The Real Time Interactive Display Environment (RTIDE), a Display Building Tool Developed by Space Shuttle Flight Controllers.

Thomas A. Kalvelage

Rockwell Shuttle Operations Company
NASA Johnson Space Center
Houston,TX

ABSTRACT

NASA's Mission Control Center, located at Johnson Space Center, is incrementally moving from a centralized architecture to a distributed architecture. Starting with STS-29, some host-driven console screens will be replaced with graphics terminals driven by workstations. These workstations will be supplied realtime data first by the Real Time Data System (RTDS), a system developed in-house, and then months later (in parallel with RTDS) by interim and subsequently operational versions of the Mission Control Center Upgrade (MCCU) software package. The Real Time Interactive Display Environment (RTIDE) was built by Space Shuttle flight controllers to support the rapid development of multiple new displays to support Shuttle flights. RTIDE is a display building tool that allows non-programmers to define object-oriented, event-driven, mouseable displays. Particular emphasis was placed on upward compatibility between RTIDE versions, ability to acquire data from different data sources, realtime performance, ability to modularly upgrade RTIDE, machine portability, and a clean, powerful user interface. The paper discusses the operational and organizational factors that drove RTIDE to its present form, the actual design itself, simulation and flight performance, and lessons learned in the process.

Key words: Space Shuttle, Mission Control Center, display building tool, RTDS.

INTRODUCTION

The U.S. Space Shuttle is monitored and controlled from the Mission Control Center (MCC) at NASAs Johnson Space Center (JSC) in Houston, Texas. The flight controllers involved in realtime interaction with the Shuttle work for the Systems Division of the Mission Operations Directorate (MOD).

In the MCC, the Shuttle telemetry is fed into a large minicomputer (the Telemetry Preprocessor Computer, or TPC). This machine decommutates the stream and passes it to a mainframe, the Mission Operations Computer (MOC). The MOC does simple limit checking and drives all the displays used by the flight controllers.

Workstations are used in the MCC to process offline programs. Flight controllers and support personnel have written many general and discipline-specific applications for these machines.

INCO Expert System Project

John Muratore, a NASA flight controller, began the INCO Expert System Project (IESP) in 1986 (INCO is the callsign for the Instrumentation and Communication Officer front room flight control position). This project's goal was to develop and test realtime rule-based expert system applications in an operational environment, i.e., during a Shuttle mission.

Because of safety considerations, the project could not use the MOC or TPC. To get realtime shuttle telemetry into a workstation, a Loral ADS-100 off-the-shelf telemetry processor was used. It decommutated the data stream and passed the data to the workstation, where it was moved to an applications interface with custom-built software. This entire system was called the Real Time Data System (RTDS), and it delivered realtime data to MCC workstations years earlier than previously planned.

RTDS and a set of hand-built application programs were used successfully on STS-26. These applications were certified for use in making critical flight calls during ascent.

Impact of Early Delivery of Realtime Data to Workstations.

To begin exploring the possibilities of improved displays, it was decided to remove a few MOC-driven CRTs from consoles and replace them with RTDS-driven graphics terminals.

The author, as a flight controller whose primary CRT was to be replaced, and as an IESP applications programmer, volunteered to write a few specific display applications. The original intent was to hand-code one or two narrowly focussed applications.

The idea of replacing CRTs with workstation terminals gained favor, and more CRTs were scheduled for replacement, including one of the INCOs CRTs. The INCO is a primary, front-room flight controller, and needs to monitor a large number of systems. It would be impractical to hand-code all the displays the INCO would need, so the author began building a tool (called the Real Time Interactive Display Environment, or RTIDE). Originally, this tool was to be a programmers toolkit, allowing rapid development of hardcoded displays. An internal survey was taken to determine requirements.

OPERATIONAL DESIGN CONSIDERATIONS

In general, the displays that RTIDE produced had to satisfy the users. To support this broad guideline, specific requirements were

drawn up.

The user interface had to be intuitive, consistant, and reliable.

To reduce console clutter, the mouse was chosen as the primary input device.

To reduce the chance of flight controller confusion, all mouse buttons had to be treated identically.

To reduce the possibility of selecting the wrong mouseable object, RTIDE had to inform the user when the mouse cursor was over a object (absolutely required, due to safety concerns).

RTIDE had to allow the user to interrogate the display for additional data.

RTIDE had to provide a consistent method of passing information to the user.

RTIDE had to show data in a variety of ways: as a digital value with highlighting when limits are exceeded; as a symbolic message when a value is zero or nonzero; in graphical plot form; and in bar graph form. All these had to make maximum use of color graphics.

RTIDE had to be able to support display of dynamic schematics, with lines and boxes driven by telemetry.

MAINTENANCE DESIGN CONSIDERATIONS

RTIDE was designed to provide a powerful user interface, but other considerations had higher priority. RTIDE would be maintained by flight controllers whose primary job was flight control, not software and data file maintenance. Maintenance phase costs had to be reduced to a minimum.

RTIDE displays had to be buildable by nonprogrammers. There were too many displays to be done by the limited number of flight controller programmers.

RTIDE had to be upwardly compatable with display definition files. Having to change display definition files because of changes to RTIDE is unacceptable.

RTIDE had to be easily expanded. Not only would this help the RTIDE manager incrementally improve the system, but it helps other disciplines who build graphical objects on their own.

RTIDE display definition files had to allow embedded comments. With this, the documentation of a particular display can be included in the display definition file. Then the file contains the entire description of the display and no costly parallel documentation need be maintained.

ORGANIZATIONAL DESIGN CONSIDERATIONS

Although RTIDE would be built and maintained by flight controllers, the hardware RTIDE ran on and the data sources RTIDE used generally were not. Consideration must be given to future changes to RTIDEs environment.

Multiple Data Sources

RTIDE had to be able to access different data sources. RTDS, an internal MOD development system, was the original data source. However, in 1990 the production Mission Control Center Upgrade (MCCU) realtime data interface will become available, and will have to be used.

In addition, a data retrieval system called Near Real Time (NRT) already operates in the MCC workstations, and RTIDE should run off of NRT data files. Besides providing a method of reviewing flight events, this will assist in training flight controllers.

Hardware Independance

RTIDE had to be hardware independant. Currently the MCC is transitioning from its five-year-old Masscomps to new models, requiring software changes to many offline programs.

Configuration Management

Configuration management was a key factor in basic systems design of RTIDE. Flight controllers do not have system manager authority over the machines they use. RTIDE was designed to be as simple and robust as possible, to increase reliability and to reduce the chance of misconfiguration.

Time Constraint

RTIDE was started in 6/88, and had to be ready for STS-29, in 2/89.

RTIDE DESIGN

The basic structure had to be powerful enough to support any reasonable improvement, and simple enough to be maintained by novice programmers unfamiliar with RTIDE.

Organization

The emphasis was on simplicity. RTIDE is a single process, its executable located in a single file, reducing the chance of having file permissions changed or files deleted. It also eliminates having to have the workstation configured a particular way for interprocess communication. RTIDE uses a single display definition file for each display. All the documentation for the display can be included in the same display definition file, eliminating the lag between documentation and implementation.

Event Driven

RTIDE is event-driven. The user (by pressing a key or mouse button, or by moving the mouse), the data sources (by supplying new data), or the operating system (by sending interrupts) may all trigger events that are detected by RTIDE. Event flags are either used directly by RTIDE or sent to the object currently selected by the user's mouse cursor. Event types can be added as desired.

Object Oriented

In the graphics sense, RTIDE is object-oriented. The dynamic symbols on the screen driven by data are objects. RTIDE keeps track of which object the mouse cursor is on, and sends event flags to the object when appropriate.

The hard code determining each objects behavior consists of five standard functions that are located in one source file (generally 500-1500 lines long). The behavior of an instance of an object is determined by a data structure maintained by RTIDE. Adding new objects can be done easily by building this file and adding a structure definition to the master include file.

User Interface

The user interface is designed to be highly interactive, using the mouse, and as simple as possible. Interaction is needed to request further data from the display (limit sets, telemetry status, value range, description, etc).

Display Definition File

The ASCII (for ease of maintenance) display definition file specifies the display's initial condition. Each entry is a series of arguments, each setting some variable (e.g., object colors, messages, data source, etc.).

Program Execution

RTIDE begins by opening the display definition file and reading in

the entries there one at a time. Some entrys (screen_size, data_source, etc) are used to configure RTIDE. The static graphics entries are stored in case the screen is refreshed later. Object entries are stored in the object list. Comments are not currently saved. RTIDE then initializes the graphics processor, displays all the objects, and falls into the main loop.

RTIDE moves constantly through a busy wait loop, first looking for events, then reacting to them. Every cycle, RTIDE pauses for less than a tenth of second, allowing the CPU to run other processes. Because pressing a mouse button interrupts this pause, a user can increase RTIDEs CPU usage by rapidly pressing the mouse buttons.

If new data appears at the interface (nominally, once a second), RTIDE begins to update the screen.

To minimize flicker, RTIDE divides its screen update into two seperate cycles, the process cycle and the display cycle. In the process cycle, RTIDE goes through the objects one at time (using each objects process function), using the new data to update the object. If the object needs to be updated, a draw_me flag is set. Then RTIDE goes through the display cycle, looking for draw_me flags. If one is set, RTIDE redraws it using the objects draw function.

Every cycle RTIDE polls the mouse to find its location. RTIDE compares the location with the information in its object list to see if it has entered or left an object. RTIDE, once the mouse cursor enters an object, only looks to see if it has left the object. This limitation prevents displays from using objects inside of objects.

The cycling continues until an object (usually the exit object) forces RTIDE to exit.

FUTURE DEVELOPMENTS

RTIDE continues to be developed, with new objects being developed for new applications. There is more emphasis being placed on telemetry-driven schematics to increase the efficiency of displays.

Right now we are placing all the documentation into the display definition file. That data is not retained by RTIDE. A later version of RTIDE will save that information, so a user can click on an object and see a complete description of what the measurement displayed means.

SIMULATION PERFORMANCE

RTIDE was installed in the MCC console on 2/16/89. After a few weeks to get the display definition files debugged, RTIDE provided high quality displays for flight controllers. Flight controllers particularly like to be able to get more information from the

display on request.

RTIDE will support STS-29 in March of 1989.

LESSONS LEARNED

Small Size

We found that running a display, its supporting algorithms, numerous fault detection algorithms and several other realtime applications does stress the machine. Keeping the display as efficient as possible is necessary to allow the entire workstation to keep up with the data. RTIDEs relative simplicity, originally specified for other reasons, has kept its executable size down to 237Kb (approximately 100Kb of which is the Masscomp graphics library).

Health and Status Messages

Experience has shown that it is vital to avoid misleading flight controllers, and a display should do its part by telling the controller when the data displayed is useable. The status should be more than a simple GO/NOGO; it should give the controller enough information to begin troubleshooting any data problem.

CONCLUSION

A user-friendly display-building tool has been developed. The object-oriented approach allows rapid display building in realtime command and control environments. The highly interactive user interface allows the user to easily access additional data describing the displays. This tool is being used in support of Space Shuttle missions.

ACKNOWLEDGEMENTS

The author would like to thank Rich Rodriguez of NASA, for his patience, encouragement, and support; John Muratore, for the opportunity to work on the INCO Expert System Project; and Mike Guzzo and Sarah Murray, for all the good work they've done on RTIDE.

REFERENCES

1. Muratore, October 1987, Trends in Space Shuttle Telemetry Applications, Proceedings of the International Telemetering Conference.