INTRODUCING THE VISTRAILS PROVENANCE EXPLORER PLUGIN FOR PARAVIEW

In order to analyze and validate various hypotheses, it is necessary to create insightful visualizations of both the simulated processes and observed phenomena, using powerful data analysis and visualization tools like ParaView. But to explore data through visualization, scientists need to go through several steps. They need to select data products and specify series of operations that need to be applied to these data to create appropriate visual representations before they can finally view and analyze the results. Often, insight comes from comparing the results of multiple visualizations. Unfortunately, today this process is far from interactive and contains many error-prone and time-consuming tasks. As a result, the generation and maintenance of visualization data products has become a major bottleneck in the scientific process, hindering not only the ability to mine scientific data, but the actual use of scientific data in every day applications. In particular, scientists and engineers need to expend substantial effort managing data (e.g., scripts that encode computational tasks, raw data, data products, and notes) and record provenance (history) information so that basic questions can be answered, such as: Who created a data product and when? When was it modified and by whom? What was the process used to create the data product? Were two data products derived from the same raw data?

As a step toward addressing this problem, we have developed VisTrails (www.vistrails.org), an open-source, provenance-enabled scientific workflow system that can be combined with a wide range of tools, libraries, and visualization systems. VisTrails provides a comprehensive solution to the problem of managing data and processes used in data analysis and, in particular, data exploration through visualization. By capturing detailed provenance of the visualization and analysis pipelines—both of how they evolved over time and the data products they derive, VisTrails maintains documentation key to preserving data, determining the data's quality and authorship, and reproducing as well as validating results. The system also leverages provenance information to support collaboration, knowledge sharing and re-use, and it also provides intuitive interfaces that simplify common tasks in data exploration. These include: a visual interface for comparing pipelines and their results; a scalable mechanism for the exploration of large parameter spaces; and an analogy operation, which allows users to re-use pipeline refinements as a template for creating new pipelines.

The VisTrails Provenance Explorer plugin for ParaView brings provenance tracking and many of the benefits of provenance to ParaView users. The source code for the plugin can be downloaded from www.vistrails.org; it has been tested under Windows, Mac OS X, and Linux. The plugin is included with the ParaView 3.6 binaries.

USING THE VISTRAILS PROVENANCE EXPLORER PLUGIN FOR EXPLORATORY VISUALIZATION

The VisTrails Provenance Plugin for Paraview automatically and transparently tracks the steps a user followed to create a visualization. In contrast to the traditional undo/redo stack, which is cleared whenever new actions are performed, the plugin captures the complete exploration trail as a user explores different parameters and visualization techniques.

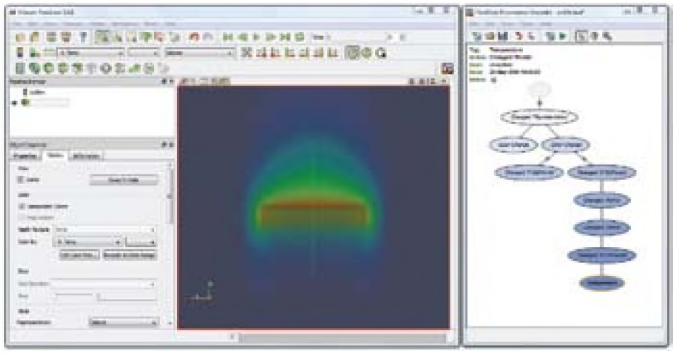

Figure 1. ParaView and the VisTrails Provenance Plugin. The Provenance Plugin (right) captures all the actions a user performs to create visualizations. These actions are stored as a tree that serves as a guide for future exploration, allowing users to return to a previous version in an intuitive way, to undo bad changes, to compare different visualizations, and to be reminded of the actions that led to a particular result.

Consider a user exploring the disk_out_ref.ex2 example dataset, derived by a simulation of the air around a heated rotating disk. The user may load the data and start by experimenting with a volume rendering of the temperature within the domain. After some time spent tweaking the visualization, he decides to examine what the temperature looks like, and labels the current version of the visualization in the Provenance Explorer as "Temperature" (see Figure 1). To look at just the air flow, he undoes all the volume actions, until the pipeline has only the data reader.

He then starts a new visualization by creating a streamline filter. After creating this new visualization and labeling it "Air Flow" (see Figure 2(b)), the user decides that a volume

rendering of the pressure would help complete his view of the simulation. He would like to begin by simply reverting to the temperature visualization, and changing the scalar used for coloring to the pressure variable. Although this would not be possible using the standard undo/redo interface, because the VisTrails plugin never discards old states, the temperature visualization can easily be restored by clicking the corresponding version (node) in the Provenance Explorer window. The user can then continue modifying the volume rendering to explore the pressure data (see Figure 2(c)).

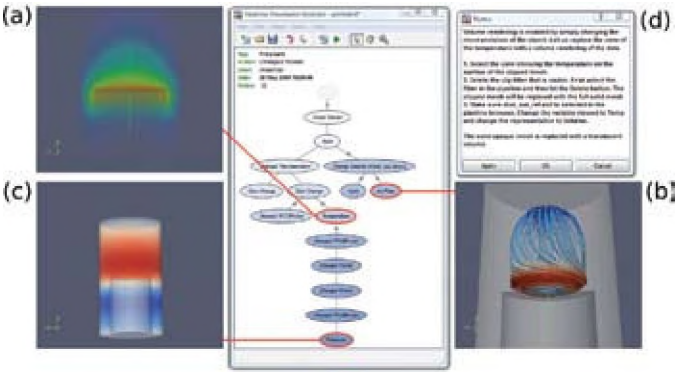

Figure 2. (a), (b) and (c) represent different visualizations generated by a user exploring the disk_out_ref.ex2 example dataset. Besides tracking all visualizations a user explored, how they evolved over time, and enabling them to be reproduced, the VisTrails Provenance Plugin captures additional metadata about the visualizations, including: a descriptive tag and notes (d), the date and time when the visualization was created, and the user who created it.

Once the different visualizations have been created by the user and recorded by the VisTrails plugin, switching between and replaying (re-generating) them to provide comparisons is fairly straightforward. This is different from the multiview visualizations supported by ParaView. The multiview mode in ParaView allows multiple instantiations of a common pipeline with different parameters. Switching between different versions in the Provenance Explorer creates the different pipelines for each version, which can reduce the clutter in the Pipeline Browser when they are significantly different. However, a disadvantage of this mode of comparison is that they cannot be viewed simultaneously side-by-side.

The VisTrails plugin has additional benefits when compared to the undo/redo framework. A tree-based view of the history of actions allows a user to return to a previous version in an intuitive way, undo bad changes, compare different visualizations, and be reminded of the actions that led to a particular result. Also, there is no limit on the number of operations that can be undone, no matter how far back in the history of the visualization they are. Last, but not least, the history is persistent across sessions. The VisTrails plugin can save all of the information needed to restore any state of the visualization in .vt files, which can be reloaded across ParaView sessions and shared among collaborators. This also allows multiple visualizations to be shared with a single file.

In addition to the capturing and replaying capabilities, the VisTrails plugin has several other features that make the full visualization provenance more manageable. Over the course of creating a visualization, many versions may be recorded by the VisTrails plugin. If the user often tries something and then reverts back to an old version, the version tree in the Provenance Explorer will also have a large number of branches, many of which will be "dead ends." To remove clutter from the window, the user can select branches to be hidden. This does not actually delete them---it just removes them from the view. The full version tree can be restored at any time. This makes browsing the version tree easier, since only the most significant versions are visible.

Each version recorded by the plugin has several annotations associated with it: the user that created it, the time and date that it was created, and the name of the action (see Figure 2). By default, each version is labeled in the Provenance Explorer window with the action name, as reported by ParaView. The user may additionally tag a version with a name to override this label, which makes key versions of the visualization easy to find among the others. There is also a field for notes where the user can leave comments, either for themselves, or for collaborators that they wish to share their visualizations with.

The display of the nodes of the version tree in the Provenance Explorer window also makes use of these annotations to give some immediate cues to their provenance. The versions created by the current user are colored blue to distinguish them from versions created by other users. Also, new versions use bolder coloring than old versions, making more recent changes stand out as well.

ACKNOWLEDGEMENTS

The VisTrails Provenance Explorer plugin for ParaView is the work of many talented and devoted individuals. The work is derived from the VisTrails scientific workflow middleware system. Huy Vo wrote a first version of the ParaView plugin, which was later refined and extended by Geoff Draper and John Schreiner, with substantial contributions from Tilo Ochotta and Steve Callahan. Funding has been provided by grants from the Department of Energy, the National Science Foundation, the State of Utah, and the University of Utah.

Claudio Silva *is an Associate Professor of computer science and faculty member of the SCI Institute at the University of Utah, co-leader of the VisTrails project, and co-founder of VisTrails Inc. His research interests are in visualization and data analysis, geometry processing, and scientific data management.*

Juliana Freire *is an Associate Professor of computer science and faculty member of the SCI Institute at the University of Utah, co-leader of the VisTrails project, and co-founder of VisTrails Inc. Her research interests are in databases, web technologies, and scientific data management.*

John Schreiner *is a senior software engineer at VisTrails Inc. He is the technical lead on the VisTrails Provenance Explorer plugin for ParaView. He received a Ph.D. in computer science from the University of Utah for his thesis on geometric processing in 2008.*