WebGLot: High-Performance Visualization in the Browser

Dan Lecocq * Markus Hadwiger † Alyn Rockwood ‡

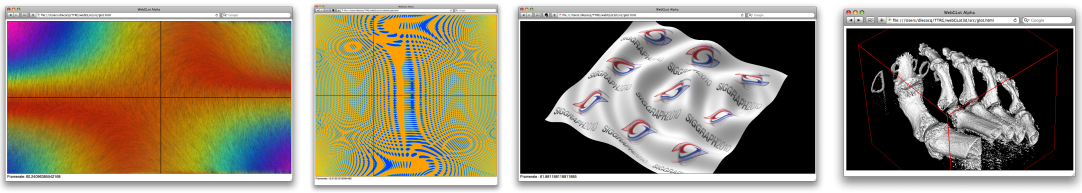

Figure 1: *A few capabilities of WebGLot: (a) a visualization of a flow-field using noise texture advection, (b) the solution to a Poisson equation calculated on a 2200x2200 mesh with a high-order stencil (c) a texture-mapped surface defined by a function of x, y, and t (d) raycasting an isosurface from CT scan data of a foot.*

Keywords: real-time, visualization, mathematical rendering, volume rendering, gpgpu

1 Introduction

Rendering mathematical primitives should be as easy as describing them. There are several tools available for doing much of this but there is still room for improvement in interactivity, flexibility and robustness. WebGL + plot = WebGLot seeks to fill this role, providing a library for function plotting and data visualization while maintaining flexibly and performance from within the browser [Khronos b][Khronos a].

Among its strengths is its ability to offload intensive algorithms effectively onto the GPU while providing a scripting interface for easy exploration or application-building. The fact that it runs within the browser makes it a good candidate for cross-platform portability and ease of update. There has already been a great deal of work in the field of streaming and asynchronous communication, which solve many problems surrounding interactive web use. WebGLot is poised to build upon much of that existing work to the end of streaming field data or simulation results to a remote machines for live review and experimentation.

2 Applicability

All major modern browsers support JavaScript (and many are getting quite fast interpreters). This means that WebGL can in principle run on any system on a number of devices, and WebGLot seeks to use it to be a highly portable and ubiquitously supported visualization and plotting library. Researchers often spend a great deal of time struggling to get application-specific code to compile on a new machine; By embracing a very widely-used distribution format we can eliminate much of this difficulty.

The only additional requirements are to have a graphics card supporting OpenGL ES 2.0 or better, and a reasonable OpenGL driver. Both of these are often requirements for visualization for most research applications anyway.

By exploiting existing streaming techniques in JavaScript, we are able to easily support a number of important applications for streaming data visualization. From off-site visualization of cluster- or supercomputer-run simulations, to monitoring data from sensor networks in real time. We envision it providing support for a large number of such applications of growing importance.

This also allows for visualization techniques and applications to be decoupled. We augmented an existing MPI-based n-body simulation with a line to transmit intermediate results to a server instead of outputting to a file. In an afternoon, we developed a WebGLot application to visualize the results as they were calculated. APE (the Ajax Push Engine) does not require clients to poll a server for updates, but rather they are notified when new data arrives. *With minimal code changes, existing simulations can be augmented with streaming visualization.*

WebGLot could provide graphical and visualization support for e-learning projects, allowing students to explore difficult or unintuitive phenomena. From understanding complex data without needing to install highly specific applications to provide a way to experiment with mathematical concepts. Many computers have relatively high-powered graphics cards when compared to the machine's CPU and many medium-sized compute-intensive tasks can find life on the GPU. With WebGLot we were able to achieve a peak performance of 24 gigaflops on a NVIDIA 9400m (45% efficiency) with a 2D Jacobi solver. WebGL has the potential to serve as a delivery mechanism for HPC kernel use, and advanced visualization techniques. *We aim to lower the barrier for using advanced techniques in visualization and GPGPU.*

3 Goals

WebGLot seeks to be extremely easy-to-use, and only requires basic knowledge of JavaScript and familiarity with desired visualization techniques. It's flexible enough for a wide variety of applications, but robust enough for computation-intensive work. Adjacent to streaming technologies it enjoys easy support for streaming, while requiring only JavaScript and a graphics card.

References

KHRONOS. Webgl public wiki. `http://www.khronos.org/webgl/wiki/Main_Page`.

KHRONOS. Webgl specification. `https://cvs.khronos.org/svn/repos/registry/trunk/public/webgl/doc/spec/WebGL-spec.html`.

ROETTGER, S. The volume library. `http://www9.informatik.uni-erlangen.de/External/vollib/`.

*email: dan.lecocq@kaust.edu.sa
†email: markus.hadwiger@kaust.edu.sa
‡email: alyn.rockwood@kaust.edu.sa

SIGGRAPH 2010, Los Angeles, California, July 25 – 29, 2010.
ISBN 978-1-4503-0210-4/10/0007