Reprinted from

Proceedings
Graphics Interface '82

Réimprimé des

Comptes rendus
Interface graphique '82

AN OBJECT EDITOR FOR A REAL TIME ANIMATION PROCESSOR

S.P. Ressler

17-21 May/mai 1982
Toronto, Ontario

An Object Editor for a Real Time Animation Processor

S.P. Ressler

Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

The object editor allows one to use the real time capabilities of an animation processor to create and/or modify graphic objects. These objects can subsequently be used in other application programs. The editor enables the user to change the definition of objects, which are being processed and displayed by the animation processor, in an interactive manner. One is presented with a "rubber solid", which is completely manipulable. The user is also given the option of displaying duplicate objects, thus enabling two independent views of the current object. Objects, or pieces thereof, which appear to be quite solid can be stretched and/or squashed in real time.

The animation processor does hidden surface calculations in real time while the object editor modifies the description of the object, resulting in a dynamically changing image. This editor is not meant to be a production line system for interactive object design, rather it is a research tool for exploring the possibilities of interactive object manipulation.

1. Introduction

The function of the object editor is to allow a user to manipulate the description of an object while it is being displayed. Display of the object is performed by the animation processor *(AP)*[1]. The term *object* as used in this paper is a collection of x,y,z coordinates which define a group of polygons that can be manipulated and referred to as a single entity. An object in space has a single location and orientation (attitude), which are part of its data structure. The object editor as it exists is not meant to be a production line design tool, rather it is a research tool for exploring the possibilities of real-time interaction with computer generated objects.

2. Problems & Questions

In designing the interface through which a user is to interact with the system some questions arose as to the nature of that interaction, specifically as they relate to the usage of a real time animation system. This particular system provided an opportunity to explore and define some graphic tools which are appropriate for a real time capability.

The AP system software enables one to assign and define the functions of the joysticks, sliders, and buttons in a simple and straightforward manner. The existence of firmware to allow one to rotate and position the objects under the control of these input devices was also part of the system software as well an numerous Macros in the language which facilitated manipulation of the data description of the objects.[2] The type of questions therefore asked in the design of the object editor were ones concerned with the high level user interaction with the system. What kinds of physical to functional organization of the input devices are most usable? How can one remember which devices serve which functions? What kinds of visual to functional mappings can be done with a real time system? Assuming that real time systems will become more prevalent, what kind of useful graphical tools can be defined which are only possible and practical on a real time system?

3. User Interaction

The user interacts with the system by using the joysticks, buttons and sliders present on the *AP (see figs 1 & 2)*. Many of the operations used, have been multiplexed onto the buttons located on the base of the joysticks. One button has a select mode function which changes the functions of the other buttons. Feedback for the current mode is seen immediately on the graphics screen via blinking status indicators, and on the terminal which presents auxiliary text information *(see fig 3)*. The two circular images on the terminal correspond to the two joysticks, as viewed from the top. The surrounding words are labels which correspond to the buttons located on the joysticks. The words are dynamic labels and as the functions of the buttons change so do the labels on the terminal.

The decision to have most of the functions on the buttons was made because they are very discrete controls, which make them easy to use. The sliders which can also be used, provide a means of quickly doing something, (i.e. positioning the object) but they can get clumsy for accurate activities. One can effectively use the buttons, to single step actions if desired. The speed at which the buttons affect things is controlled by a slider. The use of a "speed" slider combined with the buttons

Figure 1: Joystick with buttons

make these controls very usable, for most of the system's functions.

The user is given the choice of several modes which follow a logical order, smaller operations to larger. Selecting one or another mode affects the functions of the other buttons in a very straightforward manner. The functions of the x,y,z buttons change depending on the context of the mode but always serve to affect the x,y,z dimensions independently. These particular buttons cause an action to occur as long as they are pressed and stop the action when released. For example in point mode the x,y,z buttons serve to move a single point (the current one). In polygon mode the same buttons will move the current polygon, which can be identified by hitting the show poly button. In volume scale mode the buttons act as scaling controls and change the scale of the volume box in the three dimensions. By keeping the idea of *x,y,z symbols* separate from the specific function (movement or scaling), the user need only remember which buttons are for x,y and z and will determine the specific function by observing the current mode, from the status indicators.

The joysticks are used to control the orientation of the objects. One joystick controls the primary object and the other, if desired, the duplicate. Since we are dealing with a real time environment the need to precisely specify the orientation of the object, is eliminated. One simply moves the joystick that is controlling the orientation of the object, to the desired view. The object can also be made to rotate continuously while other operations are performed. Throughout the entire system an effort has been made to provide the user with as much visual feedback as possible. The system as a whole has many variables which the user has to deal with and this feedback is not a luxury but a requirement.

4. System Capabilities

Most of the various capabilities of the system are presented to the user in the form of several modes. These modes are selected via the mode select buttons, which cycle

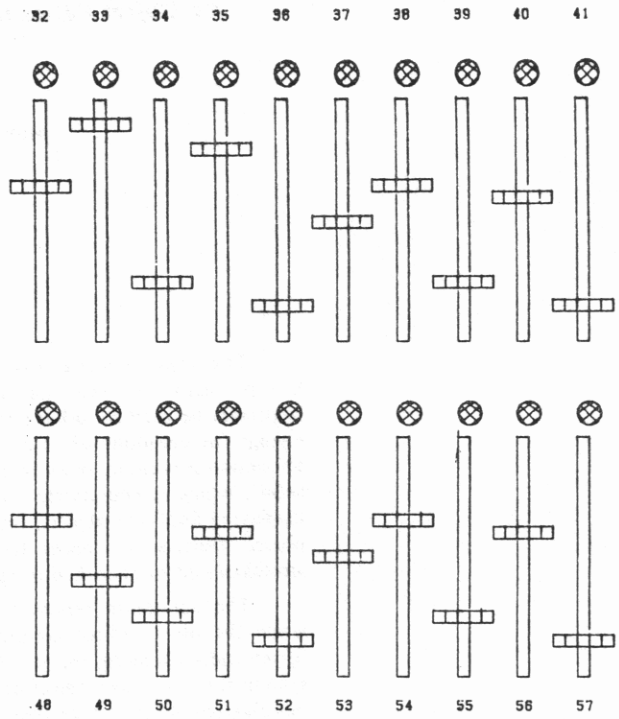


Figure 2: Slider Board

one through the modes either forwards or backwards. The possible modes are: *point, polygon, volume and addition.* The mechanism used to select or step through the data description is via the *stepping button.* This button will step you along point by point when in point mode, by polygons when in polygon mode. It makes no sense to step along volumes so the stepping button does nothing when in volume mode. If a capability makes no sense, then the button will do nothing. Again the symbolic idea of stepping is separate from the exact function of what we are stepping through. If the system is in polygon mode, the*copy button* copies polygons. If it is in point mode, the *delete button* deletes a point.

Feedback for the various modes comes in various forms. When in point mode a blinking dot visually appears at the point where one is currently located in the data description. The current polygon can be identified by pushing a button which causes it to blink. Addition mode is a little different then the others. It adds pieces to the object, rather then modifying existing pieces. This is different from copying because completely new points are added. When in addition mode four of the joystick buttons get redefined to enable the addition of a dot, vector, rectangle, or cube.

One interesting aspect of the system is the graphical interaction while the system is in volume mode. A box is present on the graphics screen which is used as a device to select areas or volumes of an object which are not necessarily contiguous in the data structure but which are visually related. The desired portion of an object to be moved or scaled can be selected by moving the volume selection box. All points which appear inside the volume box will become activated. After the desired points have been selected they can be moved or scaled.

The scaling is done so the points move away from the center of the box. This particular graphical tool is most appropriate for a real time system because the movement of the volume selection box around the desired points would be a clumsy task if one could not move the box and rotate the object quickly. It is also a quite intuitive and visual way to manipulate objects and functions in a manner analogous to the classic *rubber band line*, except in this case one can think of the area being manipulated as a *rubber solid*. The user is given the capability of playing with an object by stretching and squashing all of its parts until the object looks visually appealing. This technique should be quite useful for any real time system that needs a visual way of selecting and manipulating areas of points.

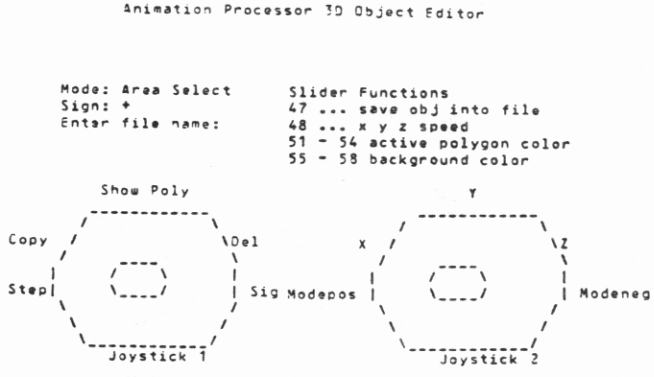


Figure 3: Information on Terminal

5. Implementation Environment

The object editor exists as one part of a complete animation system currently under development. The *AP* is run together with an LSI 11/23 and share a common memory and bus. The system uses a Micropolis winchester disk for storage and runs stand alone. The *AP* software is primarily an animation language which is written in C [3] under UNIX† V7, using YACC and Lex. This animation processor language written by Carl Christensen, was used to write the object editor. The editor also functioned as a a large scale test bed for various aspects of the language.

6. Getting Hard Copy

Another facility in the object editor is the ability to take a "snapshot" for creating Unix type plot files. The images created by this method are simple one color line drawings. Moving one of the sliders causes the system to ask for a file name. The display list of the animation processor is then placed into that file. To create a plot file say *toplot file* and a file will be created with the name you gave appended with a .f . This is a Unix device independent plot file and can be used to produce plots on any of the devices supported by plot.

†Unix is a Trademark of Bell Laboratories.

7. Conclusion

A system for the graphics manipulation of objects has been demonstrated. The use of a real time system for design presents one with tremendous possibilities for a more creative form of design. A user need not be as concerned with exactly the numbers to type in to get some graphic form to look just right. You simply twist and turn some devices until it looks good. There is however a great need for defining how the user should interact with such a system in ways which prove to be useful. One such tool, a volume manipulation device, has been demonstrated and should be applicable to any real time, graphics system. The usefulness of this device as well as the system is based solely on the author's opinion and has yet to be tested in a productions sort of environment. Many more such tools are needed until a system which is friendly, and powerful will exist.

8. Acknowledgements

Thanks to Carl Christensen who wrote the animation processor language and was always willing to answer sticky questions and add needed features. Much thanks also to David Baraff who wrote the hidden line routines which were used in the Appendix.

References

[1] H.G. Alles, W.C. Fischer, *An Animation Processor for Action Oriented Three Dimensional Color Graphics* to be published.

[2] C. Christensen *An Animation Processor Language* to be published

[3] Kernighan and Ritchie, *The C Programming Language*, Prentice Hall, 1978.

Mode: Volume Select

Feedback Volume cube blinks blue.

Description: Moves the volume cube around to enable the selection of any point inside the cube, for further manipulation, in volume pts move or volume pts scale modes.

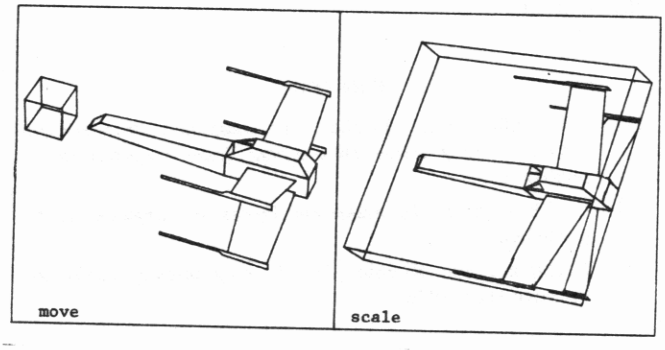
Usage: The x,y,z buttons on the joystick move the volume selection cube in the x,y,z directions. The speed slider is active, and sign affects movement direction.

Mode: Volume Scale

Feedback Volume cube blinks green.

Description: Scales the volume cube so more flexible volume can be selected.

Usage: The x,y,z buttons on the joystick scale the volume selection cube in the x,y,z directions. The speed slider is active, and sign affects movement direction.

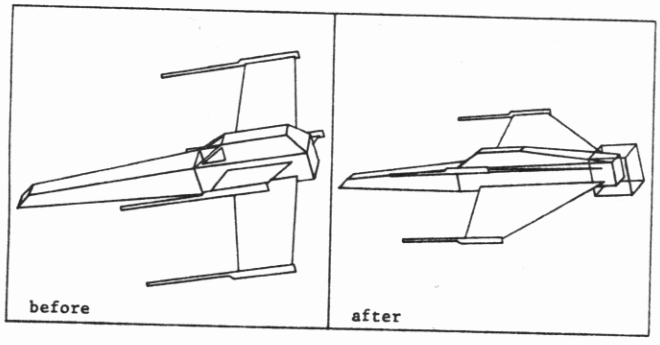


Mode: Volume Points Movement

Feedback Open square in status indicators blinks.

Description: Points inside the volume cube are activated for movement.

Usage: The x,y,z buttons on the joystick move all points inside the volume cube in the x,y,z directions. The speed slider is active, and sign affects movement direction. All points will move regardless of whether or not only part of a polygon is in the volume cube.

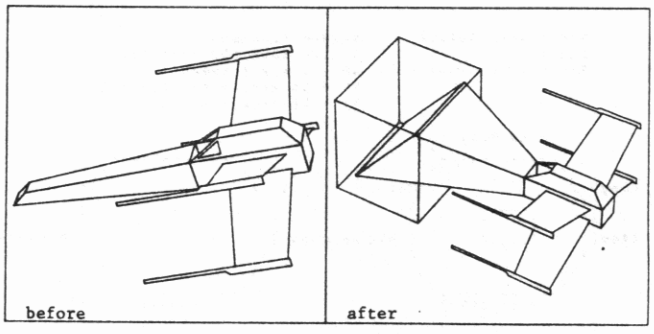


Mode: Volume Points Scale

Feedback Line above open square in status indicators blinks.

Description: Points inside the volume cube are activated for scaling.

Usage: The x,y,z buttons on the joystick scale all points inside the volume cube in the x,y,z directions. The scaling is always towards or away from the center of the volume cube. The speed slider is active, and sign affects the direction of the scaling. All points will scale regardless of whether or not only part of a polygon is in the volume cube.

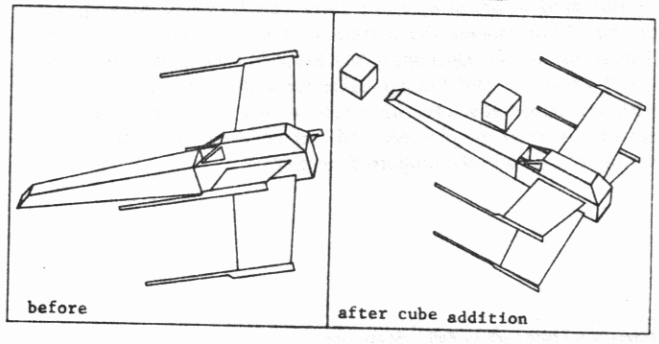


Mode: Addition

Feedback Dot, vector, rectangle and cube in status indicators blinks. Terminal display the dot, vector, rectangle and cube button.

Description: Allows the addition of a dot, vector, rectangle or cube to the object description.

Usage: The buttons on joystick 1 get redefined when this mode is entered, to be dot, vector, rectangle or cube addition buttons. Press one of these buttons once and the corresponding shape will appear at the center of the object (location 0,0,0). Enter one of the movement modes to subsequently move this new shape to its desired location.



Appendix: Mode Descriptions

Mode: Point

Feedback Blinking dot in the status indicators. Blinking dot at the current point.

Description: Enables the manipulation of any *single* point in the object being edited.

Usage: The x,y,z buttons on the joystick move the current point, in the x,y,z directions. The speed slider affects the speed of x,y,z movement. The sign button is a toggle between plus and minus. Point movement is in the sign direction for each axis. The sign is displayed in the status indicators and on the terminal. The copy and delete buttons will copy and delete a single point. The point is selected via the step button which will step forwards if the sign is plus and backwards if the sign is minus. Point stepping will "wrap around" through the entire object description.

Mode: Polygon

Feedback Blinking solid square in status indicators. Show poly button blinks current polygon.

Description: Enables the manipulation of any polygon in the object being edited.

Usage: The x,y,z buttons on the joystick move the current polygon, in the x,y,z directions. The speed slider is active, and sign affects movement direction. The copy and delete buttons will copy and delete a single polygon. The current polygon is selected via the step button which will step through the object description a polygon at a time in the sign direction. The show poly button is a toggle to blink and unblink the current polygon. (Note: Make sure to unblink the polygon before moving on to another polygon, as color information can get lost)