GEGL

Generic Graphics Library

Victor M. de Araujo Oliveira

May 2, 2011

What is it?

GEGL is a graph based image processing framework.
GEGL's main purpose is to be the default way GIMP manipulate and process images. But it can be used as a stand-alone lib as well.

- Floating point images
- Supports many image formats
- Larger than RAM images (automatic disk swapping)
- Python, Ruby and Vala Bindings
- Lots of filters

How to Install

git clone git://git.gnome.org/gegl

- Ubuntu
  sudo apt-get install gegl

How to Install

git clone git://git.gnome.org/gegl

- Ubuntu
  sudo apt-get install gegl
- ./configure;make;sudo make install

GEGL in GIMP

{Open GIMP and introduce GEGL}

- How to enable and use GEGL.
- Show and explain some filters.

Tiled Buffers

GEGL uses tiled buffers. This means some operations can be well-supported and others doesn't map well.

Tiled Buffers

GEGL uses tiled buffers. This means some operations can be well-supported and others doesn't map well.

- Brightness-Contrast

Tiled Buffers

GEGL uses tiled buffers. This means some operations can be well-supported and others doesn't map well.

- Brightness-Contrast
  Processing a pixel only needs information about that pixel - OK

Tiled Buffers

GEGL uses tiled buffers. This means some operations can be well-supported and others doesn't map well.

- Brightness-Contrast
  Processing a pixel only needs information about that pixel - OK
- Blur

Tiled Buffers

GEGL uses tiled buffers. This means some operations can be well-supported and others doesn't map well.

- Brightness-Contrast
  Processing a pixel only needs information about that pixel - OK
- Blur
  Processing a pixel only needs information about that pixel and a fixed neighborhood around it - OK

Tiled Buffers

GEGL uses tiled buffers. This means some operations can be well-supported and others doesn't map well.

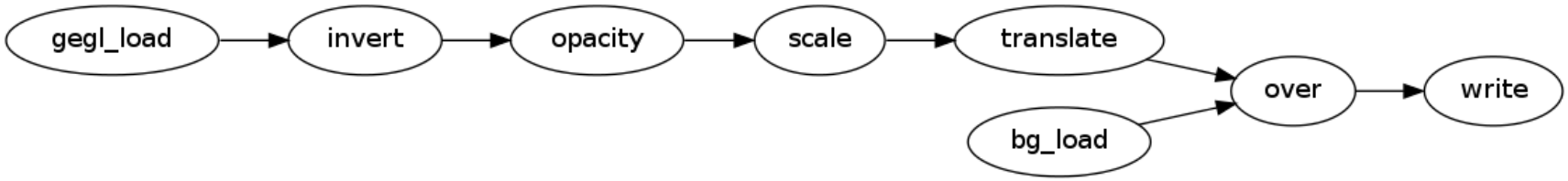
- Brightness-Contrast
  Processing a pixel only needs information about that pixel - OK
- Blur
  Processing a pixel only needs information about that pixel and a fixed neighborhood around it - OK
- Watershed

Tiled Buffers

GEGL uses tiled buffers. This means some operations can be well-supported and others doesn't map well.

- Brightness-Contrast
  Processing a pixel only needs information about that pixel - OK
- Blur
  Processing a pixel only needs information about that pixel and a fixed neighborhood around it - OK
- Watershed
  Processing a pixel needs information from many others random ones - NO!

Processing Graph



Nodes represent operations in the image.
Edges link the output of a filter with the input of the next one.

Creating a Node

```
GeglNode *scale = gegl_node_new_child (gegl,
                                       "operation", "gegl:scale",
                                       "x", 2.0,
                                       "y", 2.0,
                                       NULL);
```

Linking nodes

```
gegl_node_link_many (gegl_load, invert, opacity, scale, translate, NULL);
gegl_node_link_many (bg_load, over, write, NULL);
gegl_node_connect_to (translate, "output", over, "aux");
```

By default, the "output" attribute of a node is connected to the "input" of another one, but some nodes [like gegl:layer, gegl:over] can be have many input attributes.

Processing

```
gegl_node_process (write);
```

In order to evaluate 'write' node, GEGL evaluates all dependency nodes.

Some real code!

{Let's see some GEGL code - ops/test.c}

XML Description

A Processing Graph can be described by a XML where tags are operations.
{Let's see an XML example - ops/test.xml}

Python Bindings

We can use GEGL in Python!
Very useful to:

Python Bindings

We can use GEGL in Python!
Very useful to:

- Test composition of operations in an interactive way.

Python Bindings

We can use GEGL in Python!
Very useful to:

- Test composition of operations in an interactive way.
- Use Python's flexive way to manipulate strings and XML files to generate custom filters.

Python Bindings

We can use GEGL in Python!
Very useful to:

- Test composition of operations in an interactive way.
- Use Python's flexive way to manipulate strings and XML files to generate custom filters.
- GEGL can be easily used in the Python ecosystem [like a webserver that generates images according user data].

Python Bindings

We can use GEGL in Python!
Very useful to:

- Test composition of operations in an interactive way.
- Use Python's flexive way to manipulate strings and XML files to generate custom filters.
- GEGL can be easily used in the Python ecosystem [like a webserver that generates images according user data].
- Infinite possibilities...

{Let's create the example from the previous section interactively in a Python shell}

Operations

- GeglOperationPointFilter
  - Brightness-Contrast
  - Threshold
  - Color-to-Gray
- GeglOperationAreaFilter
  - Motion Blur
  - Sobel Edge Detection
- GeglOperationSource
  - Load Image
- GeglOperationSink
  - Save Image
  - Display Image

{Let's implement a Brightness-Contrast operation.}

GSoC - OpenCL in GEGL

OpenCL (Open Computing Language) is a framework for writing programs that execute across heterogeneous platforms consisting of CPUs, GPUs, and other processors.
My proposal is about making possible to write GEGL plug-ins in OpenCL. But the main point of the work is extending GeglBuffer to automatically make the memory transferences between the CPU and GPU.
Also, I'm going to implementing some operations in OpenCL.

Example of OpenCL code

```
__kernel square(
    __global float *input,
    __global float *output,
    const unsigned int count)
{
    int i = get_global_id(0);
    if (i < count)
        output[i] = input[i] * input[i];
}
```

Questions?