SES CUPOLA INTERACTIVE DISPLAY DESIGN ENVIRONMENT

Bang Q. Vu & Kevin R. Kirkhoff

Lockheed Engineering & Sciences Company

INTRODUCTION

The Systems Engineering Simulator, located at the Lyndon B. Johnson Space Center in Houston, Texas, is tasked with providing a real-time simulator for developing displays and controls targeted for the Space Station Freedom. These displays and controls will exist inside an enclosed workstation located on the space station. The simulation is currently providing the engineering analysis environment for NASA and contractor personnel to design, prototype, and test alternatives for graphical presentation of data to an astronaut while he performs specified tasks. A highly desirable aspect of this environment is to have the capability to rapidly develop and bring on-line a number of different displays for use in determining the best utilization of graphics techniques in achieving maximum efficiency of the test subject fulfilling his task.

The Systems Engineering Simulator now has available a tool which assists in the rapid development of displays for these graphic workstations. The Display Builder was developed in-house to provide an environment which allows easy construction and modification of displays within minutes of receiving requirements for specific tests.

SOFTWARE DESIGN

Program Structure

The Display Builder is compiled to run under UNIX AT&T System V on a Silicon Graphics' IRIS 4D/70 GT graphics workstation. It has fourteen modules, and nearly 11,500 lines of C source codes. Four modules are dedicated entirely to 2-Dimensional (2D) graphics; four are dedicated to 3-Dimensional (3D) graphics, and the rest to user interface and display list management. The executable size is roughly 400K bytes.

Data Structure

The display list is implemented as a doubly linked list. Each node in the list contains various house-keeping data as well as an union (set) of all structures (records) representing 2D and 3D primitives. The advantage of using the union feature of C is that although the primitives have variable length, they all fit into a node, thus greatly simplifying the task of data management.

USER INTERFACE DESIGN

Any software, especially an interactive graphics application system, is often judged primarily on its success to deliver its functionalities to the users. Even if the system is computation-intensive, what good is it if it fails to communicate effectively with the human operator? In the worst case the acceptability of the whole program may be invalidated because even the experienced users shy away from a poorly-designed User Interface (UI). The Display Builder employs a direct-manipulation UI popular on many modern interactive graphics systems.

UI Design Strategy

The Display Builder's UI lets the user manipulate objects directly on the screen. This type of interface is popular because it is easy to learn, and easy to use; however, it is also one of the most difficult to implement. Direct-manipulation UI is often complex due to stringent performance requirements (rapid actions and feedbacks), elaborate graphics, asynchronous input devices, and various ways to give the same command (keyboard and mouse). Currently the most successful strategy to create a reliable UI is the Iterative Design method. Under this method, prototypes of the Display Builder were iteratively tested and modified based on users' comments to generate the final UI.

Command Language

Command languages appear in all computer systems. A command language is the set of actions a user is allowed to have and the methods through which he can request a particular action. In designing the Display Builder's command language, the following issues were resolved:

Command Style: The Builder is both keyboard-dialogue and menu driven. Dialogues are of a very simple nature and conducted inside a dialogue box. For example, the Builder may prompt for a string or a number, in which case the user would proceed to type in the requested data. It may ask for an answer, such as "yes" or "no", in which case the choices are presented and can be selected. Most of the time, the Builder is menu driven. Selecting either an answer from the dialogue box or a choice from a menu can be done by the mouse or keyboard shortcut (see FIG 1,2,3).

Command Modes: The builder has a dual mode, 2D and 3D, command language, and therefore interprets user actions in two different ways. For instance, if the Builder is in 2D mode and the user requests to move an object, he can move only 2D objects. The user switches from one mode to the other via the main menu (see FIG 2). The dual mode method is chosen because it improves the UI by cutting down the size of many menus, and contributes heavily to program maintainability by keeping modules handling 2D and 3D graphics separate.

Command Abort & Error Handling: These are usually the most critical areas of any interactive system because of the presence of an unpredictable human in the process. The Builder allows the user to abort any single- or multi-step command by pressing the ESC key. In case of user error, such as opening a nonexistent file, a beep would

sound, the file name together with an error message appears in the dialog box and the user can then correct the error or abort the command. Because the Quit action irretrievably destroys the drawing, it is implemented as a two-stage command. If the user chooses this action, the following warnings occur: a beep would sound, and a prompt appears in the dialog box. Then the user can abort the command or confirm it by clicking the mouse's right button.

Information Displays

This is one of the most subjective and therefore troublesome areas of UI design: how to display information in the most "effective" manner. Screen layout and object display are the two items of interest.

Screen Layout: The prompts for the universal command abort key (ESC), current file name, current drawing color, and current builder mode (2D/3D) are displayed in the upper left corner. The dialog box is located in the upper right corner (see FIG 1). The drawing area can be created and moved to any place on the screen. Menus appear on the right side of the screen only when needed.

Object Display: The Display Builder makes no distinction between 2D and 3D objects; they coexist in the same drawing space. The user can allocate any area of the screen, either same or separate, to 2D and 3D objects. All objects will automatically be clipped to fit inside its allocated space.

Interactive Graphical Input Techniques

The user interacts with the computer through a graphical display. He is generally not skilled in graphics, and only interested in how fast and easy it is to accomplish his task. Interactive techniques reduce the need for great manual dexterity and the effort required to draw and manipulate objects visible on the screen. Feedback, Selecting, and Positioning are the techniques of interest.

Feedback: This is an essential component of graphical interaction. Feedback techniques help to provide immediate confirmation to the extent or intention of the user's action. For instance, if the user selects multiple objects to delete, and no feedback is provided, the user is left to wonder whether he has made the right selections. The uncertainty will eventually be answered when he gives the delete command, but the efficiency of interaction is severely hampered. The Builder uses highlighting, bounding box, blinking color, prompts, and beeping noise to provide feedbacks to the user.

Selecting: the need to select primitive(s) to manipulate is one of the most basic interactive graphics techniques. The Builder supports different selection techniques depending on the mode it is in.

2D selecting: The user can select one or many 2D primitives unambiguously by pointing to and clicking on specific spots located on these primitives. These selection points appear only when required in logical places such as the center of a circle or the bottom left corner of a button (see FIG 5). Feedback is provided by immediate highlighting of selected selection points.

3D selecting: The user can select one or many 3D primitives by clicking on their names in a linear display list which appears when required. Feedback is provided by immediate blinking of selected primitive(s).

Positioning: The Builder supports the following techniques to position a new 2D primitive or relocate an existing one: grid, rubber banding, dragging, and aligning.

Grid: A grid is provided. The user has the option of choosing a background or foreground grid or no grid at all.

Rubber Banding: Most 2D primitives are created with rubber bands. For example, the user defines the first point of a line. As he moves the mouse, the Builder draws a line from the first point to the current cursor position. When the second point is defined by the second click, the rubber band disappears, and the permanent line is drawn. This technique takes the guess work out of positioning a new primitive as the user can see instantly how large and where it is on the screen.

Dragging: This technique is used with moving or copying 2D primitives. For example, the user starts by selecting the primitive(s) that he wants to relocate. Then he can "drag" their outlines to the new location. The selected primitives are permanently moved when the user defines the final location by clicking the mouse.

Aligning: There are four ways to align 2D primitives: left, right, top, and bottom. All selected primitives are automatically aligned.

3D graphics does not lend itself well to the above techniques. Currently, the only way to position a new 3D primitive or reposition an existing one is to enter new data via the dialog box. After the object has been completely defined, it will be redrawn at the new location.

FUNCTIONAL CAPABILITIES

Anything drawn inside a display is done through a primitive, for example, a line, a number or a button. A primitive is a packet of data used to construct a numerical or graphical representation of information in a display. This data is stored in the display's data file and used by the real-time software to draw each primitive. The following terms will be used to describe some features of the primitives:

SYSID: An index into a block of shared memory that interfaces between the display and the simulation.

THRESHOLD: Used to denote a caution state (yellow), and critical state (red). Thresholds are optional within each primitive.

TRANSITION STATE: In the switch primitive, the time between the user requesting a switch be activated, and getting an indication from the math model that the switch has been activated.

2D Actions & Primitives

The user can create 2D primitives in a display with minimal effort. For instance, he builds a circle by pointing to a spot on the screen where it will reside, defining the center with the first click of the mouse, rubberbanding the circle to the desired size, terminating with another mouse click. Data for that primitive may be edited at that time, or left until the entire display is created.

The following actions can be performed on 2D primitives:

Edit	change any or all data of a selected primitive. (See FIG. 5).
Delete	remove selected primitive(s).
Move	move selected primitive(s).
Copy	duplicate selected primitive(s).
Align	top, bottom, left or right justify selected primitive(s).
Save	save selected primitive(s) to a disk file.
Read	read primitive(s) from a disk file and insert them into the display list.

2D Primitive List (see FIG 3):

Header - Contains display size, font and the length, in bytes, of the display file.
Integer - Contains thresholding.
Single Precision Real - Contains thresholding.
Double Precision Real - Contains thresholding.
Hexidecimal - Contains thresholding.
Ascii message - Shows eight characters of variable text.
Static text - Shows eight characters of fixed ascii text.
Button or Switch - May be a toggle or momentary switch. These primitives are similar in function and makeup except that a momentary switch is activated when the mouse button is pressed, and de-activated when released.
Keyboard input - Allows the user to enter data into the simulation from the keyboard.
Page call - Activates new displays.
Default - Allows a user to customize a screen layout .
Indicator - Reflects the state of its related sysid.
Circular gauge - These come in two types, increasing and decreasing, and three sizes. A needle moves between the upper and lower limits in a forty-five degree, six o'clock to three o'clock pattern. The actual value is displayed in the lower right quadrant of the gauge. Contains thresholding.
Meter bars - Dynamically sized by the user, and may be horizontal or vertical. They are rectangular with a cyan bar and are functionally similar to the gauges. Contains thresholding.
Dynamic position indicator - A pointer which moves, horizontally or vertically, on a bar, proportionally between an upper and lower limit. This pointer may be designated as a caret, cross-hair, and filled or empty square, circle, triangle.
Line - Explicit in primitive name.
Circle - May be filled or empty.
Rectangle - May be filled or empty.
Polygon - Can have a maximum of ten vertices and be empty or filled.

3D Actions & Primitives

There are many ways to construct 3D objects; for example, one can build an image of a car by connecting different surfaces, or an image of a house by using various primitives such as boxes or cylinders. The Display Builder lets a user construct a 3D object by combining 3D wireframe primitives together using data entered through the keyboard.

The following actions can be performed on 3D primitives:

Edit	change any or all data of a selected primitive.
Delete	remove selected primitive(s).
Copy	duplicate selected primitive(s).
Save	save selected primitive(s) to a disk file.
Read	read primitive(s) from a disk file and insert them into the display list.

3D Primitive List (see FIG 6): There are two categories of 3D primitives: Graphic and Control.

The following graphic primitives are visible on the screen:

3D line - Must specify x,y,z coordinates.

Box - Must specify x,y,z coordinates.
Cylinder - Must specify center, diameter, length, number of wireframe lines and angle of rotation (used for defining partial cylinders).
Sphere - Must specify center, radius and number of wireframe lines.

The following control primitives position and orient a 3D object:

Header - Contains distance of the eyepoint from the origin, near and far clipping planes, angle of perspective, and whether or not the display will be z-buffered.
Begin object frame - Contains sysids for securing state vector data for real-time positioning of an object.
End frame - end of object for state data application.
Ref Frame - define the coordinate system used to build a 3D object.
Rotation - Must specify the axis of movement, and degree of movement per pass of the real-time software.
Translation - Must specify the axis of movement, and distance of movement per pass of the real-time software.
Scale - Must specify the axis of movement, and distance/degree of movement per pass of the real-time software.

CONCLUSION

Because of the highly developmental nature of the simulation workstation displays, it was essential that a display builder be created that has the capability to quickly create and modify a display, for evaluation in the simulation, in a short amount of time. This "rapid prototyping", along with a conscious effort to design and write software that is easy to maintain and add prototypes as needed, makes the Display Builder a useful and essential tool in the generation and modification of displays for use in the SES.

ACKNOWLEDGEMENTS

We would like to thank the following people who have made significant contributions over the last two years to the Display Builder: Judy Murphy, Mike Red, and Mike McFarlane.

REFERENCES

1. Foley, J. D., and Van Dam, A., Fundamentals of Interactive Computer Graphics, Addison Wesley, 1982.

2. Linton, M. A. et al., "Composing User Interfaces with Interviews," IEEE Computer, Vol. 22, No. 2, February 1989, pp. 8-22.

3. Myers, B. A. , "User-Interface Tools: Introduction and Survey," IEEE Software, Vol. 6, No. 1, January 1989, pp. 15-23.

4. Newman, W. M., and Sproull, R. F., Principles of Interactive Computer Graphics, McGraw-Hill, 1979.

5. Peltz, D. L., "3-D in Perspective," MacWorld, Vol. 5, No. 12, December 1988, pp. 108-119.

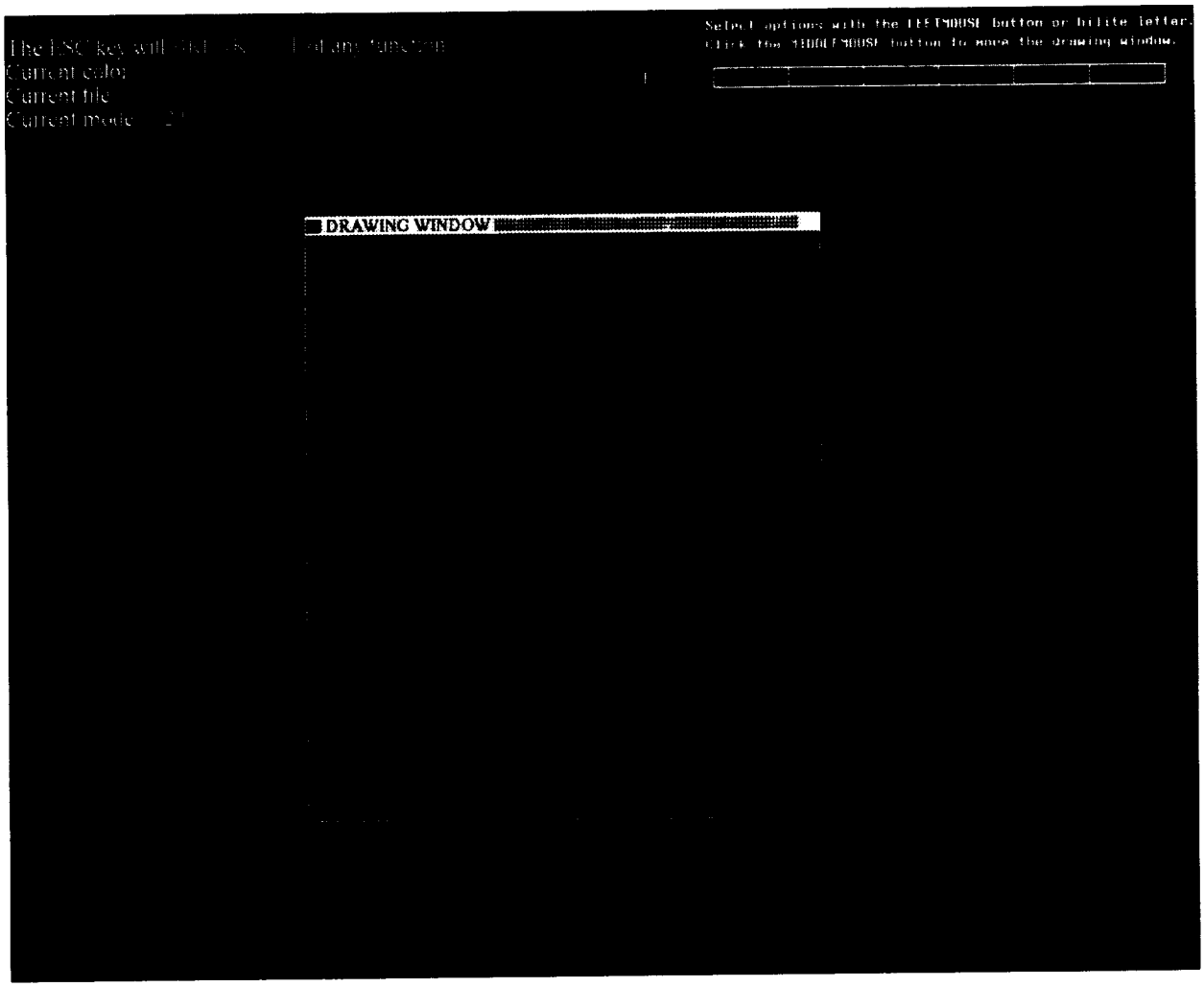

FIG. 1 SCREEN LAYOUT

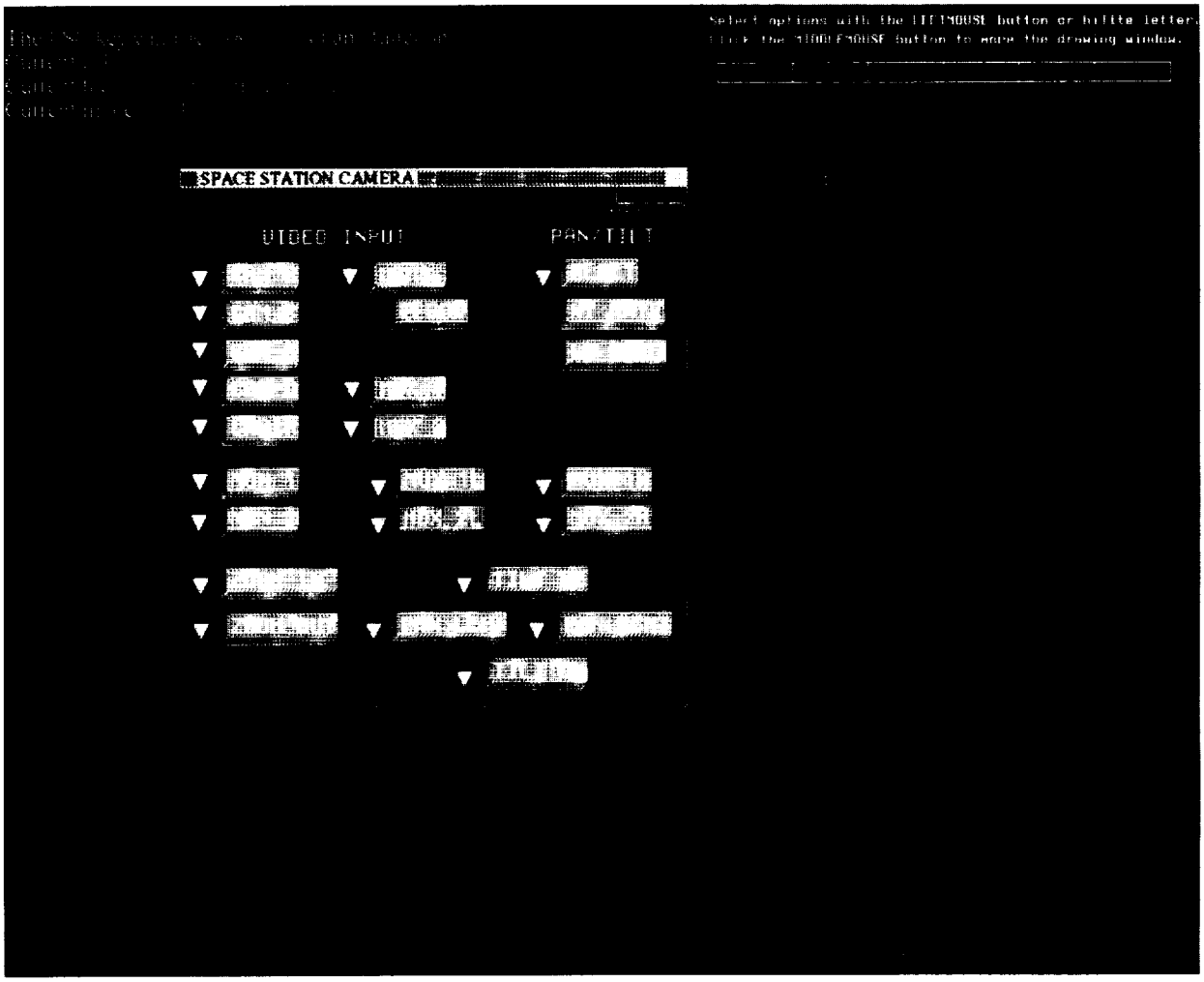

FIG. 2 DIALOG BOX WITH MAIN MENU

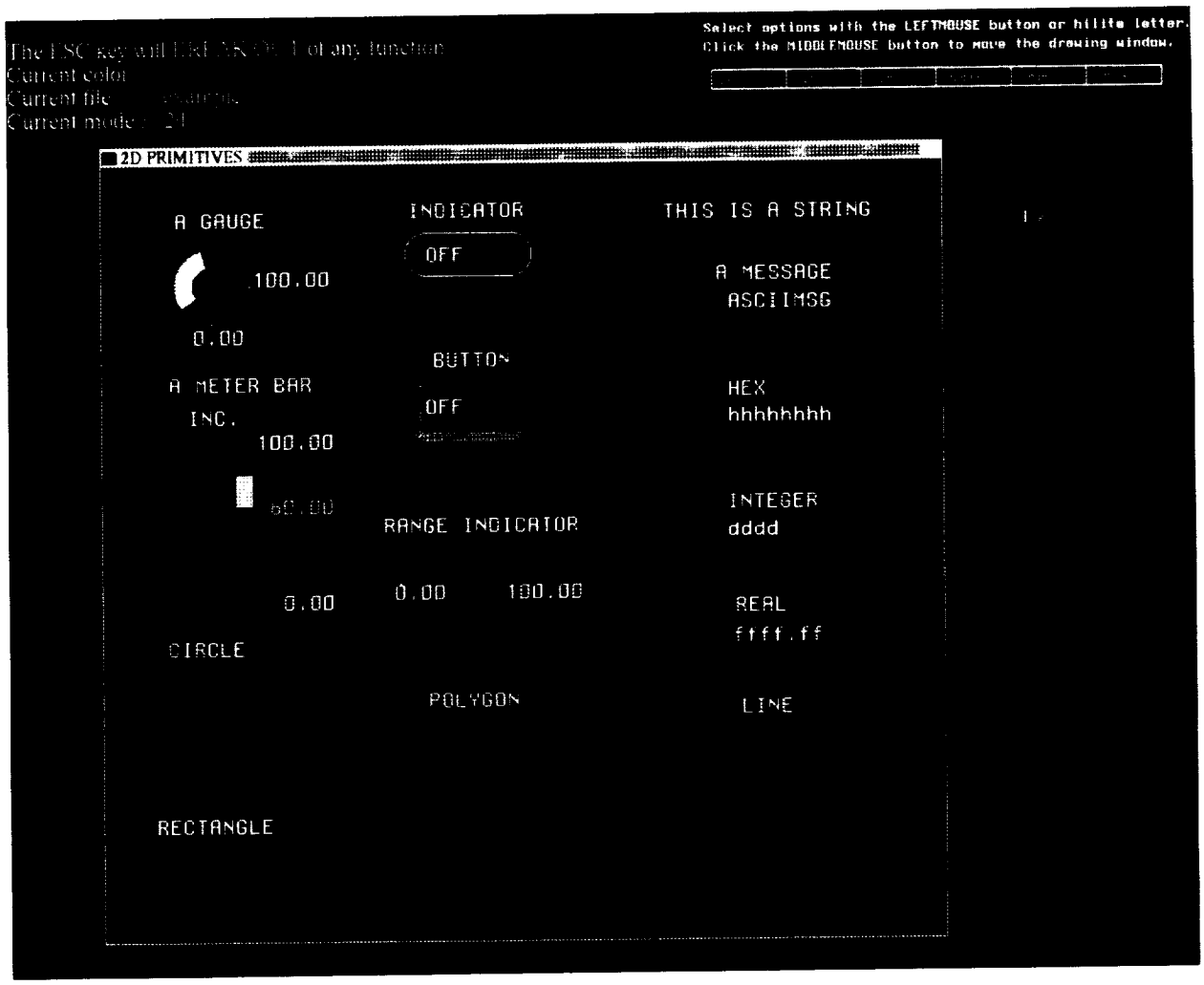

FIG. 3 2D PRIMITIVES

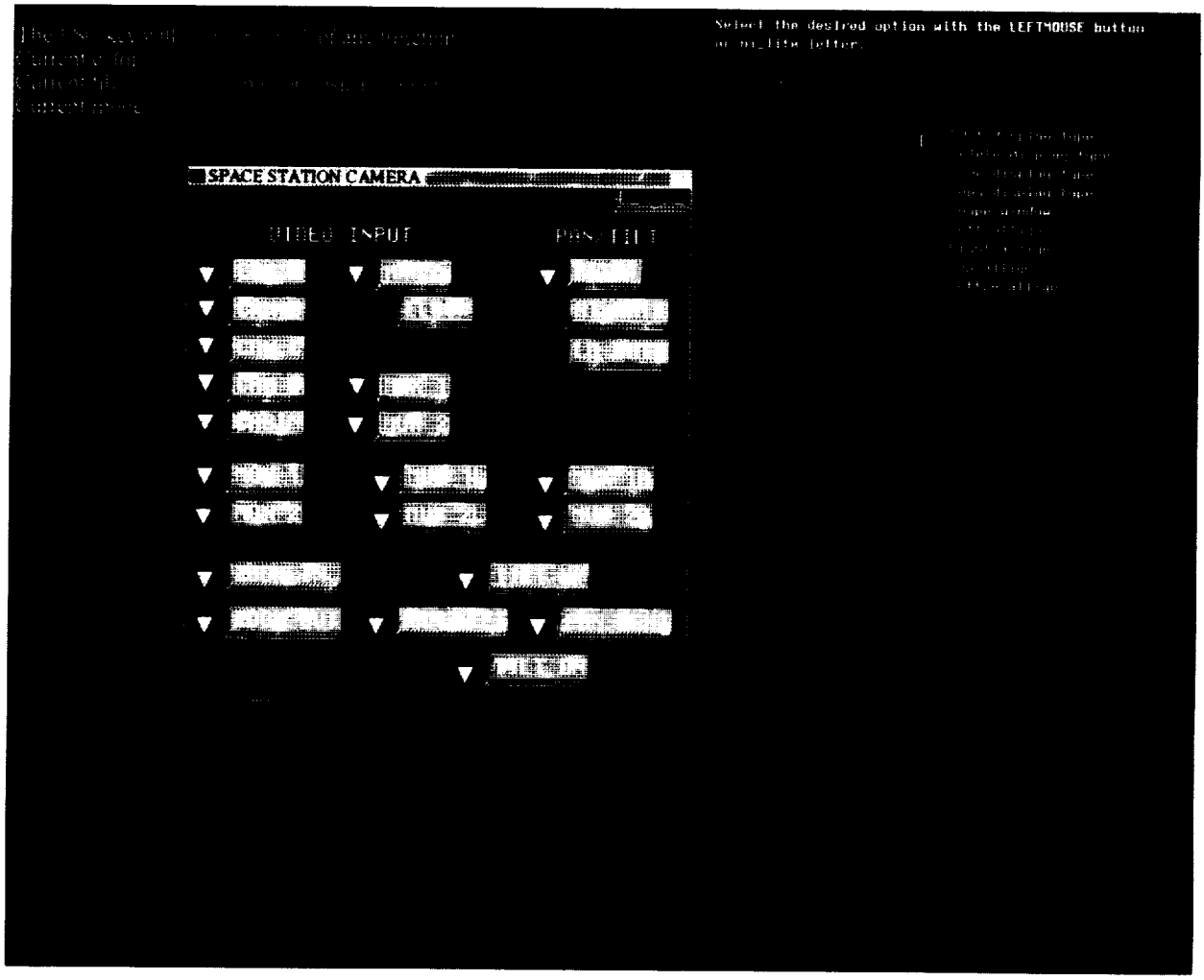

FIG. 4 2D ACTIONS

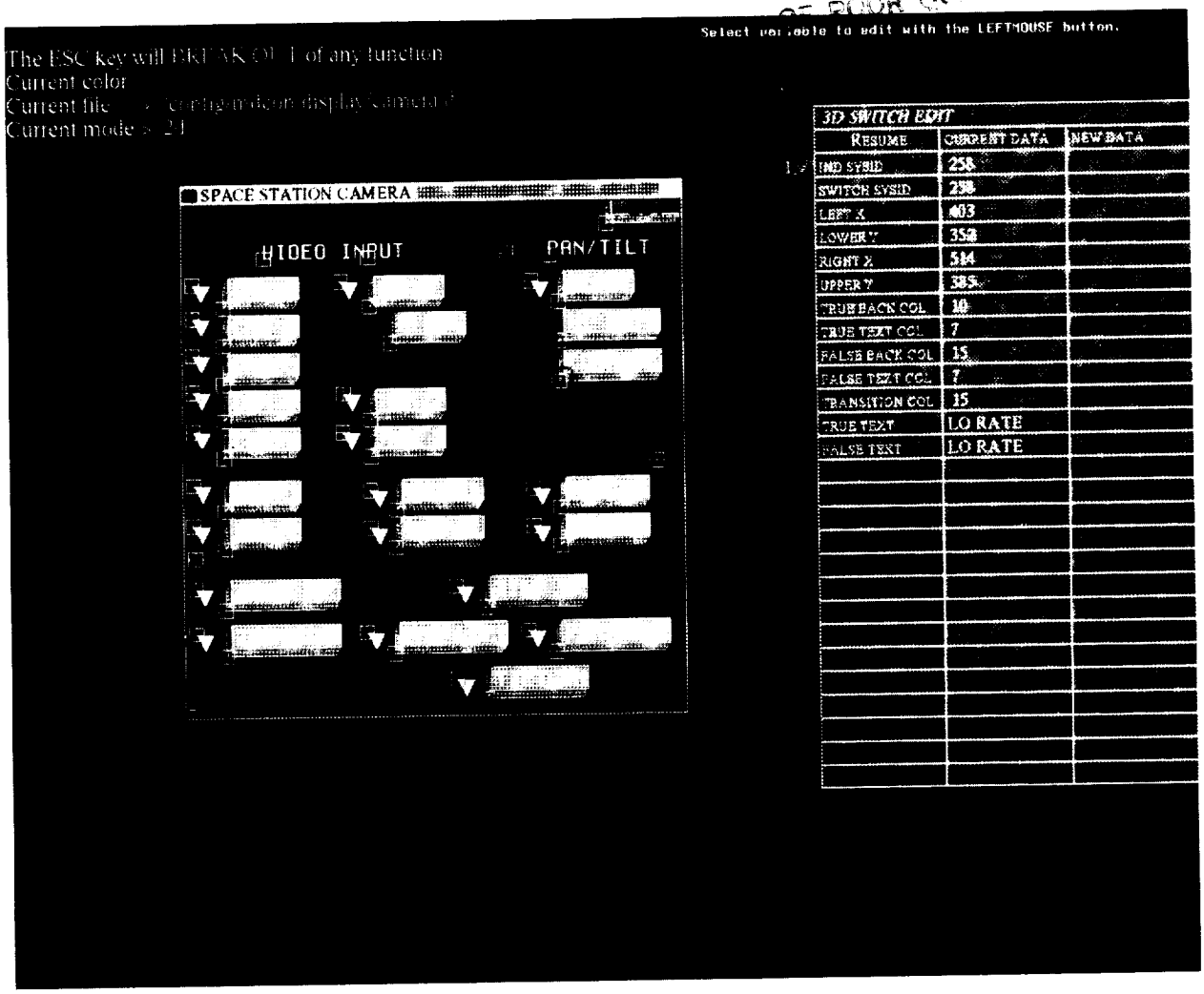

FIG. 5 2D EDITING

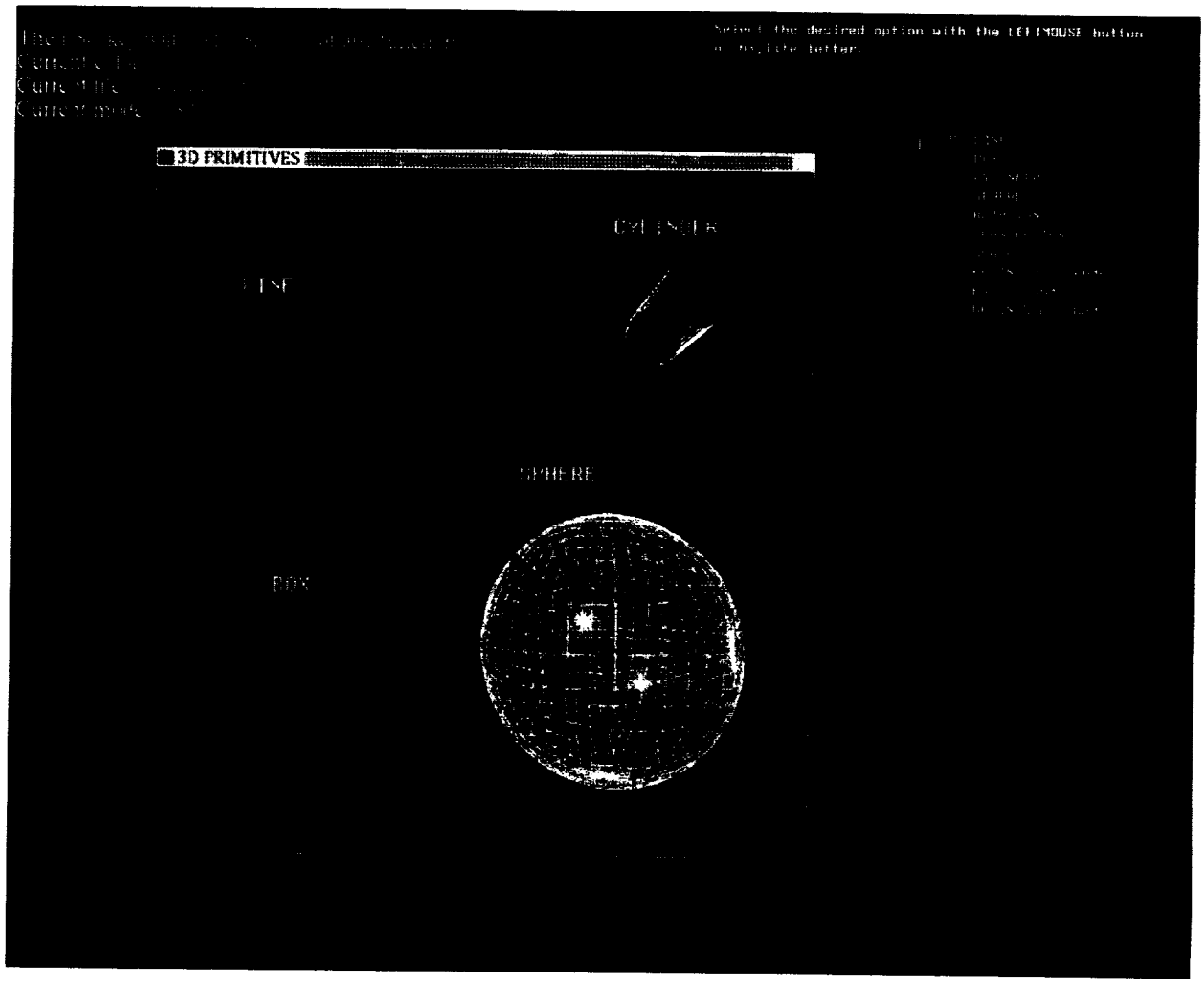

FIG. 6 3D PRIMITIVES