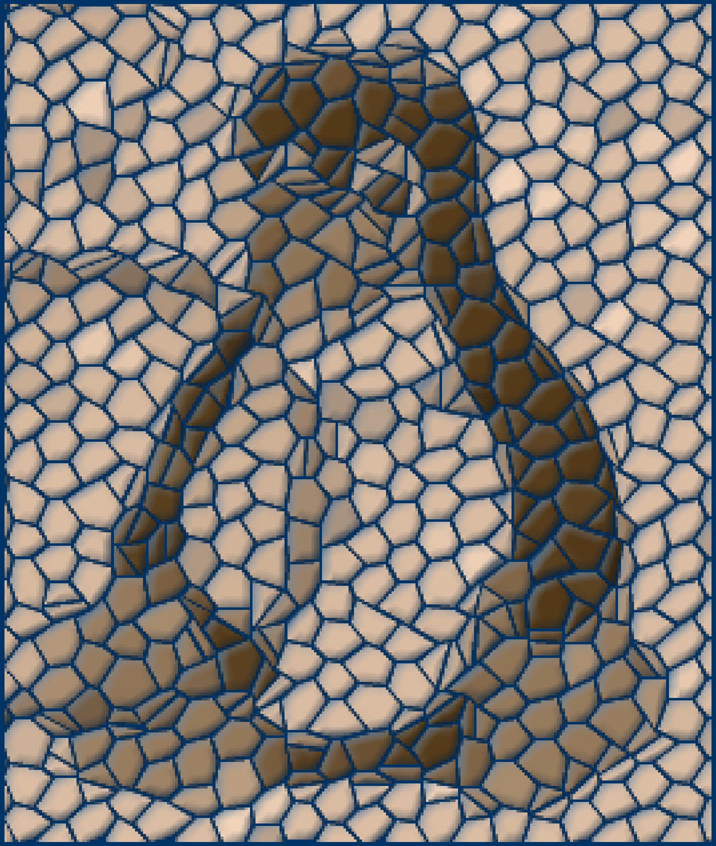


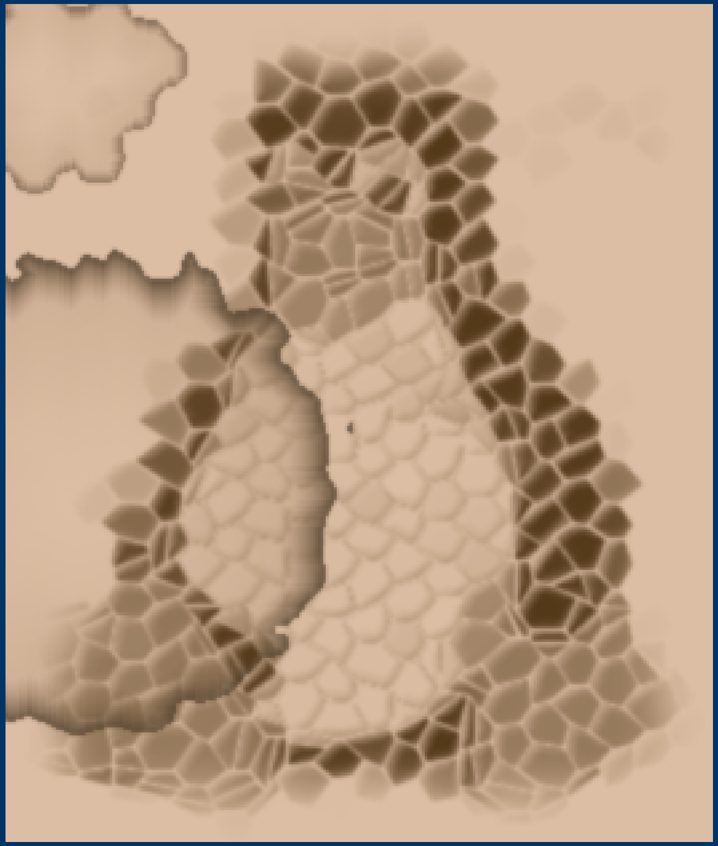
Mosaic

Photo Filter

Coffee Stain

Comparing the Results





Sample XML

```xml
<node>
  <node operation='shift'>
    <params>
      <param name='x'>10.000000</param>
      <param name='y'>280.000000</param>
    </params>
  </node>
  <node operation="load" id="clone0">
    <params>
      <param name="path">car.jpg</param>
    </params>
  </node>
</node>
```

Running GEGL

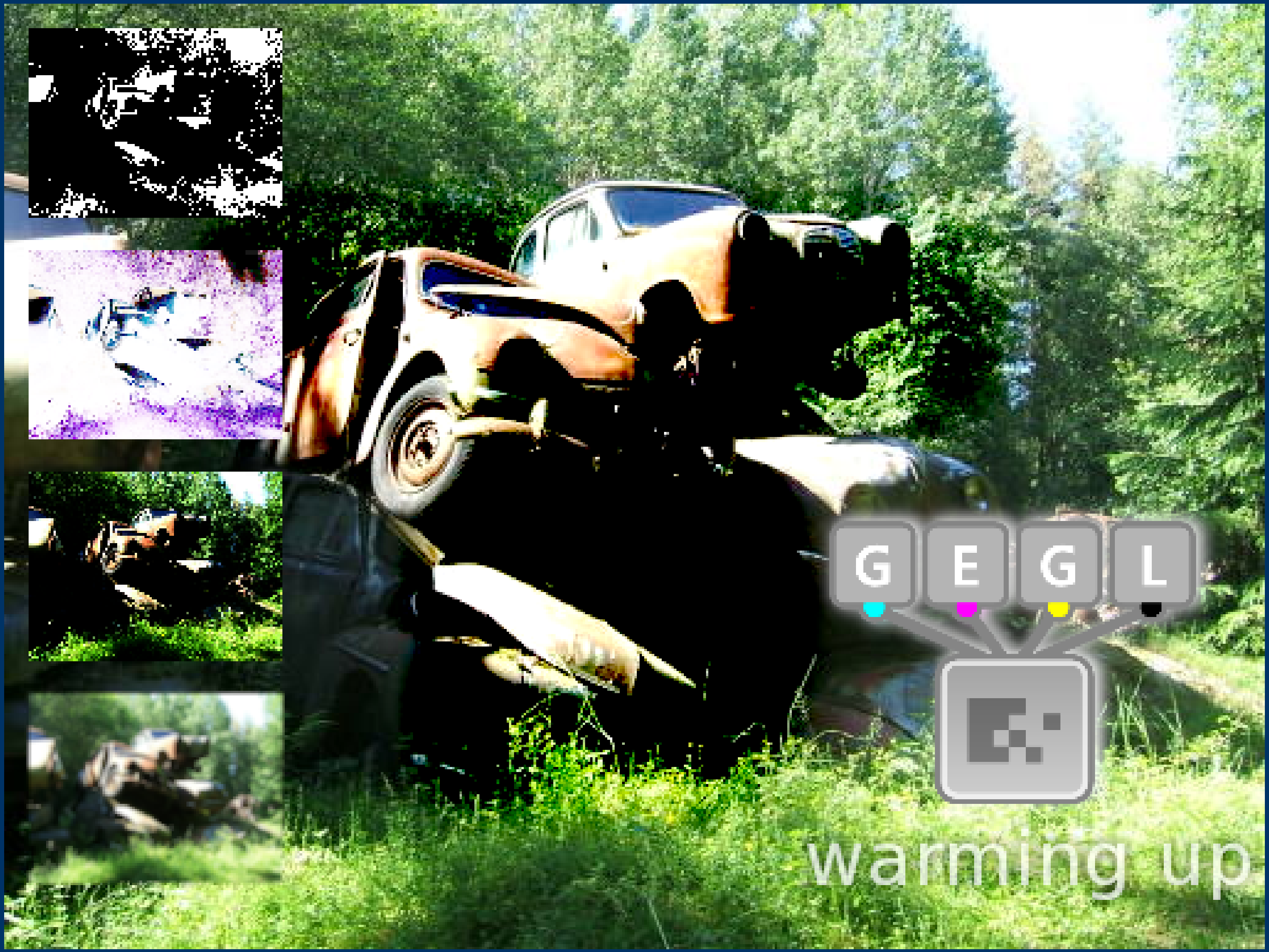
- Simple command line

```
gegl <file.xml> -o <output.png>
```

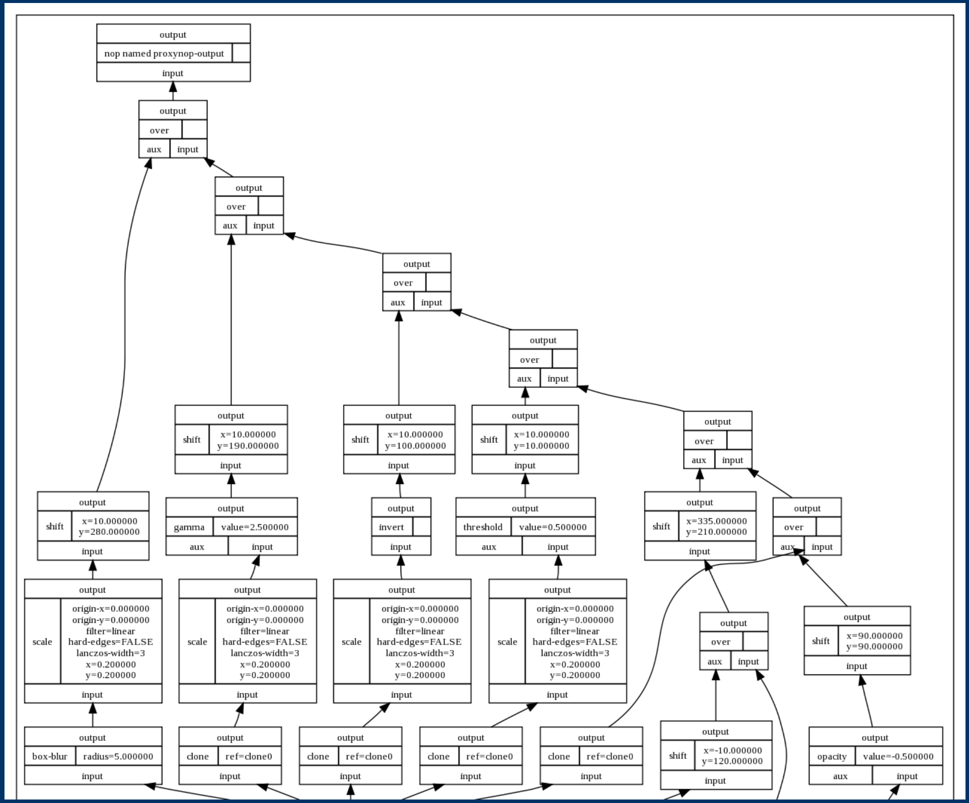
- Virtually everything controlled by XML
- --dot produces graphvis of workflow
- Current output format is PNG

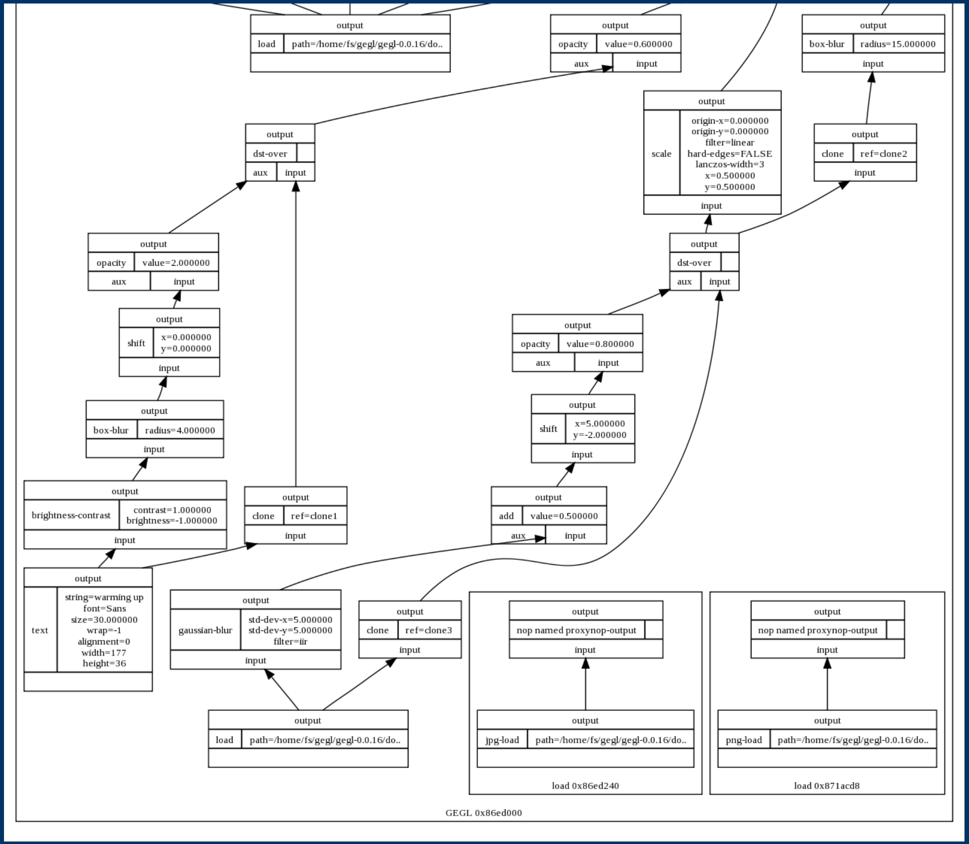
Sample - Input Files

Sample - Output



Sample - Workflow


output | nop named proxynop-output | input
output | over | aux | input
output | over | aux | input
output | over | aux | input
output | over | aux | input
output | shift | x=10.000000 y=190.000000 | input
output | shift | x=10.000000 y=100.000000 | input
output | shift | x=10.000000 y=10.000000 | input
output | over | aux | input
output | shift | x=10.000000 y=280.000000 | input
output | gamma | value=2.500000 | aux | input
output | invert | input
output | threshold | value=0.500000 | aux | input
output | shift | x=335.000000 y=210.000000 | input
output | over | aux | input
output | scale | origin-x=0.000000 origin-y=0.000000 filter=linear hard-edges=FALSE lanczos-width=3 x=0.200000 y=0.200000 | input
output | scale | origin-x=0.000000 origin-y=0.000000 filter=linear hard-edges=FALSE lanczos-width=3 x=0.200000 y=0.200000 | input
output | scale | origin-x=0.000000 origin-y=0.000000 filter=linear hard-edges=FALSE lanczos-width=3 x=0.200000 y=0.200000 | input
output | scale | origin-x=0.000000 origin-y=0.000000 filter=linear hard-edges=FALSE lanczos-width=3 x=0.200000 y=0.200000 | input
output | over | aux | input
output | shift | x=90.000000 y=90.000000 | input
output | box-blur | radius=5.000000 | input
output | clone | ref=clone0 | input
output | clone | ref=clone0 | input
output | clone | ref=clone0 | input
output | clone | ref=clone0 | input
output | shift | x=-10.000000 y=120.000000 | input
output | opacity | value=-0.500000 | aux | input


output | load | path=/home/fs/gegl/gegl-0.0.16/do..
output | opacity | value=0.600000 | aux | input
output | box-blur | radius=15.000000 | input
output | scale | origin-x=0.000000 origin-y=0.000000 filter=linear hard-edges=FALSE lanczos-width=3 x=0.500000 y=0.500000 | input
output | dst-over | aux | input
output | clone | ref=clone2 | input
output | opacity | value=2.000000 | aux | input
output | dst-over | aux | input
output | shift | x=0.000000 y=0.000000 | input
output | opacity | value=0.800000 | aux | input
output | box-blur | radius=4.000000 | input
output | shift | x=5.000000 y=-2.000000 | input
output | brightness-contrast | contrast=1.000000 brightness=-1.000000 | input
output | clone | ref=clone1 | input
output | add | value=0.500000 | aux | input
output | text | string=warming up font=Sans size=30.000000 wrap=-1 alignment=0 width=177 height=36
output | gaussian-blur | std-dev-x=5.000000 std-dev-y=5.000000 filter=iir | input
output | clone | ref=clone3 | input
output | nop named proxynop-output | input
output | nop named proxynop-output | input
output | load | path=/home/fs/gegl/gegl-0.0.16/do..
output | jpg-load | path=/home/fs/gegl/gegl-0.0.16/do..
output | png-load | path=/home/fs/gegl/gegl-0.0.16/do..
load 0x86ed240
load 0x871acd8
GEGL 0x86ed000

GEGL In Gimp

- For those who don't keep your grocery lists in XML...
- Gimp 2.5 beta
  - First steps of integrating GEGL into core
  - Color tools
  - GEGL “pass-through” experimental tool
  - Much more work to fully leverage GEGL

Gimp + GEGL vs Photoshop

- Recent versions of Photoshop have limited “smart filter” operations
  - Sharpen
  - Blur
  - Adjust levels, intensity, contrast
- GEGL provides potential to model *everything*
  - Cut/paste/copy/crop
  - Clone tool (photo repair)
  - Etc...

Performance Optimizations

- Constantly calculating entire graph gets expensive quickly
- Cheat!
  - Per-node caching
  - Efficient subregion evaluation
  - Don't recalculate anything you don't have to
  - Allows (relatively) efficient processing of images larger than RAM

Highly Extensible

- Language bindings
  - C
  - C#
  - Python
  - Ruby
- Plug-in API
- Anything that can read and write XML

Getting GEGL

- GEGL:
  - Current snapshot is 0.0.16
  - http://www.gegl.org
  - Currently shipping in Fedora 9
- Gimp 2.5 beta
  - http://www.gimp.org
  - Source code release only

Conclusions...

- Still long way to go
  - Usability
  - Complete Gimp integration
  - Ability to perform simple ops without XML
- Huge benefits for large projects
  - Photo restoration/retouching
  - Complex composite images
  - “Small” changes to old projects
- Potential to set Gimp apart from everything else out there

Questions?
Comments?