Comments on "Hyperbola overlay program experience"

Rick Ottolini

I agree with Jon's overall thesis that the current generation of graphics workstations makes it easier to write interactive seismic graphics, but disagree with some of the details. The chief difference between the 'Movie' style program which was not written for a Sun and an 'Overlay' style program which was written for the Sun is that the Sun provides powerful support for pointing at a part of the data image and manipulating it. The old style interactive programs were limited more towards global display parameter changes because it was both painful to code and slow to use a pointing type of interaction.

The mystery of user interface design

Beside pointing at the data image, modern graphics workstations allow pointing and interaction with various command images- control buttons, menus, pop-up windows, etc. We are still learning how to achieve a good balance in a graphical command interface. One must avoid too few, too hard to find, and too hard to understand commands as well as not overwhelm the user with too many features. It took us a year to optimize the design of the keyboard-menu interface for controlling the Movie program.

Program organization

First attempts at interactive graphics programs (including early versions of of 'Movie' and 'Overlay') result in massive codes which are hard to extend or recycle. There are a large number of variable types and inter-related subroutines. People tend to group code segments together according to the utilities they use, e.g. all of the parameter fetches are written together. The result is that many widely separate parts of the code may have to be altered to add a new interactive command. Modifications are difficult for the program's author if not impossible for others.

The widely touted rules of programming discipline apply to interactive graphics programming more than ever. These rules include organizing code into groups about shared variables and operations (i.e. data abstraction or object-oriented style) and designing precise interfaces between various code modules. There are modern programming languages which make the implementation of these disciplines more convenient and self-organizing (modula-2, C++, smalltalk, ada). However, they are either not widespread or reliable. Thus we have to take special care when using our old standby—C.

The seismographical toolkit

Sun software provides graphical abstractions from primitives (e.g. copy a raster rectangle) to machine independent standards (e.g. GKS) to interactive objects (e.g. Sunview windows, menus). The *seismographical toolkit* continues these abstractions to the seismic domain. They include the databasing of seismic data, the graphical display of seismic data, and the numerical processing of seismic data. Because I have been concentrating on data cube dissection applications such as animation, slicing, nibbling, and transparency, my toolkit contains the abstractions of Data_cube, Seis_color, Velocity_function, and Cube_projection, etc.

These abstractions are implemented as reusable C-language modules based upon the programming disciplines discussed earlier. Not only do they include the routines applicable to the particular data abstraction, but provide dynamic data management as well. Since it is not completely known before program run time how many and what kind of data abstractions will be requested by the program user, there are routines for creating, initializing, and destroying instances of data abstractions. Modules also contain parameter definitions, variable templates (C typedefs), and connection subroutines for communicating between the various data abstractions. Precise coding guidelines are discussed in the document *Sun Programming Tips* which accompanies distribution of my source code.

REFERENCES

Claerbout, 1986, Hyperbola overlay program experience: this SEP report