Movies on the Macintosh II

Rick Ottolini

ABSTRACT

The Macintosh II computer was my first successful attempt to run seismic movies with a mouse-window interface. Previously, the demands of high speed graphics interfered with a responsive user interface. A mouse-window interface does not seem better than the old keystroke interface, except for adjusting the shape of the seismic image and interrogating the image.

INTRODUCTION

There are some data features that cannot be readily seen in data unless the data is animated. Animation is the general term for high speed data browsing which also includes magnifying, windowing, scrolling, rotating, warping, carving, overlays and transparencies of two and three-dimensional data images.

Interactive control of display parameters is essential to high speed data browsing. My first control interface was keystroke based. A single keystroke in most cases changed the display. It was not easy to learn, but very efficient to get some display state once you got the hang of it.

Affordable mouse-window color graphics became available after I had initially developed movies. Such a system promised a richer control interface than a keystroke method. However, early graphics workstations were hampered by inadequate color resolution (Macintosh I, Microvax II), slow graphics display (Microvax GPX, Apollo DN660) or interactive deadlock under high speed graphics requests (SunView, Sun NeWS). The Macintosh II was the first workstation to meet my demands.

Goals of a data browser

My goals, in descending order of importance are:

- high speed, i,e. animation;

- mouse-window control, i.e. cursor, menu, buttons, etc.;
- accomplish as much as possible by pointing at the image rather than through indirect controls;
- arbitrary number of entities such as datasets, views, colorings;
- three, or more, dimensional raster volumes;
- arbitrary magnification and windowing;
- good labeling;
- printer-independent hardcopy.

Claerbout's Balloon program (1988) emphasized labeling and printing because he was interested in preparing report illustrations.

After a description of Macintosh movie, I shall discuss some implementation details and future prospects.

TOUR OF MAC-II MOVIE

Sample display

Figure 1 shows an sample display. The four edge windows contain commands that control the data windows in the interior. A few additional commands are called by clicking parts of the data windows.

Along the top edge are the menu commands. These are listed in the appendix table. The commands are categorized as:

- input data files and saved images [`File`];
- documentation and display state information [`Info`];
- axial and cursor-selected views [`View`];
- animation loop control [`Loop`];
- image rotations, shear, shape and size [`Zoom, Rotate, Shear`];
- rendering as density or vector images, or audio [`Render, Audio`];
- color and contrast [`Color`];
- system requests [`Apple, File`].

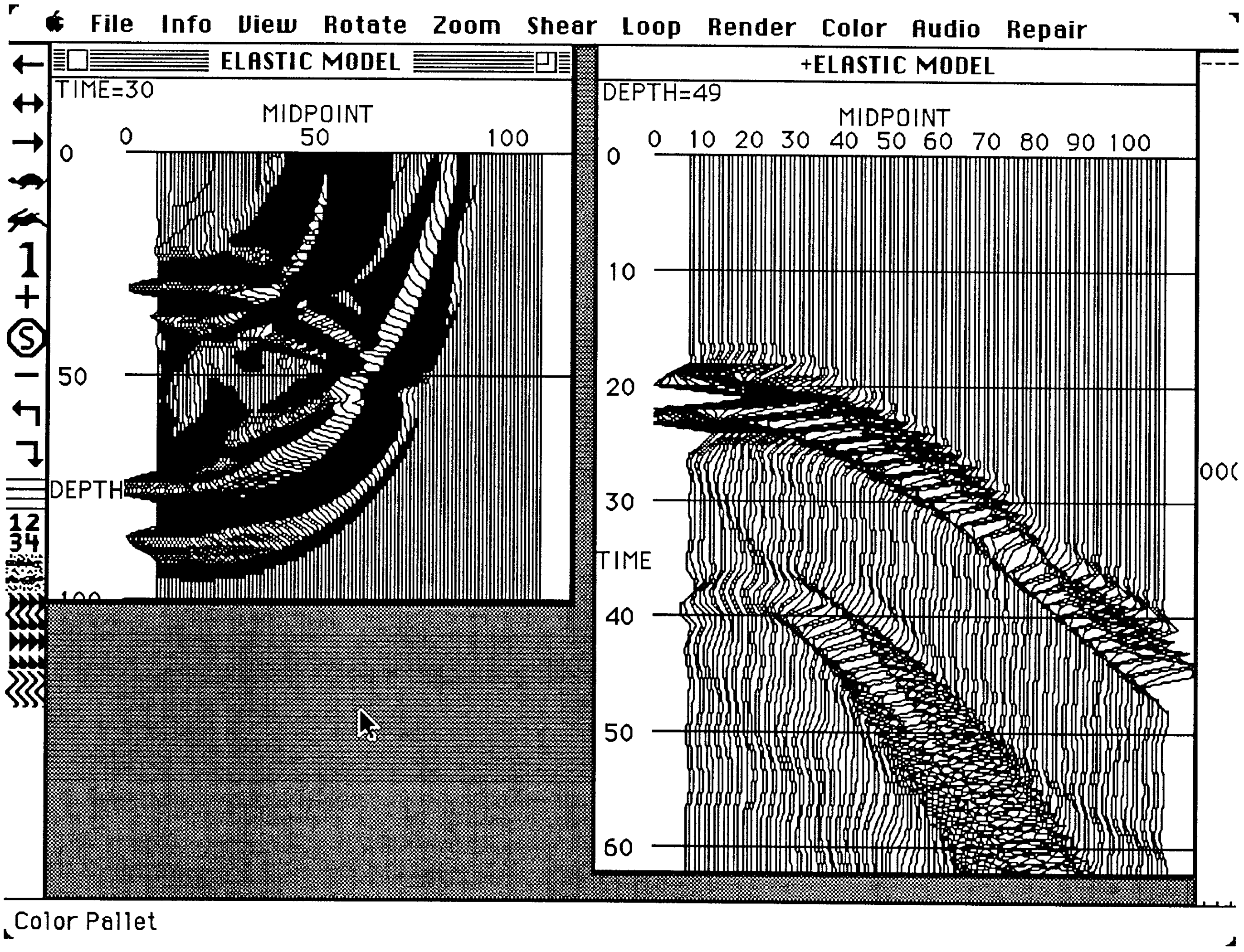


FIG. 1. Sample display of Macintosh II movie. Each of the four edge windows is a control mechanism—top, menus; left, icon commands; right, adjustable color pallet; bottom, message box.

Common commands are also accessible through icons along the left border or keystroke equivalents. These are also listed in the appendix table.

Popup control panels allow fine-tuning. They are accessible from the menu *adjust* commands, e.g. `Adjust View`. A control panel integrates all of the commands for a display object.

Commands operate on the currently *active* window, that with the darkened titlebar. Windows are activated by clicking the mouse somewhere within them.

Some commands change the behavior of the mouse without immediately changing the display. Feedback is given by the cursor shape. These are summarized in Table 1.

Table 1: Cursor Mode Changes

COMMAND	MODE	CURSOR
`DEFAULT`	print coordinates of data	arrow
`VIEW Horz Select`	select horizontal cross section	horizontal bar
`VIEW Vert Select`	select vertical cross section	vertical bar
`VIEW Box Select`	select rectangle of data	cross
`AUDIO Horz Select`	play horizontal trace	horizontal bar & speaker
`AUDIO Vert Select`	play vertical trace	vertical bar & speaker
`AUDIO Box Select`	play rectangle of data	cross & speaker

Along the right edge is a color pallet. Dragging the mouse-cursor in the pallet changes contrast.

Along the bottom edge is a message box. This gives redundant feedback, since almost every command causes an immediate change in the graphical display.

Running Mac II movie

Clicking the movie program icon draws the control edges and a standard Macintosh file selection box (Figure 2). Input data files are standard *seplib* format. That is, a binary image file accompanied by a text description file. The selected file is loaded and rendered as variable density. Then the data can be manipulated through the movie commands.

An arbitrary number of datasets and dataset views can be displayed until one runs out of screen space or core memory. New views grow out of the four screen corners in clockwise order.

IMPLEMENTATION NOTES

Software development environment

Macintosh II Movie was programmed in C under the *Macintosh Programmer's Workbench* (MPW) software development environment. The modular, object-oriented

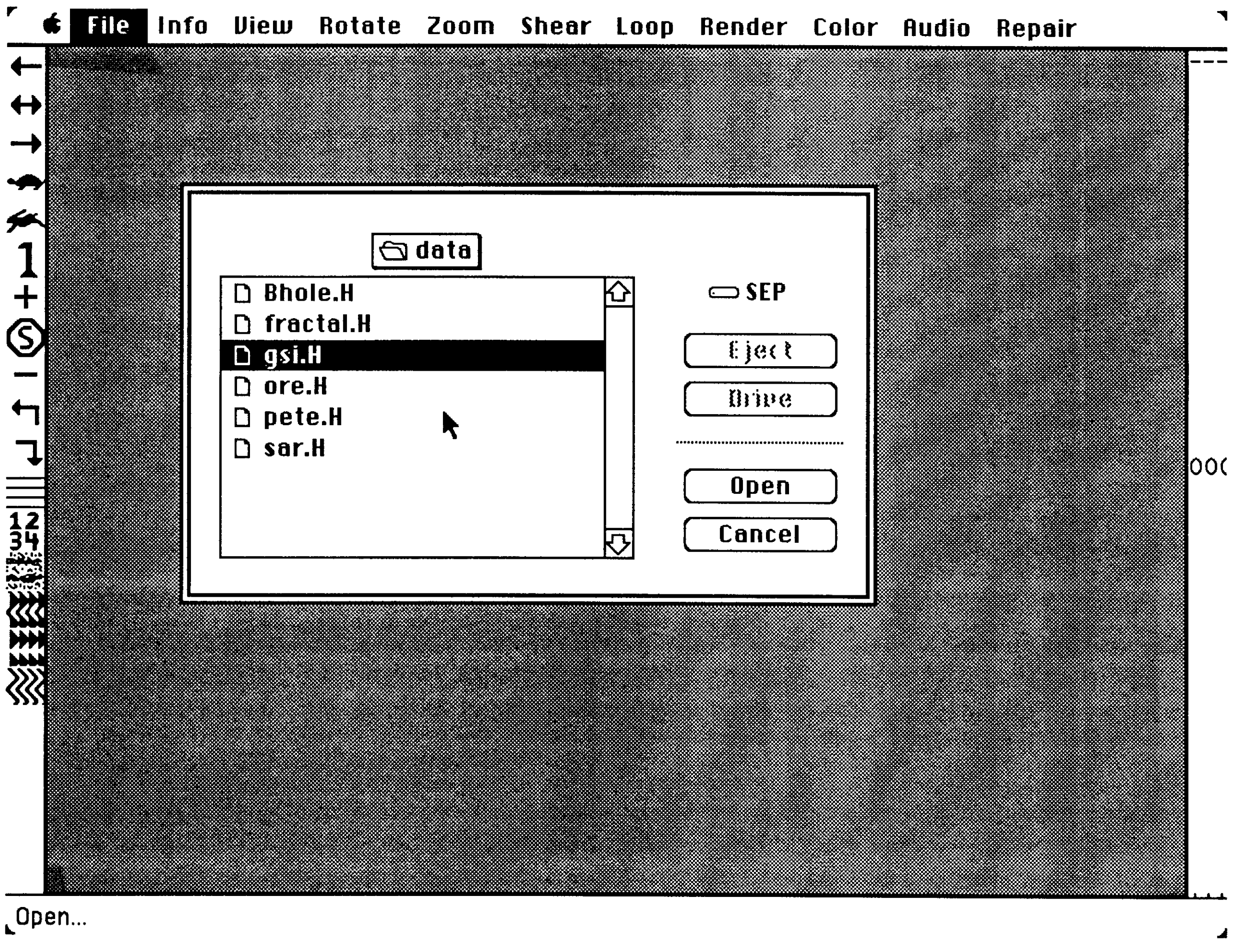


FIG. 2. Macintosh movie startup screen. File selection panel automatically comes up. *Seplib* parameter file names are displayed.

methodology of Ottolini (1987) was used. This methodology (1) allows multiple datasets, views, color tables, and (2) separates out machine-dependent code. Program objects are summarized in Table 2.

Table 2: Movie Objects	
Audio	audio rendering
Axis	data and screen axes
Color	color tables
Command	menu, keystroke, & icon commands
Data	*seplib* dataset
Loop	film loop control
Main	initialization and event loop
Message	message window
Palet	color pallet window
Render	graphics rendering
Rotate	view rotation
Select	view part selection
Shear	view shear
Simple	object primitives
View	data view
Zoom	view size
Macintosh Objects	
Control	icon buttons
Cursor	cursor shape
File	file names
Screen	screen

The disadvantages of this methodology are mainly logistic. Each object requires a description file, code file and optional Macintosh resource file. Lots of files are cumbersome and slow down program linking. These problems will go away as Macintosh C language tools improve.

Color

Macintosh II color monitors act like standard monitors with 256-color lookup tables. Operating system software also supports 1-bit, 4-bit and 24-bit (soon) color monitors, although my software doesn't. I compress the seismic data color-continuum from 255 colors to 64 colors as data is read from disk because the operating system does not make all 256 hardware colors available.

Apple monitor crispness is the best I've seen in color. It is a pleasure to work with such a display.

Speed

The image display speed is as fast as any other 68020 CPU workstation on the market (Ottolini, 1988). Apple recommends using a system routine [`Copybits`] rather than write directly into video display memory. Though the system routine solves several problems—variable color planes, screen clipping, multiple monitors, and zoom—it is a order of magnitude slower.

The speed of the Macintosh in drawing user-interface objects—windows, menus, icons—is the fastest I've seen. It is also a pleasure to work with.

Zoom

Image size control is a crucial part of browsing software from both an user interface and implementation point of view.

The old graphics terminals had hardware zoom. This was important in filling the screen when image transmission from the CPU was slow. However, hardware zoom made text labels difficult, if not impossible, and resulted in cryptic displays.

My first attempts at software zoom imitated the hardware zoom pixel replication algorithm. This allows integral zooms and sub-samplings. Various programming tricks, such a unrolling-short-loops and byte-to-word table lookups, speed up zoom. This took about fifty special purpose subroutines including combinations of positive, negative, horizontal and vertical zooms.

Claerbout (1988) suggested a simpler method for arbitrary zoom. The above method could be considered *push-zoom* and is analogous to Kirchoff migration by smearing. Claerbout's method is *pull-zoom* and is like diffraction summation. The algorithm is to pre-compute a zoom interpolation vector, then convolve it with a succession of data images. This algorithm works the same for positive and negative zooms. One zoom vector is for the horizontal display axis and another for the vertical axis. By applying a scale and bias to the interpolation vector, zooming can be combined with clipping, cross sections, rotation and shearing.

The algorithm for computing the zoom interpolation vector is:

```
for each output point i {
    index = (i * zoom_length * data_element_separation) / (data_length);
    }
```

The convolution algorithm is:

```
data_base = first_column_element * column_element_separation +
    first_row_element * row_element_separation;
for each output row j {
        row_base = data_base + row_index[j];
        for each output column i {
            image[i,j] = row_base[column_index[j]];
            }
        }
```

Using pointers for arrays, the inner loop requires three memory accesses and one addition per output image point. The speed is linear with the size of the output image.

3-D zoom

The previous zoom algorithm solves the *mechanics* of three-dimensional cube displays with arbitrary orientation and size, but not how to specify such. Earlier versions of the movie program had twenty predefined cross section and cube orientations. Windowing and zooming were done on the single panel cross sections and "remembered" in the cube views.

Though not fully implemented at the time of this article, I envision the following cubical windowing and zoom (Figure 3):

- The axes, shape, and bounds of the two-dimensional cross section forming the front face of the cube is obtained with current commands.
- Cube side and top panels of arbitrary size are "grown" by grabbing one of the corners of the front face (Figure 3a).
- The cube is reshaped by grabbing one of the square-angle corners (Figure 3b).
- The cross hairs, showing where within the volume each face comes from, are adjustable (Figure 3c).
- Selection rectangles do windowing (Figure 3d).

One problem of three-dimensional specification is combinatorial—there are many more orientation and shaping possibilities than for two-dimensions. Another problem is the ambiguity of specifying a three-dimensional operation on a two-dimensional surface (Chen, et al, 1988). Often there are two or more possible actions for a certain mouse behavior. In other words: there is a $+z$ and $-z$ solution to the behavior equations for a given (x, y) input.

Memory

The Macintosh operating system does not support virtual memory. This limits movie dataset sizes to core memory size minus operating system and program overhead. This burdens the programmer with memory management.

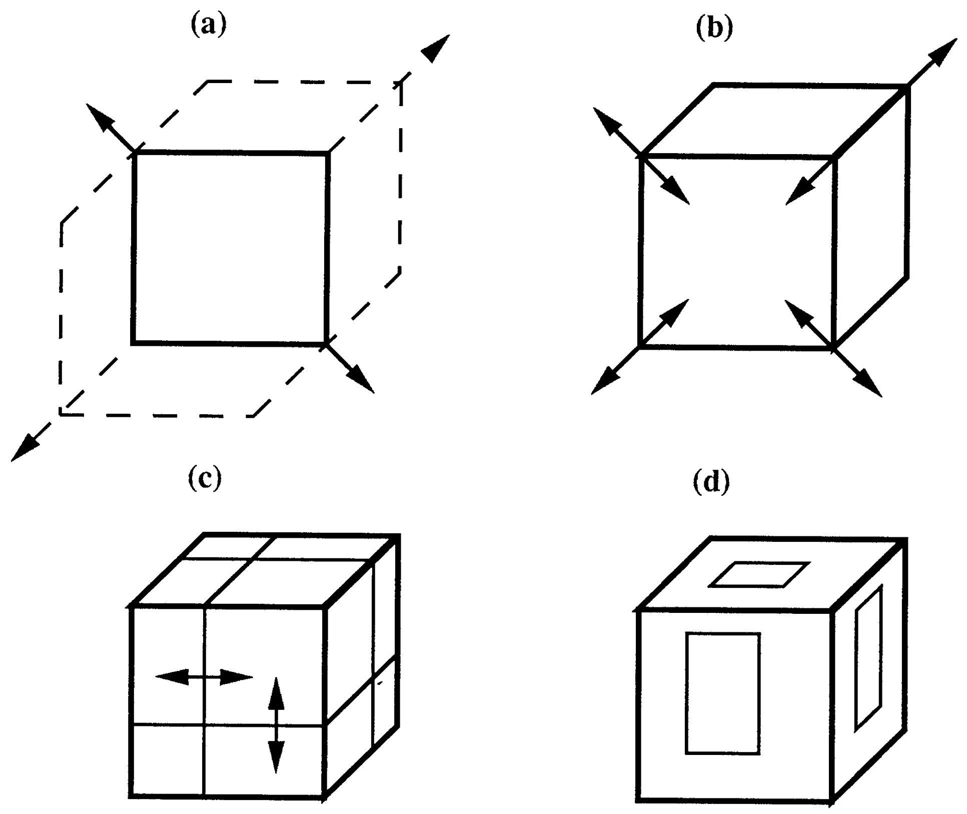


FIG. 3. Scheme for specifying size and orientation of three-dimensional image: (a) shape, (b) size, (c) frame intersections, (d) windowing.

Another problem is program size restrictions. A segment of compiled code must be less than 32KB (16-bit address limit). Also, the number of linkages between segments is limited to four thousand.

DISCUSSION

Merits of the mouse-window interface

It took some time to perfect the keystroke movie interface. One wishes to get rapidly to any possible display state without too much work. There are very few multiple keystroke commands (until we started running of keys). Too many choices is as bad as too few. We dropped the joystick controlled commands, much of the color flexibility, and most of the possible cube views for that reason.

We are on a similar learning curve for mouse-window interfaces. I generally don't like menu commands because they less accessible than icons or keystrokes. However, there is not enough icon display space or keys to replace all menu commands. People find keystrokes harder to learn than menus and icons. But keystrokes are more efficient when giving a lecture and controlling the screen simultaneously. The mouse is nice for adjusting window shape and selecting parts of the window contents. However, the mouse is tedious to move it across a large screen to reach the menus and icons.

Special features of the Macintosh

The graphics is fast. The monitor is crisp. The menubar is tedious on a large screen. Better automatic memory management is needed for large programs and datasets. The graphics software is a mixed bag. It has all the functions, not an uniform syntax. It is proprietary, so learning to program the Macintosh is like reinventing-the-wheel. A source-code level debugger is needed to track down bugs. It is hard to get data in and out of the Mac. Ethernet hardware exists, though file transfer software is primitive. The better integrated AppleTalk is much slower. Floppy disks are too small to move much data.

Future work

Future work involves solving three-dimensional user interface issues. Lessons learned on the Macintosh should be integrated into Sun and Raster Tech movie programs. We should decide whether to move our other interactive graphics software to the Macintosh.

REFERENCES

Chen, M., Mountford, S.J., and Sellen, A., 1988, A study in interactive 3-D rotation using 2-D control devices: Computer Graphics, **22**, 4, 121-130.

Claerbout, J.F., 1988, The balloon program: SEP-**57**, 549.

Ottolini, R., 1987, Techniques for organizing interactive graphics programs: SEP-**51**, 421.

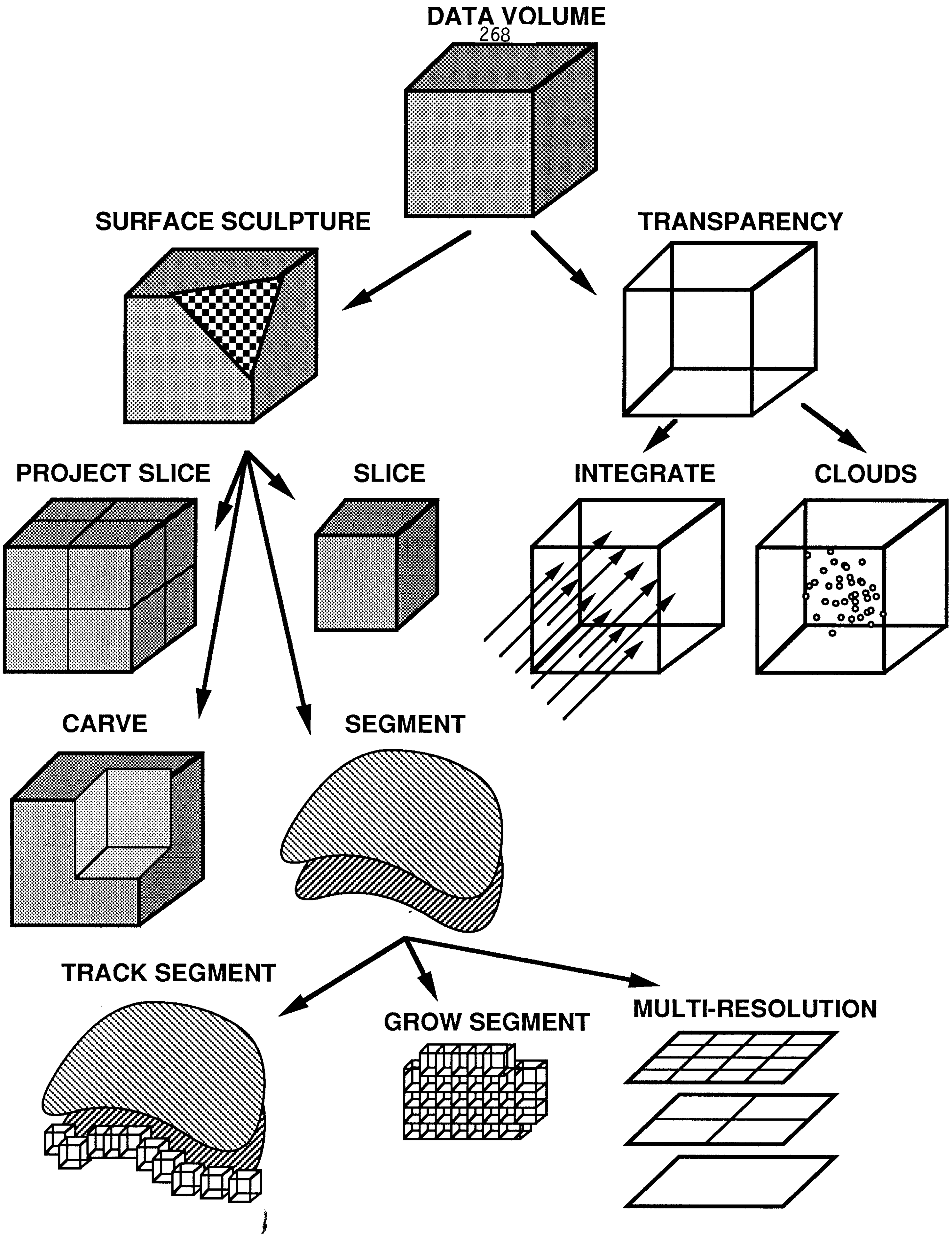
Ottolini, R., 1988, SEP workstation update: SEP-**57**, 579.

APPENDIX: Movie Commands

Movie Commands		
NAME	KEY	ICON
File		
Open ...	o	
Open As ...		
Close	w	
Save As SEPLIB		
Save As VPLOT		
Quit	q	
Info		
This Data		
This Loop		
This Window		
Horz Axis		horizontal line
Vert Axis		vertical line
Loop Axis		box
This Color		
About Movie		
Window Icons		
Mouse in Pallet		
View		
Front	1	
Side	2	
Top	3	
Horz Select ...		
Vert Select ...		
Box Select ...		
View Adjust ...		

Movie Commands, ***cont.***		
NAME	KEY	ICON
Rotate		
Original	u	
Left	h	bent left arrow
Right	l	bent right arrow
Horz	h	
Vert	v	
Zoom		
Original	z	window zoom box
Maximum	Z	window zoom box
Square	=	
Horz Shrink	x	
Horz Expand	X	
Vert Shrink	y	
Vert Expand	Y	
Zoom Adjust		window grow box
Shear		
None		
Left		
Right		
Up		
Down		
Loop		
Forward	f	right arrow
Reverse	r	left arrow
BothWays	b	double arrow
Stop	0	stop sign
First	1	1
Step Forward	n	+
Step Backwards	m	-
Slower	s	tortoise
Faster	S	rabbit
Delay 0.0 sec		
Delay 0.1 sec		
Delay 0.2 sec		
Delay 0.5 sec		
Delay 1.0 sec		
Delay 2.0 sec		
Loop Adjust ...		

Movie Commands, *cont.*		
NAME	KEY	ICON
Render		
Density	D	density icon
Variable Area	V	variable area
Positive Area	P	positive area
Wiggle	W	wiggle lines
TimeLine	T	three horizontal lines
Labels	L	one two three four
Color		
Gray	I	
Clip	C	
Half	H	
Flag	F	
Blue	B	
Multi	M	
File	U	
Adjust Contrast		color pallet
Audio		
Horz Select ...		horizontal line
Vert Select ...		vertical line
Box Select ...		box
Audio Adjust ...		
Repair		
Active View		
Icon Bar		
Color Pallet		
All Windows		


DATA VOLUME
SURFACE SCULPTURE
TRANSPARENCY
PROJECT SLICE
SLICE
INTEGRATE
CLOUDS
CARVE
SEGMENT
TRACK SEGMENT
GROW SEGMENT
MULTI-RESOLUTION