THE REAL TIME DISPLAY BUILDER (RTDB)

Erick D. Kindred and Samuel A. Bailey, Jr.
DUAL & Associates, Inc.
1300 Hercules, Suite 208
Houston, TX 77058
(713) 486-1984

ABSTRACT

The Real Time Display Builder (RTDB) is a prototype interactive graphics tool that builds logic-driven displays. These displays reflect current system status, implement fault detection algorithms in real time, and incorporate the operational knowledge of experienced flight controllers. RTDB utilizes an object-oriented approach that integrates the display symbols with the underlying operational logic. This approach allows the user to specify the screen layout and the driving logic as the display is being built. RTDB is being developed under UNIX in C utilizing the MASSCOMP graphics environment with appropriate functional separation to ease portability to other graphics environments. RTDB grew from the need to develop customized real-time data-driven Space Shuttle systems displays. One display, using initial functionality of the tool, was operational during the orbit phase of STS-26 Discovery. RTDB is being used to produce subsequent displays for the Real Time Data System project currently under development within the Mission Operations Directorate at NASA/JSC. This paper discusses the features of the tool, its current state of development, and its applications.

INTRODUCTION

The Real Time Display Builder (RTDB) is the result of the effort to provide timely display building support to the Real Time Data System (RTDS). RTDS is a prototype project to integrate commercial off-the-shelf telemetry equipment with mini-computer workstations to monitor shuttle systems telemetry data in real time. One of the initial goals of RTDS was to develop a display of the hydraulics system for the Mechanical, Maintenance, and Crew Systems (MMACS) flight controllers for operation during STS-26 Discovery. With three months to define the activity, choose the target MMACS system, layout the display, create the symbols, define and program an operating environment, build the databases, build the screen, and test the system, there was a need for time-saving tools.

To expedite development, the display operating environment and the drawing file format were developed concurrently. The drawing file format was initially built by hand, but was designed with a graphics oriented builder in mind. After the initial display was built and the MMACS flight controllers began their review, it was obvious that the most time-consuming effort would be the fine-tuning of symbol positions, colors, and sizes. At this point, initial functionality of what was to become RTDB was coded. The initial functions addressed the high priority items of position, color, size and rotation. These functions provided the capability to quickly prepare the display for flight monitoring.

As RTDS support of other systems

disciplines has expanded, functionality has been added to accommodate the required uses. RTDB has been used to build a graphical representation of the External Tank Ullage Pressure System for the Booster flight controllers. The display was used to test the ullage pressure fault detection algorithm. Also, RTDB has been used to build two other MMACS displays (Brakes/Tires and Elevons), modify the Hydraulics display, and build three new Integrated Communications Officer (INCO) displays operating with a new Real Time Interactive Display Environment (RTIDE) file format. The RTIDE file format was developed to provide specific functionality for the INCO discipline. All of these new developments were operational during STS-29 Discovery.

APPROACH

RTDB development was conceived as a phased introduction of features. Initial functionality was designed and implemented to satisfy initial project requirements, but were integrated into an environment that facilitated the introduction of new features and the enhancement of old. The development required a highly structured and modularized design with well-defined module interfaces, yet flexible data structures.

The data structures that described objects had to be flexible to accommodate the variety of attributes that determined object behavior. Presently, there are over 60 MMACS and RTIDE objects supported by RTDB. To maintain modularity and flexibility, the object attributes are processed internal to the object function. This relegates uniqueness to the object's code and independence from other parts of the system.

An overview of RTDB functional modularity is presented in Figure 1. The major levels are: the User Interface, the Menus, the Processes, the Object Libraries, the Translator, and the Graphics Libraries. The structure is top-down with higher levels deriving functionality from lower levels. The User Interface defines the user's operating environment. The primary interface to RTDB is through a mouse and minimal keyboard data entry. The mouse determines functionality through menu selection.

The Menus define command selection and execution. They act to interpret mouse manipulations and to provide the inputs to command processing.

The Processes direct command execution. The appropriate code is executed as defined by the command string or object record passed from the Menus. This level acts as the link between desired command behavior and the active elements that perform the command. The command behavior is stored as the object attributes; the active elements are executing processes of object instances; and object instances are the definitions of object behavior. This method of object selection, execution, and definition provides the separation necessary to maintain a high degree of modularity.

The Object Libraries consist of the file formats, object attributes, and process definitions. These represent the object-oriented nature of RTDB's operation. The libraries can be thought of as a pool of commands and objects upon which RTDB may call to perform various operations. This is where new functionality will be added. The higher levels will accommodate any additions.

The Translator is a proposed set of macros, functions, and makefile strategies that will facilitate porting RTDB to other

graphics environments (e.g. X-Windows, IBM PC, MacIntosh, MS-Windows, etc.). This level is still in the conceptual stage. It will consist of macro definitions and function calls that map the various graphics primitives into a consistent set of function names and argument lists. The makefile strategies will build an appropriate run-time module for the target system configuration. With a proposed conversion of RTDS to the X-Windows environment, this level will become a necessity.

The Graphics Libraries are a proposed set of graphics environments that will be supported. Currently the MASSCOMP Graphics is the only supported environment. Conversions to other environments will allow more people and machines to participate in the development process.

FUNCTIONS

RTDB is a mouse and keyboard operated, menu-driven, interactive, graphics application development tool that builds displays that operate under the RTDS environment. The best way to discuss its functionality is to trace through the menu hierarchy. Figure 2 depicts the menu hierarchy. It shows current functionality, current development (+), and proposed development (*).

The Main Menu provides the user access to subsequent functions. A menu item must be selected to transfer control to another functional mode.

FILE allows the user to retrieve and store drawing files. Drawing files contain the symbol records that define a display. A proposed feature is to retrieve and display scanned images. These images will act as backdrops for displays or become mouse sensitive with the placement of additional mouse sensitive regions.

COLOR allows the user to select a particular color from a color map, define a new color map, or select from a pre-defined set of color maps.

EDIT allows the user to modify any object attribute, display invisible objects or symbols, and to undo a previous change.
ADD allows the user to add a new object to a display. Available objects are displayed in a pop-up window. The object is selected and placed with the mouse. Multiple objects may be selected and placed while in this mode.

COPY allows the user to duplicate any visible object. After the initial object selection, subsequent left button clicks will place multiple copies. This mode must be exited and reentered to select a new object.

DELETE allows the user to remove an object from the display. Multiple objects may be deleted while in this mode.

MOVE allows the user to reposition any visible object. The object will be repositioned with subsequent left mouse button clicks.

EXIT allows the user to exit the RTDB environment. The user may elect to exit with a save or a no save of the current display buffer contents.

MOVE BEHIND allows the user to reposition an object in the drawing file. The source object's record is physically placed prior to the target object's record in the drawing file. This results in the source object being drawn prior to the target object and allows the target to be drawn on the top of the source.

MOVE IN FRONT OF allows the user to reposition an object in the drawing file. The source

object's record is physically placed after the target object's record in the drawing file. This results in the source object being drawn after the target object and allows the source to be drawn on top of the target.

LOGIC is a proposed function that allows the user to build dynamic logic-driven displays from within RTDB. The proposed method utilizes a combination of two existing tools: the Computational Development Environment (CODE), an RTDS Tool, and the C Language Integrated Production System (CLIPS), an expert system language. Displays have already been dynamically modified using CLIPS. The remaining development requires the integration of CODE and CLIPS through the graphical interface of RTDB.

OUTPUT is a proposed function that allows the user to print the display image to a laser printer.

CREATE OBJECT is a proposed function that allows the user to interactively build a new object. Primitive drawing objects can be combined with existing objects and stored as a new object. The object attributes will be created. The new object definition will be added to RTDB at the Object Library level.

APPLICATIONS

A tool that embodies the concepts of graphics, logic, and databases has a wide area of applicability. The most obvious being process monitoring in which a system is graphically modeled with sensing and control points highlighted. This tool would be applicable to the Space Shuttle, the Space Station, oil refineries, building environmental control, computer integrated manufacturing, etc.

Other uses include simulation, system testing, and training. Simulation follows from system modeling and leads to system testing. Verification of system components and/or data sets can be verified with the use of a simulation. An extension of simulations and models is their use as training tools. What better way to conduct training than to let the student build the system, component by component, specifying component linkages and operating parameters.

The evolution of RTDB as a graphics application builder is depicted in Figure 3. The RTDB will act as a conduit through which utilities will be integrated into a comprehensive application. Those utilities will be an applications developer providing expert system knowledge, RTIDE providing special object and symbol definitions, the RTDS Tool Set providing system utilities, and CLIPS providing an expert system environment. Graphical expert system applications developed in this manner can provide a consistent, controllable applications interface to RTDS, its data sources, and flight controllers.

CONCLUSION

RTDB is an evolving tool, growing to meet the graphical needs of a complex environment. As flight control techniques change and incorporate more graphical displays, the need to develop new displays, convert old displays, and preserve expert knowledge will require the development and use of new tools and techniques. RTDB represents a step in this new direction.

ACKNOWLEDGEMENTS

We would like to express our appreciation to John Muratore and the entire RTDS Team for providing a fertile environment in which new ideas can grow.

Also, we would like to thank the INCO, MMACS, BOOSTER, and EECOM flight controllers for their invaluable inputs.

REFERENCES

1. Aldus Corporation, "Tag Image File Format Specification Revision 5.0," Aldus Corp., Seattle, Washington, and Microsoft Corp., Redmond, Washington, August, 1988.

2. Cox, Brad, OBJECT-ORIENTED PROGRAMMING: AN EVOLUTIONARY APPROACH, Addison-Wesley, Reading, Massachusetts, 1986.

3. Hertzel, William, THE COMPLETE GUIDE TO DOFTWARE TESTING, QED Information Sciences, Inc., Wellesley, Massachusetts.

4. Hopgood, F. R. A., Duce, D. A., Gallop, J. R., Sutcliffe, D. C., INTRODUCTION TO THE GRAPHICAL KERNEL SYSTEM (GKS), 2nd ed., No. 28, Harcourt Brace Jovanovich, London, England, 1986.

5. Jaeschke, Rex, PORTABILITY AND THE C LANGUAGE, Hayden Books, Indianapolis, Indiana, 1989.

6. Johnson, Nelson, ADVANCED GRAPHICS IN C: PROGRAMMING AND TECHNIQUES, Osbourne McGraw-Hill, Berkeley, California, 1987.

7. Jones, Oliver, INTRODUCTION TO THE X WINDOW SYSTEM, Prentice Hall, Englewood Cliffs, New Jersey, 1989.

8. Kalvelage, Thomas, Murray, Sarah, and Guzzo, Michael, "Real Time Interactive Display Environment: Version 0.9 User's Guide," unpublished memorandum, NASA Johnson Space Center, Houston, Texas, October, 1988.

9. Meyer, Bertrand, OBJECT-ORIENTED SOFTWARE CONSTRUCTION, Prentice Hall, New York, New York, 1988.

10. "MCCU Display Builder/Manager Level B/C Requirements," No. JSC-12348, NASA Johnson Space Center - Mission Support Directorate, Houston, Texas, September, 1988.

11. Miller, Edward, Howden, William E., TUTORIAL: SOFTWARE TESTING & VALIDATION TECHNIQUES, 2nd ed., Computer Society Press.

12. Muratore, John, et. al., "Real Time Expert System Prototype for Shuttle Mission Control," Second Annual Workshop on Space Operations Automation and Robotics (SOAR '88), Dayton, Ohio, July, 1988.

13. Riley, Gary, Culbert, Chris, Savely Robert, and Lopez, Frank, "CLIPS: An Expert System Tool for Delivery and Training, "Third Conference on Artificial Intelligence for Space Applications, Huntsville, Alabama, November, 1987.

14. Robinson, Phillip, "Power to the Process," COMPUTER GRAPHICS WORLD, Vol. 12, No. 3, March, 1989, pp. 71-76.

15. Robinson, Phillip, "CIM's Missing Link: Object-Oriented Databases," COMPUTER GRAPHICS WORLD, Vol. 11, No. 10, October, 1988, pp. 53-58.

16. Salmon, Rod, and Slater, Mel, COMPUTER GRAPHICS SYSTEMS AND CONCEPTS, Addison-Wesley, Reading, Massachusetts, 1987.

17. Schmucker, Kurt, "Using Objects to Package User Interface Functionality," JOURNAL OF OBJECT-ORIENTED PROGRAMMING, Vol. 1, No. 1, April, 1988, pp. 40-45.

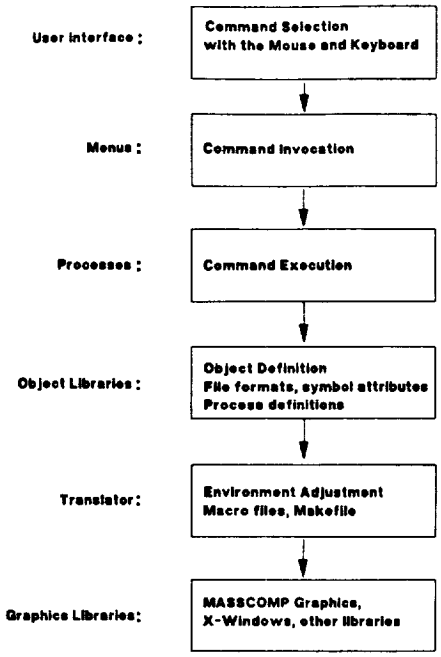

Figure 1 - Design Philosophy

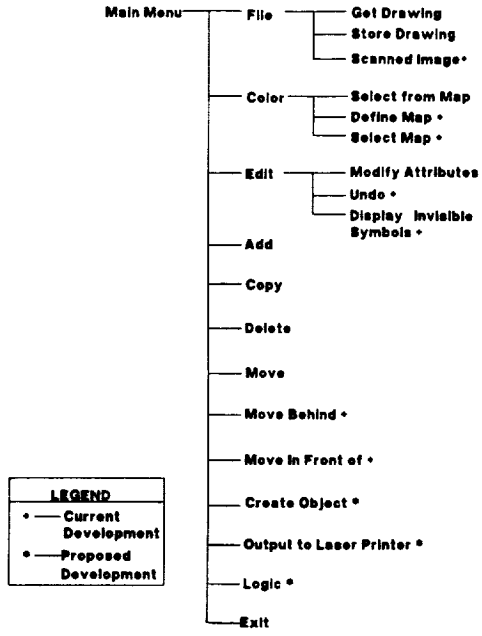

Figure 2 - Functions

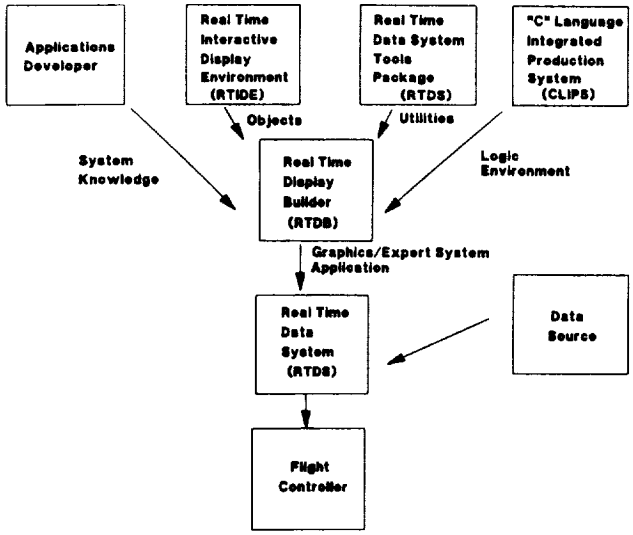

Figure 3 - RTDB Interface to RTDS