OpenGL GPU-based Buffer and Operations in GEGL

Proposed by Jerson Michael Perpetua for Google Summer of Code 2009

Abstract:

GEGL (Generic Graphics Library) is a new back-end to replace GIMP's old code for handling various image processing tasks. It uses a directed acyclic graph (DAG) to represent image processing operations. GEGL integration will introduce much anticipated features like layer effects, layer groups, adjustment layers, CMYK and high bit-depth support (among other things) to GIMP.

GPUs nowadays are very powerful. Capable of massively accelerating graphics tasks through parallelization, there is an ongoing movement to harness this power in general computing. However, because of the nature of GPU parallelization, not all code can be executed in the GPU. GEGL's architecture of pixel operations yields itself to GPU parallelization, making it a good candidate for GPU acceleration.

Benefits to the GIMP Community:

With the ever increasing trend of video card performance these days and with the integration of GIMP's new graphics back-end, GEGL, artists will be able to take advantage of the best features & performance in GIMP through OpenGL/GPU support.

Deliverables:

The GeglBuffer module will be extended to support storing image data inside GPU memory. Afterwhich, pixel shaders will be implemented for both GeglOperationPointComposer and GeglOperationPointFilter. As an option, GPU support for several other GeglOperations and resamplers will be realized if time permits.

As the purpose of this project is to extend GEGL's internals to support GPU acceleration, no graphical user interfaces (GUI) will be developed for neither GIMP nor GEGL.

Functionality Details:

Because this project aims to modify GIMP's infrastructure (i.e. GEGL), it will not introduce any direct improvements to user interaction. However, by the time GEGL is fully integrated to GIMP, users will notice speed improvements with regards to pixel operations as, AFAIK, GIMP's current core only supports CPU acceleration. For example, hiding or showing a layer or performing color operations like changing the brightness of an image will take relatively less time.

Implementation Details:

This is how I plan to implement GPU support for GEGL:

- Extend the current GeglBuffer implementation to store a copy of image buffers/tiles in GPU memory (e.g. as textures). The image data in the GPU memory will be accelerated as they are already cached in the GPU as opposed to the ones that are stored in primary memory.
- Extend both GeglOperationPointComposer and GeglOperationPointFilter using pixel shaders that operate on cached GeglBuffer data on the GPU memory.
- (optional) Implement a hardware bilinear sampler to get a feel of how to implement a GEGL resampler through OpenGL shaders.

- (optional) Examine and extend some other operations to support GPU acceleration.
- (optional) Extend GEGL's current resampler (nohalo) implementation to support GPU acceleration.

Development Methodology:

Since this is a challenging topic (at least, for me), I will spend much of my time modeling & implementing the relevant interfaces to provide GPU support for GeglBuffer. This is the hardest part. GPU support inside GeglBuffer will provide infrastracture for the GEGL operations to store image data in GPU memory. After this is done, I will start adding GPU support for GEGL operations using OpenGL shaders.

During development, mentor feedback is paramount. I will collaborate with my mentor organization on a daily basis and blog about the design decisions along the way. Source code will be hosted on an online DVCS and performance benchmarks will be blogged as soon as benchmarkable features are implemented.

Project Schedule:

I will spend the first few weeks until the start of the coding proper to study OpenGL & the GEGL source code. The bulk of the development period will be spent modeling and implementing GPU support for GeglBuffer until the midterms.

After the midterms, I will start implementing OpenGL shaders for GeglOperationPointComposer and GeglOperationPointFilter to pave way for GPU support to other GeglOperations. The remaining time will be used to implement some optional features (see implementation details above), write tests and fix bugs.

Bio:

I am a BS Computer Science student from the University of the Philippines Mindanao. My native language is Filipino (Tagalog) but I can speak and write English quite fluently. I am an experienced C/C++ developer, working for software ranging from school projects to small to mid-sized business applications. Additionally, I am very comfortable with object-oriented languages like C++ (as previously mentioned) and Java.

Working as a part-time student, I have implemented other students' theses. Examples are a neural network, an economic model and time-tabling problems. I have also developed a parser (a necessary component of a compiler), a kiosk network-restriction application and a case-management application for our Municipal Trial Court. Earlier, I have been able to implement a humble graphics library to display bitmaps and 3D polygons in DOS.

Though I have little experience with GTK+ & GObject, I am confident that I can learn as I develop. This last bit springs from my passion in Free Software and programming in general. I am quite resourceful and can work with the least amount of supervision. Also, if ever I get accepted, I will be working full-time for this project as I have no other commitments this summer.

Being quite a bit of a graphics enthusiast myself, I am interested in the unique fusion of art and technology. I have been a long-time GIMP user and wanted to contribute code-wise. In fact, I have used GIMP in a free-lance, part-time web graphics job before.

My interests are in the field of 3D graphics (OpenGL, GPGPU, etc.), 2D graphics and interface usability. I have prepared for this year's GSoC by learning the required technologies (OpenGL, GTK+, GObject, etc.) and reading GIMP and GEGL's source code.

I have been intending to participate in the Free Software movement by contributing code. This is a good opportunity for me to get started and to familiarize myself with the internal workings of GIMP. Also, to be frank, the stipend will be of much help as I prepare for an independent life ahead doing free-lance Free Software development.

I usually use Daerd as nick in IRC.

(document template taken from Blender Foundation's application form)