Evolving a Rapid Prototyping Environment for Visually and Analytically Exploring Large-Scale Linked Open Data

Marc Downie, Paul Kaiser
Open Ended Group
(marc@openendedgroup.com)

Dylan Enloe, Peter Fox, James Hendler
Tetherless World Constellation/RPI

Eric Ameres, Johannes Goebel,
Experimental Media and Performing Arts Center/RPI

ABSTRACT

The lack of development environments for interdisciplinary research conducted on large-scale datasets hampers research at every stage. Projects incur large startup costs as disparate infrastructure is assembled; experimentation slows when software components and environment are mismatched for specific research tasks; and findings are disseminated in forms that are hard to examine, learn from, and reuse. Behind these problems is a common cause — the lack of good tools. When large, heterogeneous and distributed data is added to the equation, further frustration, at the least, ensues. As a result users frustrated with their tools; and findings are disseminated in forms that are hard to examine, learn from, and reuse. Thus tools of digital artists to bear on the aforementioned data analysis and visualization challenges. Here we report on the current state of progress in adapting Field for large-scale, web-based scientific data analysis and visualization with an emphasis on Linked Open Data [1] and especially the current data hosted by RPI [2].

Keywords: Open-source, rapid development, visualization, analysis, linked data, world-wide-web.


1 ABOUT FIELD: INTEGRATED META-AUTHORING ENVIRONMENT

Field [3] is a stable open-source programming environment, architected in such a way as to support interdisciplinary projects that call upon domain-specific tools, libraries, and languages. It comprises a minimal core and a powerful plug-in system. Field provides a hybrid text editor that can incorporate text-based programming at the same time with graphical user-interface elements. Its flexible and extensible interface provides space as necessary for notation, visualization, interface construction, and debugging. In addition, it provides an advanced GPU-accelerated graphics system. Since Field was created in the context of widely divergent interdisciplinary projects, its aim is to give its users not only the ability to work rapidly, but to shape their Field environment extensively and flexibly for their own demands. Field thus serves as a meta-authoring environment. Field takes a promiscuous approach to other tools, libraries, and languages. This stems from the realization that since the best work available in traditional domain-specific disciplines is already coded, such code is better appropriated than constructed again. Thus Field can easily wrap up, examine and choreograph other programs and libraries.

2 NEW PROGRESS

The first is a new addition to Field’s 3D graphics system — directly embedded web browser components. By embedding WebKit into Field we are now able to draw 3d geometry textured with rendered web content. By remapping mouse events from window-space into geometry-space we are able to interact with these surfaces as if they were tabs in a running web-browser. By inspecting the Document Object Model of the underlying HTML content we are able to render overlays and visualizations of web content along side these 3d windows. In the poster presentation we will present a playground for radically extended Web-browsing.

2.1 Javascript extensions for Field

To forge the most immediate connection between Field and the Linked Open Data capability, we extended Field to include full support for remote, web-browser-hosted JavaScript execution. The most immediate benefit seems to be development speed. Unlike traditional JavaScript environments this Online Plugin for Field allows JavaScript programs to be executed and constructed piecemeal — built up and tuned interactively — while keeping the execution of the JavaScript online and in web-browsers (see Fig. 1). Of course, unlike Field’s previous language and runtime bridges, in the online case multiple execution hosts may be available — each web-browser visiting a page being served by Field’s

Figure 1. Generic chart plotting interface, running in a browser, using the Javascript extensions.
new embedded web server. These execution hosts come and go over the lifecycle of the server. Field solves this problem by using a form of journaled XML long-polling, allowing browsers that connect after Field has already sent code to other browsers to catch up, while keeping browsers that are up to date primed to receive the next piece of code to execute. This allows the Field to be used as a complete development environment for the authorship of online visualizations. Similar to the existing bridges that Field supports, this plugin connects both language and execution environment — auto completion, introspection and Field’s timeline and animation concepts are available in in-browser JavaScript. This suggests the possibility not only of much faster exploration and development of the kinds of visualizations but also of new kinds of guided or “broadcast” visualizations — where the animation and timeline concepts of Field are used to choreograph the delivery of streams of executable code to browsers – an example is shown in Fig. 1 and more will be demonstrated at the poster session.

2.2 Generic chart-plotting API for Field

Field’s new charting reuses the dynamic scoping techniques used elsewhere in Field’s codebase to create a set of reusable, modifiable class of chart types in the tradition of the Google Motion Chart API — adding, of course, a 3d dimensional layout, canvas and GLSLang shader (see Fig. 2) processing of the geometry being drawn. Animated transitions between potentially completely customized markers and axes are automatically generated by Field’s graphics system.

2.3 Low-level drawing acceleration for the new common case

Field’s graphics system has typically made a strong distinction between static and dynamic geometry (driven by OpenGL’s different handling of these kinds of material) this has resulted in two divergent APIs for creating material to be rendered — one, designed for carefully preparing large amounts of complex, immutable geometry privileges flexibility over speed; the other is a much lower level interface for creating and updating geometry in-place (that will typically be constantly changing during rendering every frame). The result was a near complete revision of the immutability of static geometry in Field — now “static” charts consisting of custom markers still render with minimal CPU load, at interactive frame-rates, on contemporary graphics cards to the ~1000 marker case. Furthermore, when animated, this geometry uses CPU resources that scale only with the amount of moving material and now exhibit vastly less GPU resource and heap churn when it does move.

3 FURTHER WORK

With the release of Field 14 (for Mac OS and Linux), many of the new developments, including many not included here are available to public users of Field. Our present work involves two major components: dissemination and outreach for the new capabilities, and expansion into data mining and analysis plug-ins for the increasingly large datasets. Dissemination includes the development of new video tutorials (hosted by vimeo.com), tutorials at selected professional society meetings such as the American Geophysical Union, and targeted outreach to groups we know are in need of rapid visualization early in the research cycle [4]. The new analysis capabilities are key as the size and complexity of datasets now accessible to Field (linked data) is rapidly increasing and it is time to exploit graph-based analysis opportunities, also in the ‘exploration’ at the speed of creative thought which has driven our developments to date. This project is supported by an NSF EAGER grant IIS-1048440 to RPI.

REFERENCES:

Figure 2. Real-time GLSLang “non-photorealistic” shaders applied to chart geometry.