

Interactivity: visualization's wayward child

Of all the elements of scientific or medical visualization, components related to interactivity have lagged consistently behind in maturity. I refer to "interactivity" in a broad sense: those elements that allow users to interactively manipulate and explore data and extract meaning from it.

Problems:

The following problems are just a subset of widespread limitations of current interactive visualization systems:

- lack of reference implementations for user interface techniques,
- lack of support for interface devices beyond keyboard and mouse,
- no good generic frameworks for concurrency for long-running visualization,
- no good generic frameworks to support exploration-friendly interface features such as undo,
- too close a coupling between user interface components and hardware platforms and graphics toolkits,
- extension of interactive visualization tools is exceedingly difficult.

These limitations encourage imperfect reinvention and discourage innovation.

Productive approaches:

These problems could be addressed through focused development of a platform-independent user interface core for interactive visualization. (VUITK). This toolkit would provide a user interface model to support interactive control of long-running processes, a concurrent processing model, a flexible event-handling mechanism, wrappers around device- or platform-dependent interfaces to input and output devices, an abstract interface to graphical output, reference implementations of interactive techniques, an extension mechanism for creating new interface elements, and finally hooks to enable testing. Development should include experts from the visualization, UI, and distributed computation communities.

Impact:

- Jumpstart development of new, high-quality interactive applications using established UI techniques.
- Encourage development of new UI tools and techniques.
- Encourage use and further development of alternative user interface devices.
- Provide a consistent framework to permit user interface testing (UI correctness) and user studies (UI effectiveness).

Volume visualization pipelines are unclean

The most established pipelines for visualizing volume data have characteristics that limit their usefulness for solving real-world problems. The legacy of polygonal isosurface methods (*aka*, “Marching Cubes”) is one of batch polygonalization and, by implication, batch segmentation. Typical direct rendering algorithms (“volume rendering”), on the other hand, mush together the distinct concepts of data and graphics primitives to the point where segmentation, hiding in a lookup table, is almost lost as a concept. These pipelines need an academic and engineering cleaning of legacy assumptions. The reward is more flexible, interactive applications.

Problems:

- Segmentation algorithms designed to work with Marching Cubes are almost exclusively batch algorithms, discouraging interactive exploration of data and segmentation space.
- Volume rendering algorithms can be incremental, but in general use the least flexible or powerful segmentation techniques. What's more, multi-

parameter segmentation (in medical applications, for instance) and the conversion to color and opacity space is usually done in what could be called an academically cavalier manner.

Productive approaches:

These problems can be addressed by a fresh look at the typical volume visualization pipeline based on a contemporary understanding of the theoretical and practical constraints of segmentation, rendering, distributed computing, and interactive user-interface design.

Impact:

- More interactive, accurate, and flexible volume visualization algorithms, independent of type.
- More effective exploration of not just data space, but segmentation parameter space.
- Better use of distributed computational resources when driven by better-defined interactive volume rendering tools.