Simulating Vibrations in a Complex Loaded Structure

The Dynamic Response Computation (DIRECT) computer program simulates vibrations induced in a complex structure by applied dynamic loads. Developed to enable rapid analysis of launch- and landing-induced vibrations and stresses in a space shuttle, DIRECT also can be used to analyze dynamic responses of other structures — for example, the response of a building to an earthquake, or the response of an oil-drilling platform and attached tanks to large ocean waves. For a space-shuttle simulation, the required input to DIRECT includes mathematical models of the space shuttle and its payloads, and a set of forcing functions that simulates launch and landing loads. DIRECT can accommodate multiple levels of payload attachment and substructure as well as nonlinear dynamic responses of structural interfaces. DIRECT combines the shuttle and payload models into a single structural model, to which the forcing functions are then applied. The resulting equations of motion are reduced to an optimum set and decoupled into a unique format for simulating dynamics. During the simulation, maximum vibrations, loads, and stresses are monitored and recorded for subsequent analysis to identify structural deficiencies in the shuttle and/or payloads.

This program was written by Tim T. Cao of **Johnson Space Center**. *For further information, contact the Johnson Commercial Technology Office at (281) 483-3809.*
MSC-23333

Rover Sequencing and Visualization Program

The Rover Sequencing and Visualization Program (RSVP) is the software tool for use in the Mars Exploration Rover (MER) mission for planning rover operations and generating command sequences for accomplishing those operations. RSVP combines three-dimensional (3D) visualization for immersive exploration of the operations area, stereoscopic image display for high-resolution examination of the downlinked imagery, and a sophisticated command-sequence editing tool for analysis and completion of the sequences. RSVP is linked with actual flight-code modules for operations rehearsal to provide feedback on the expected behavior of the rover prior to committing to a particular sequence. Playback tools allow for review of both rehearsed rover behavior and downlinked results of actual rover operations. These can be displayed simultaneously for comparison of rehearsed and actual activities for verification.

The primary inputs to RSVP are downlink data products from the Operations Storage Server (OSS) and activity plans generated by the science team. The activity plans are high-level goals for the next day's activities. The downlink data products include imagery, terrain models, and telemetered engineering data on rover activities and state. The Rover Sequence Editor (RoSE) component of RSVP performs activity expansion to command sequences, command creation and editing with setting of command parameters, and viewing and management of rover resources. The HyperDrive component of RSVP performs 2D and 3D visualization of the rover's environment, graphical and animated review of rover-predicted and telemetered state, and creation and editing of command sequences related to mobility and Instrument Deployment Device (IDD) operations. Additionally, RoSE and HyperDrive together evaluate command sequences for potential violations of flight and safety rules. The products of RSVP include command sequences for uplink that are stored in the Distributed Object Manager (DOM) and predicted rover state histories stored in the OSS for comparison and validation of downlinked telemetry.

The majority of components comprising RSVP utilize the MER command and activity dictionaries to automatically customize the system for MER activities. Thus, RSVP, being highly data driven, may be tailored to other missions with minimal effort. In addition, RSVP uses a distributed, message-passing architecture to allow multitasking, and collaborative visualization and sequence development by scattered team members.

This tool was developed by Brian Cooper, Frank Hartman, Scott Maxwell, Jeng Yen, John Wright, and Carlos Balacuit of Caltech for **NASA's Jet Propulsion Laboratory**. *Further information is contained in a TSP (see page 1).*

This software is available for commercial licensing. Please contact Karina Edmonds of the California Institute of Technology at (818) 393-2827. Refer to NPO-30845.

Software Template for Instruction in Mathematics

Intelligent Math Tutor (IMT) is a software system that serves as a template for creating software for teaching mathematics. IMT can be easily connected to artificial-intelligence software and other analysis software through input and output of files. IMT provides an easy-to-use interface for generating courses that include tests that contain both multiple-choice and fill-in-the-blank questions, and enables tracking of test scores. IMT makes it easy to generate software for Web-based courses or to manufacture compact disks containing executable course software. IMT also can function as a Web-based application program, with features that run quickly on the Web, while retaining the intelligence of a high-level language application program with many graphics. IMT can be used to write application programs in text, graphics, and/or sound, so that the programs can be tailored to the needs of most handicapped persons. The course software generated by IMT follows a "back to basics" approach of teaching mathematics by inducing the student to apply creative mathematical techniques in the process of learning. Students are thereby made to discover mathematical fundamentals and thereby come to understand mathematics more deeply than they could through simple memorization.

This program was written by Robert O. Shelton of **Johnson Space Center,** *and Travis A. Moebes and Anna Beall of Science Applications International Corp. For further information, contact the Johnson Commercial Technology Office at (281) 483-3809.*
MSC-23614

Support for User Interfaces for Distributed Systems

An extensible Java™ software framework supports the construction and operation of graphical user interfaces (GUIs) for distributed computing systems typified by ground control systems that send commands to, and receive telemetric data from, spacecraft. Heretofore, such GUIs have been custom built for each new system at considerable expense. In contrast, the present framework affords generic capabilities that can be shared by different distributed systems. Dynamic class loading, reflection, and other run-time capabilities of the Java language and JavaBeans component architecture enable the creation of a GUI for each new distributed computing system with a minimum of custom effort. By use of this framework, GUI components in control panels and