Software

Software was an integral part in the Space Shuttle hardware systems and it played a vital role in the design and operations of the shuttle. The longevity of the program demanded the on-orbit performance of the vehicle to be flexible under new and challenging environments. Because of the flexibility required, quick-turnaround training, simulations, and virtual reality tools were invaluable to the crew for new operational concepts. In addition, ground operations also benefited from software innovations that improved vehicle processing and flight-readiness testing. The innovations in software occurred throughout the life of the program. The topics in this chapter include specific areas where engineering innovations in software enabled solutions to problems and improved overall vehicle and process performance, and have carried over to the next generation of space programs.

Primary Software

NASA faced notable challenges in the development of computer software for the Space Shuttle in the early 1970s. Only two avionics computers were regarded as having the potential to perform the complex tasks that would be required of them. Even though two options existed, these candidates would require substantial modification. To further compound the problem, the 1970s also suffered a noticeable absence of off-the-shelf microcomputers. Large-scale, integrated-circuit technology had not yet reached the level of sophistication necessary for Orbiter use. This prompted NASA to continue its search for a viable solution.

NASA soon concluded that core memory was the only reasonable choice for Orbiter computers, with the caveat that memory size was subject to power and weight limitations as well as heat constraints. The space agency still faced additional obstacles: data bus technology for real-time avionics systems was not yet fully operational; the use of tape units for software program mass storage in a dynamic environment was limited and unsubstantiated; and a high-order language tailored specifically for aerospace applications was nonexistent. Even at this early juncture, however, NASA had begun developing a high-order software language—HAL/S—for the shuttle. This software would ultimately become the standard for Orbiter operations during the Space Shuttle Program.

Software Capability Beyond Technology Limits

NASA contemplated the number of necessary computer configurations during the early stages of Space Shuttle development. It took into consideration the segregation of flight control from guidance and navigation, as well as the relegation of mechanized aerodynamic ascent/re-entry and spaceflight functions to different machines.

These considerations led to a tightly coupled, synchronized fail-operational/fail-safe computation requirement for flight control and sequencing functions that drove the system toward a four-machine computer complex. In addition, the difficulties NASA faced in attempting to interconnect and operate multiple complexes of machines led to the development of a single complex with central integrated computation.

NASA added a fifth machine for off-loading nonessential mission applications, payload, and system-management tasks from the other four machines. Although this fifth computer was also positioned to handle the additional computation requirements that might be placed on the system, it eventually hosted the backup system flight software.

The space agency had to determine the size of the Orbiter computer memory to be baselined and do so within the constraints of computer design and vehicle structure. Memory limitations posed a formidable

Personal Computer Ground Operations Aerospace Language Offered Engineers a "View"

Personal Computer Ground Operations Aerospace Language (PCGOAL) was a custom, PC-based, certified advisory system that provided engineers with real-time data display and plotting. The enhanced situational awareness aided engineers with the decision-making process and troubleshooting during test, launch, and landing operations.

When shuttle landings first began at Dryden Flight Research Center (DFRC), California, Kennedy Space Center (KSC) engineers had limited data-visualization capability. The original disk operating system (DOS)-based PCGOAL first supported KSC engineers during the STS-34 (1989) landing at DFRC. Data were sent from KSC via telephone modem and engineers had visibility to the Orbiter data on site at DFRC. Firing room console-like displays provided engineers with a familiar look of the command and control displays used for shuttle processing and launch countdown, and the application offered the first high-resolution, real-time plotting capability.

PCGOAL evolved with additional capabilities. After design certification review in 1995, the application was considered acceptable for decision making in conjunction with the command and control applications in the firing rooms and DFRC. In 2004, the application was given a new platform to run on a Windows 2000 operating system.

As the Windows-based version of PCGOAL was being deployed, work had already begun to add visualization capabilities. The upgraded application and upgraded editor were deployed in December 2005 at KSC first and later at DFRC and Marshall Space Flight Center/ Huntsville Operations Support Center.

challenge for NASA early in the development phase; however, with the technological advancements that soon followed came the ability to increase the amount of memory.

NASA faced much skepticism from within its organization, regarding the viability of using a high-order language. Assembly language could be used to produce compact, efficient, and fast software code, but it was very similar in complexity to the computer's machine language and therefore required the programmer to understand the intricacies of the computer hardware and instruction set. For example, assembly language addressed the machine's registers directly and operations on the data in the registers directly.

While it might not result in as fast and efficient a code, using a high-order programming language would provide abstraction from the details of the computer hardware, be less cryptic and closer to natural language, and therefore be easier to develop and maintain. As the space agency contracted for the development of HAL/S, program participants questioned the software's ability to produce code with the size, efficiency, and speed comparable to those of an assembly language program. All participants, however, supported a top-down structured approach to software design.

To resolve the issue and quell any fears as to the capability of HAL/S, NASA tested both options and discovered that the nominal loss in efficiency of the high-order language was insignificant when compared to the advantages of increased programmer productivity, program maintainability, and visibility into the software. Therefore, NASA selected HAL/S for all but one software module (i.e., operating system software), thus fulfilling the remaining baselined requirements and approach.

Operating Software for Avionics System

The Orbiter avionics system operation required two independent software systems with a distinct hierarchy and clear delegation of responsibilities. The Primary Avionics Software System was the workhorse of the two systems. It consisted of several memory loads and performed mission and system functions. The Backup Flight System software was just that: a backup. Yet, it played a critical role in the safety and function of the Orbiter. The Backup Flight System software was composed of one memory load and worked only during critical mission phases to provide an alternate means of orbital insertion or return to Earth in the event of a Primary Avionics Software System failure.

Primary Avionics Software System

The Primary Avionics Software System performed three major functions: guidance, navigation, and control of the vehicle during flight; the systems management involved in monitoring and controlling vehicle subsystems; and payload—later changed to vehicle utility—involving preflight checkout functions.

The depth and complexity of Orbiter requirements demanded more memory capacity than was available from a general purpose computer. As a solution, NASA structured each of the major functions into a collection of programs and capabilities needed to conduct a mission phase or perform an integrated function. These collections were called "operational sequences," and they formed memory configurations that were loaded into the general purpose computers from on-board tape units. Memory overlays were inevitable; however, to a great extent NASA structured these overlays only in quiescent, non-dynamic periods.

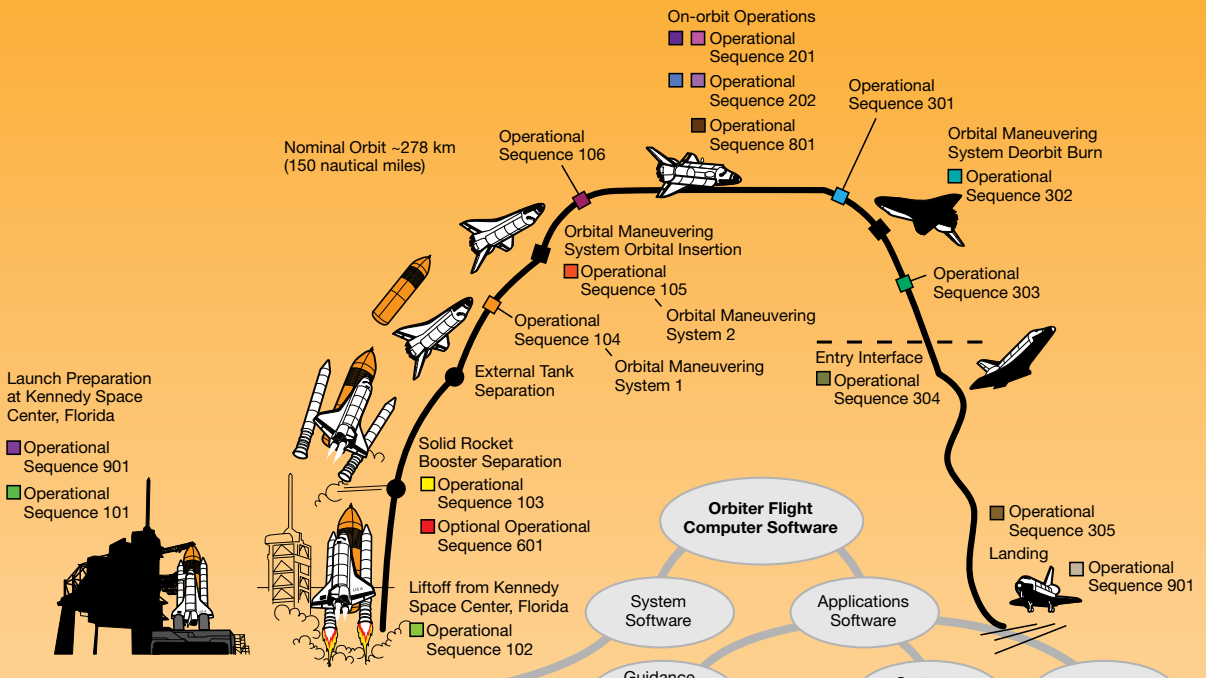
The substructure within operational sequences was a choreographed network consisting of major modes, specialist functions, and display functions. Major modes were substructured into blocks that segmented the processes into steps or sequences. These blocks were linked to cathode ray tube display pages so the crew could monitor and control the function. The crew could initiate sequencing through keyboard entry. In certain instances, sequencing could be initiated automatically by the software. Blocks within the specialist functions, initiated by keyboard entry, were linked to cathode ray tube pages. These blocks established and presented valid keyboard entry options available to the crew for controlling the operation or monitoring the process. Major modes accomplished the primary functions within a sequence, and specialist functions were used for secondary or background functions. The display functions, also initiated by keyboard input, contained processing necessary to produce the display and were used only for monitoring data processing results.

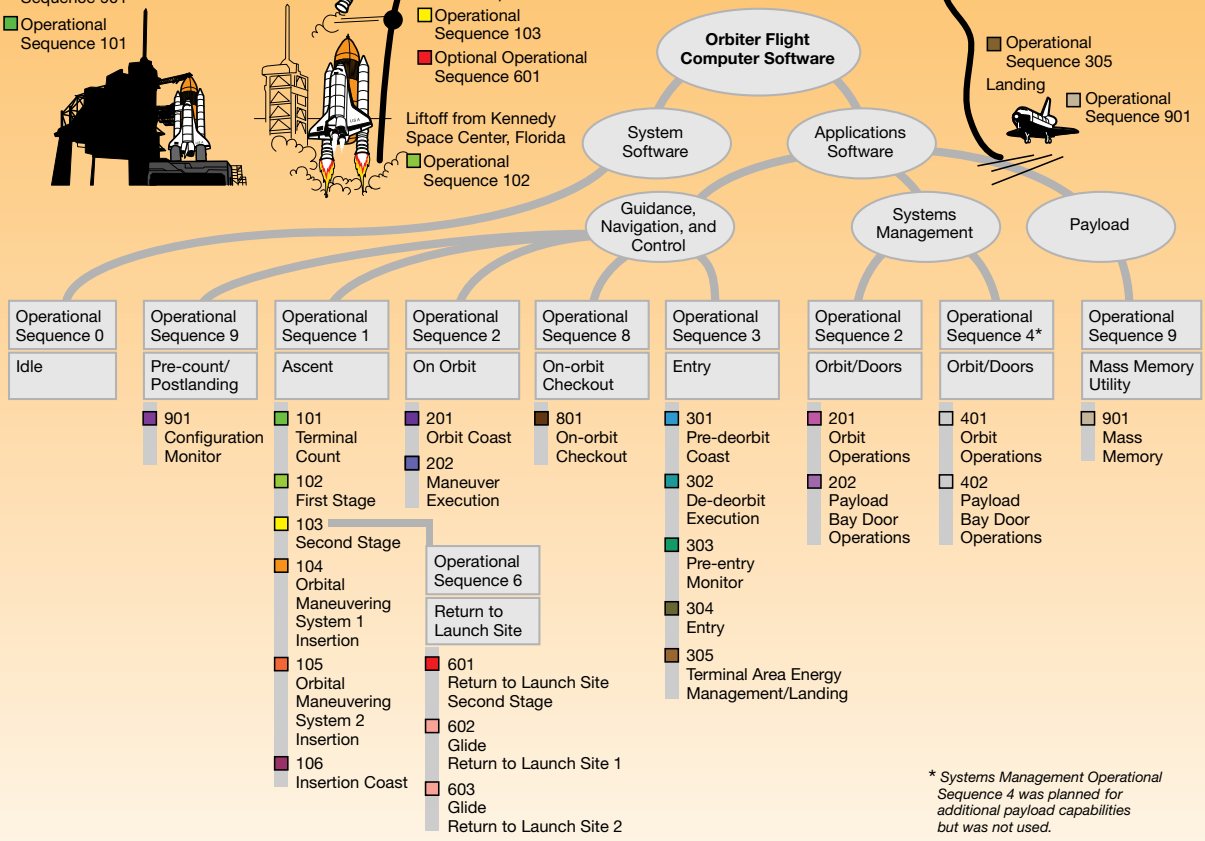
Backup Flight System

The Backup Flight System remained poised to take over primary control in the event of Primary Avionics Software System failure, and NASA thoroughly prepared the backup system for this potential problem. The system consisted of the designated general purpose computer, three backup flight controllers, the backup software, and associated switches and displays.

As far as designating a specific general purpose computer, NASA did not favor any particular one over the others—any of the five could be designated the backup machine by appropriate keyboard entry. The designated computer would request the backup

Mission Phase With Corresponding Operational Sequences and Major Modes





* *Systems Management Operational Sequence 4 was planned for additional payload capabilities but was not used.*

Due to computer memory limitations, the flight software was divided into a number of separate programs called operational sequences. Each sequence provided functions specific to a particular mission phase and were only loaded into memory during that phase of flight.

The Ground Launch Sequencer Orchestrated Launch Success

During launch countdown, the ground launch sequencer was like an orchestra's conductor. Developed in 1978, the sequencer was the software supervisor of critical command sequencing and measurement verification from 2 hours before launch time to launch time and through safing, thus assuring a steady and an appropriate tempo for a safe and successful launch.

Engineered to expedite and automate operations and maximize automatic error detection and recovery, the ground launch sequencer focused on "go/no-go" criteria. Responding to a no-go detection, it could initiate a countdown hold, abort, or recycle or contingency operations. While controlling certain monitoring aspects, the sequencer did not reduce the engineer's capability to monitor his or her system's health/integrity; however, by assuming command responsibility, it integrated launch requirements and activities, and reduced communication traffic and required hardware. Manual intervention was available for off-nominal conditions.

The four ground launch sequencer components included: exception monitoring; sequencer; countdown clock control; and safing. For exception monitoring, the sequencer continuously monitored more than 1,200 measurements. If a measurement violated its expected value, the sequencer checked whether the measurement was part of a voting logic group. If voting failed, it automatically caused the countdown to hold at the next milestone or abort the countdown.

The sequencer provided a single point of control during countdown, issuing all commands to ground and flight equipment from the designated period called T minus 9 minutes (T=time) through liftoff. It verified events required for liftoff. If an event wasn't completed, an automated hold/recycle was requested.

Clock control provided the required synchronization between ground and vehicle systems and managed countdown holds/recycles. Clock control allowed the sequencer to resume the countdown after a problem was resolved. The safing component halted the Orbiter's on-board software and, based on the progression of the sequencer, commanded ground and flight systems into a safe configuration for crew egress.

Launch countdown operations in Firing Room 4 at Kennedy Space Center, Florida.

software load from mass memory. The backup computer would then remain on standby. During normal operations, when the primary system controlled the Orbiter, the backup system operated in "listen" mode to monitor and obtain data from all prime machines and their assigned sensors. By acquiring these data, the Backup Flight System maintained computational currency and, thus, the capability to assume control of the Orbiter at any time.

NASA independently developed and coded the software package for the Backup Flight System as an added level of protection to reduce the possibility of generic software errors common to the primary system. The entire Backup Flight System was contained in one memory configuration, loaded before liftoff, and normally maintained in that machine.

Success—On Multiple Levels

NASA overcame the obstacles it faced in creating the shuttle's Primary Avionics Software System through ingenuity and expertise. Even technology that was current during the initial planning stages did not impose limits on what the space agency could accomplish in this area. NASA succeeded in pushing the boundaries for what was possible by structuring a system that could handle multiple functions within very real parameters. It also structured a backup support system capable of handling the demands of spaceflight at a critical moment's notice.

Integrated Extravehicular Activity/Robotics Virtual Reality Simulation

As the Space Shuttle Program progressed into the 1990s, the integration of extravehicular activity (EVA) and robotics took on a whole new importance when Hubble Space Telescope servicing/repair (first flight 1993) and space-based assembly of the International Space Station (ISS) tasks were realistically evaluated.

Two motivating factors influenced NASA's investigation into the potential use of virtual reality technology that was barely in its infancy at that time. The first factor was in response to a concern that once Hubble was deployed on orbit future astronauts and flight controllers would not have easy access to the telescope to familiarize themselves with the actual hardware configuration to plan, develop, and review servicing procedures.

The second factor was based on previous on-orbit experience with the interaction and communication between EVA crew members and Shuttle Robotic Arm operators. NASA discovered that interpreting instructions given by a crew member located in a foot restraint on the end of the robotic arm was not as intuitive to the arm operator as first thought, especially when both were not in the same body orientation when giving or receiving commands. The EVA crew member could, for example, be upside down with respect to the robotic arm operator in microgravity. Therefore, the command to "Move me up" left the arm operator in a quandary trying to decide what "up" actually meant.

NASA Embraces Advances in Virtual Reality

It was at this same time in the early 1990s that virtual reality hardware started to enter the commercial world in the form of head-mounted displays, data gloves, motion-tracking instruments, etc.

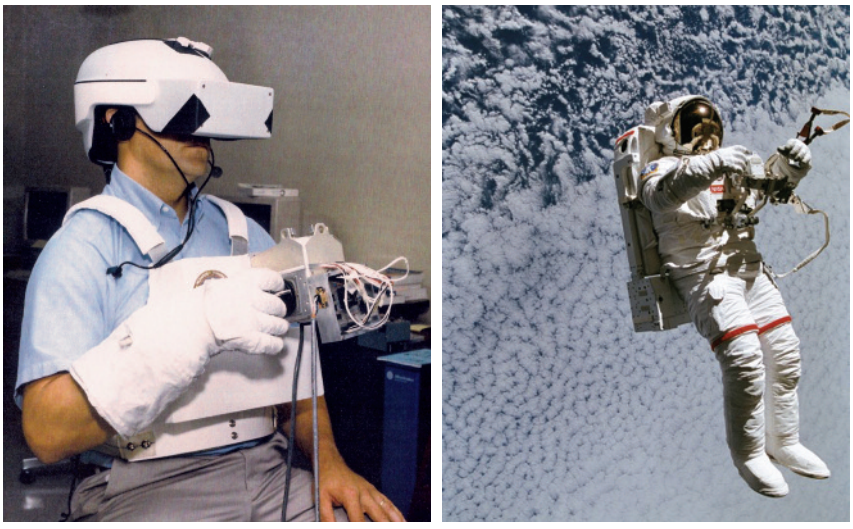
In the astronaut training world, no facility allowed an EVA crew member to ride on a robotic arm operated by another crew member in a realistic space environment. The Water Emersion Test Facility at Johnson Space Center (JSC) in Houston, Texas, provided a training arena for EVA crew members, but the confined space and the desire to not require subjects to be heads down for more than very short periods of time did not allow for suitable integrated training between the EVA crew and the robotic arm operators. Likewise, the Manipulator Development Facility's hydraulic arm and the computer graphic-based robotic arm simulators at JSC were not conducive for EVA crew interaction.

Virtual reality provided a forum to actually tie those two training scenarios together in one simulation. Working closely with the astronaut office, NASA engineers took commercially available virtual reality hardware and developed the computer graphic display software and across-platform communications software that linked into existing "man-in-the-loop" robotic arm computer simulations to produce an integrated EVA/robotics training capability.

Virtual Reality Is Put to the Test

The first use of these new capabilities was in support of crew training for Space Transportation System (STS)-61 (1993)—the Hubble Space Telescope servicing mission. The virtual reality simulation provided a flight-like environment in which the crew was able to develop and practice the intricate choreography between the Shuttle Robotic Arm operator and the EVA crew member affixed to the end of that arm. The view in the head-mounted display was as it would be seen by the astronaut working around the Hubble berthed in the shuttle payload bay at an orbital altitude of 531 km (330 miles) above the Earth.

The next opportunity to take advantage of the virtual reality software involved EVA crew members training to perform the first engineering test flights of the



Astronaut Mark Lee trains for his Simplified Aid for EVA Rescue test flight (STS-64 [1994]) using the virtual reality flight trainer (left) and on orbit (right).

Simplified Aid for EVA Rescue (SAFER) on STS-64 (1994).

The output of a dynamic simulation of the SAFER backpack control system and its flying characteristics, using zero-gravity as a parameter, drove the head-mounted display visual graphics. Inputs to the simulation were made using a flight-equivalent engineering unit hand controller. The EVA crew member practiced and refined the flight test maneuvers to be flown during on-orbit tests of the rescue unit. The crew member could see the on-orbit configuration of the shuttle payload bay, the robotic arm, and the Earth/horizon through the virtual reality head-mounted display at the orbital altitude planned for the mission. The EVA crew member was also able to interact with the robotic arm operator as well as see the motions of the arm, which was an integral part of the on-orbit tests. The robotic arm operator was also able to view the EVA crew member's motions in the simulated shuttle payload bay camera views made available to the operator as part of the dynamic man-in-the-loop robotic arm simulation.

As a result of the engineering flights of the SAFER unit on STS-64, NASA was able to validate the virtual reality simulation and it became the ground-based SAFER training simulator used by all EVA crew members assigned to space station assembly missions.

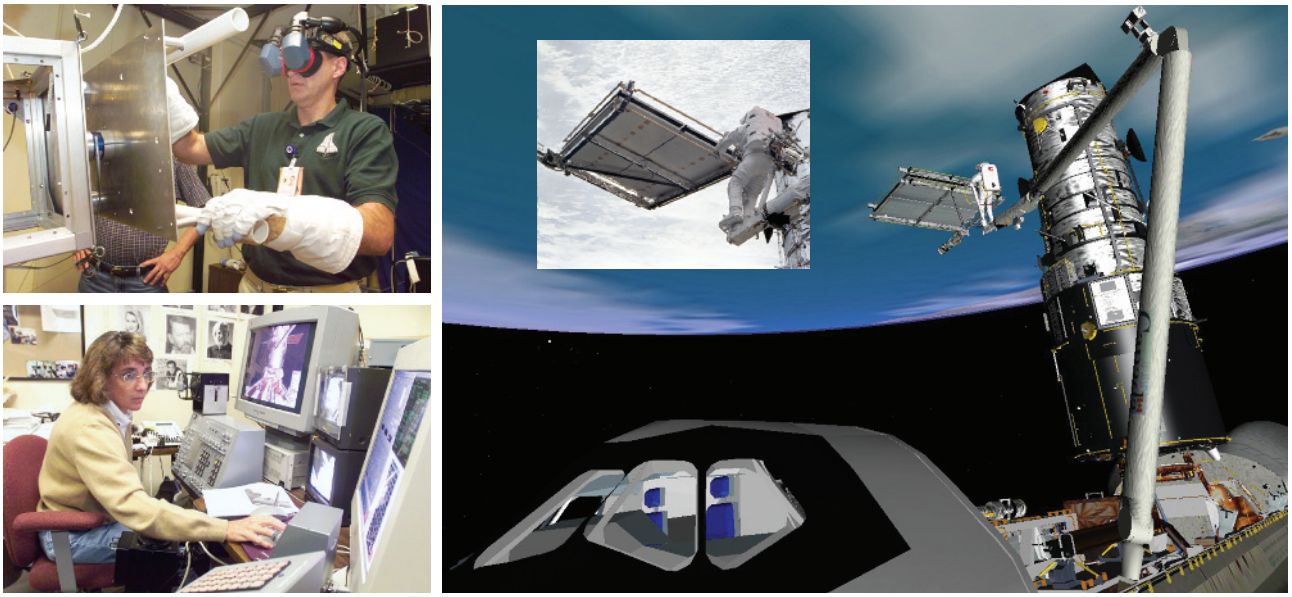
Each EVA crew member was required to have at least four 2-hour training classes prior to a flight to practice flying rescue scenarios with the unit in the event he or she became separated from the space vehicle during an EVA.

NASA also developed a trainer that was flown on board the space station laptop computers. The trainer used the same simulation and display software as the ground-based simulator, but it incorporated a flat-screen display instead of a head-mounted display. It also used the same graphic model database as the ground-based simulators. ISS crew members used the on-board trainer to maintain SAFER hand controller proficiency throughout their time on the ISS.

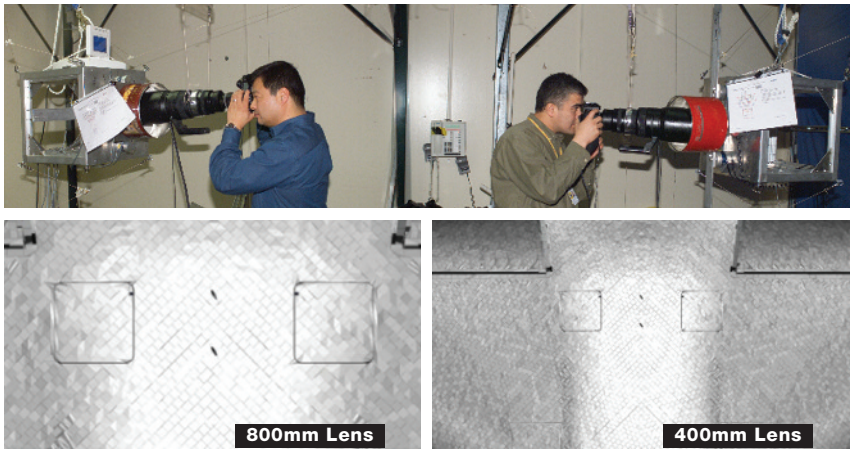
Handling Large Objects During Extravehicular Activity

Learning to handle large objects in the weightlessness of space also posed a unique problem for EVA crew members training in ground-based facilities. In the microgravity environment of space, objects may be weightless but they still have mass and inertia as well as a mass distribution around a center of gravity.

NASA engineers developed a tendon-driven robot and a set of dynamic control software to simulate the feel and motion of large objects being handled by an EVA crew member within the zero-gravity parameter. The basic concept was to mount a reel of cable and an electric drive motor at each of the eight corners of a structure that measured approximately 3 m (10 ft) on a side. Each cable was then attached to one of the eight corners of an approximately 0.6-m (2-ft) cube. In this configuration, the position and orientation of the smaller cube within the large structure could be controlled by reeling in and out the cables. Load cells were mounted to the smaller cube



Astronauts Richard Linnehan (above left) and Nancy Currie (below) use the zero-gravity mass handling simulation and the Shuttle Robotic Arm simulation to practice combined operations prior to flight. The large image on the right is a rendering of the simulation. The inset is an actual photo of Astronaut Richard Linnehan (STS-109 [2002]) unfolding a solar array while anchored to the end of the robotic arm.



International Space Station Expedition 10 crew members Leroy Chiao (left) and Salizhan Sharipov train in virtual reality to photograph an approaching Orbiter through the space station windows. The lower pictures show what each sees through his respective camera view finder.

while handrails or other handling devices were attached to the load cells. As a crew member applied force to the handling device, the load cells measured the force and fed those values to a dynamic simulation that had the mass characteristics of the object being handled as though it were in weightlessness. Output from the computer program then drove the eight motors to move the smaller cube accordingly. Once these elements were integrated into graphics in the head-mounted display, the crew member not only felt the resulting six-degree-of-freedom motion of the simulated object, he or she also saw a three-dimensional (3-D) graphical representation of the real-world object in its actual surrounding environment.

The mass handling simulation—called kinesthetic application of mechanical force reflection—was qualitatively validated over a number of shuttle flights starting with STS-63 (1995). On that flight, EVA crew members were scheduled to handle objects that weighed from 318 to 1,361 kg (700 to 3,000 pounds) during an EVA. After their flight, they evaluated the ability of the application to simulate the handling conditions experienced in microgravity.

Kinesthetic application of mechanical force reflection was deemed able to faithfully produce an accurate simulation of the feel of large objects being handled by EVA crew members following a number of postflight evaluations.

Kinesthetic application of mechanical force reflection was also integrated with the Shuttle Robotic Arm simulation, which allowed the EVA crew member riding on the end of the arm to actually feel the arm-induced motion in a large payload that he or she would be holding during a construction or repair operation around the ISS or Hubble.

NASA built two kinesthetic application of mechanical force reflections so that two EVA crew members could train to handle the same large object from two different vantage points. The forces and motion input by one crew member were felt and seen by the other crew member. This capability allowed crew members to evaluate mass handling techniques preflight. It also allowed them to work out not only the command protocol they planned to use, but also which crew member would be controlling the object and which would be stabilizing the object during the EVA.

Virtual Reality Simulates On-orbit Conditions

Following the Columbia accident in 2003, as a shuttle approached the space station, space station crew members photographed its Thermal Protection System from a distance of 183 m (600 ft) using digital cameras with 400mm and 800mm telephoto lenses.

As in previous scenarios, there was no place on Earth where crew members could practice photographing a Space Shuttle doing a 360-degree pitch maneuver at a distance of 183 m (600 ft). Virtual reality was again used to realistically simulate the on-orbit conditions and provide ground-based training to all space station crew members prior to their extended stay in space.

Engineers placed a cathode ray tube display from a head-mounted display inside a mocked-up telephoto lens. The same 3-D graphic simulation that was used to support the previous applications drove the display in the telephoto lens to show a shuttle doing the pitch maneuver at a range of 183 m (600 ft). With a real camera body attached to the mocked-up lens, each crew member could practice photographing the shuttle during its approach maneuver.

Summary

NASA took advantage of the benefits that virtual reality had to offer. Beginning in 1992, the space agency used the technology at JSC to support integrated EVA/robotics training for all subsequent EVA flights, including SAFER engineering flights, Hubble repair/servicing missions, and the assembly and maintenance of the ISS. Each EVA crew member spent from 80 to 120 hours using virtual reality to train for work in space.

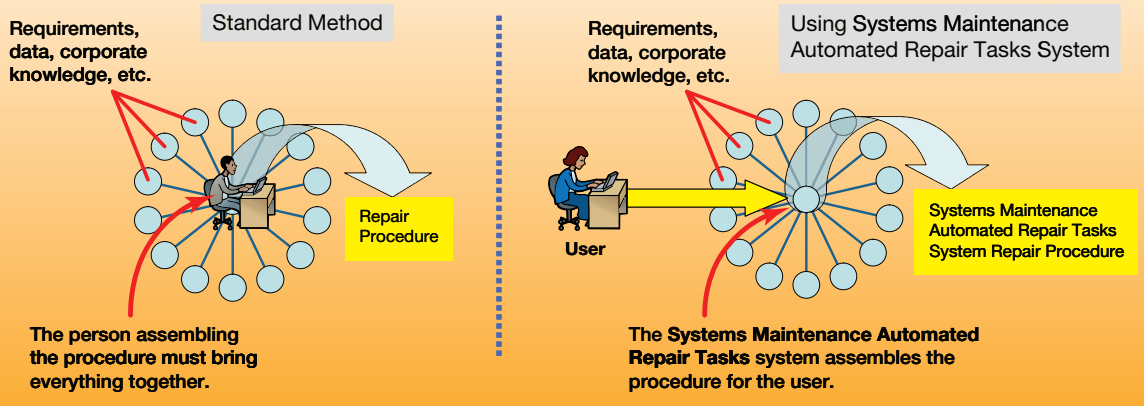
Integrated Solutions for Space Shuttle Management… and Future Endeavors

Kennedy Space Center (KSC) developed an integrated, wireless, and paperless computer-based system for management of the Space Shuttle and future space program products and processes. This capability was called Collaborative Integrated Processing Solutions. It used commercial off-the-shelf software products to provide an end-to-end integrated solution for requirements management, configuration management, supply chain planning, asset life cycle management, process engineering/process execution, and integrated data management. This system was accessible from stationary workstations and tablet computers using wireless networks.

Collaborative Integrated Processing Solutions leveraged the successful implementation of Solumina® (iBASEt, Foothill Ranch, California)—a manufacturing execution system that provided work instruction authorization, electronic approval, and paperless work execution. Solumina® provided real-time status updates to all users working on the same document. The system provided for electronic buy off of work instructions, electronic data collection, and embedded links to reference materials. The application included electronic change tracking and configuration management of work instructions. Automated controls provided constraints management, data validation, configuration, and reporting of consumption of parts and materials.

In addition, KSC developed an interactive decision analysis and refinement software system known as Systems Maintenance Automated Repair Tasks. This system used evaluation criteria for discrepant conditions to automatically populate a document/procedure with predefined steps for safe, effective, and efficient repair. It stored tacit (corporate) knowledge, merging hardware specification requirements with actual "how-to" repair methods, sequences, and required equipment. Although the system was developed for Space Shuttle applications, its interface is easily adaptable to any hardware that can be broken down by component, subcomponent, discrepancy, and repair.

Systems Maintenance Automated Repair Tasks Solution Philosophy—Variables



The Systems Maintenance Automated Repair Tasks allowed corporate knowledge to be kept in-house while increasing efficiency and lowering cost.

Three-Dimensional Graphics Provide Extraordinary Vantage Points

Astronauts' accomplishments in space seem effortless, yet they spent many hours on the ground training and preparing for missions.

Some of the earliest engineering concept development and training took place in the Johnson Space Center Virtual Reality Laboratory and involved the Dynamic Onboard Ubiquitous Graphics (DOUG) software package. NASA developed this three-dimensional (3-D) graphics-rendering package to support integrated training among the Shuttle Robotic Arm operators, the International Space Station (ISS) Robotic Arm operators, and the extravehicular activity (EVA) crew members. The package provided complete software and model database commonality among ground-based crew training simulators, ground-based EVA planning tools, on-board robotic situational awareness tools, on-board training simulations, and on-board EVA/robotic operations review tools for both Space Shuttle and ISS crews.

Level-of-detail Capability

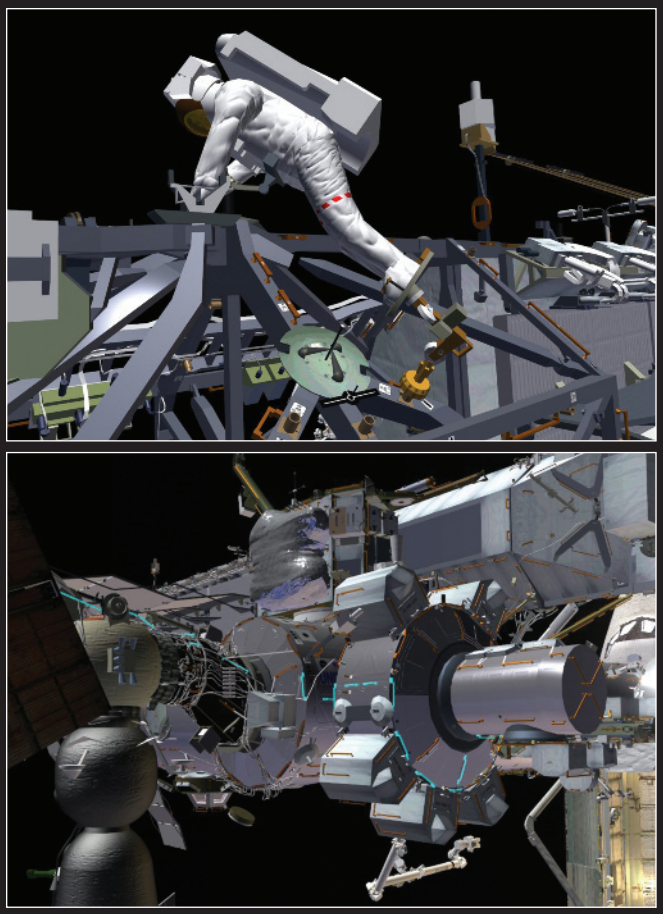
Originally, the software was written as an application programming interface—an interface that enables the software to interact with other software—around the graphics-rendering package developed to support the virtual reality

Additional Extravehicular Activity Support

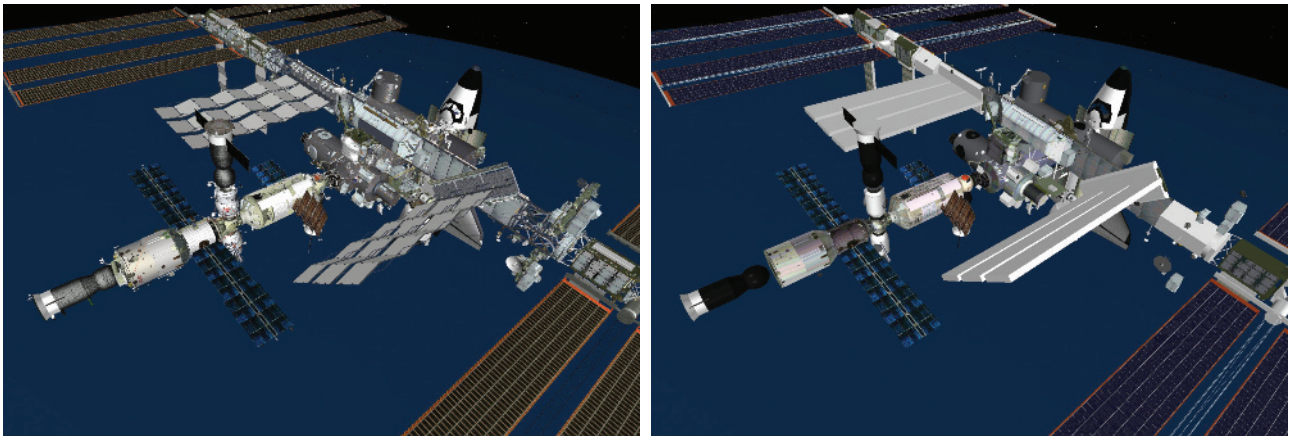
The International Space Station (ISS) has more than 2,300 handrails located on its exterior. These handrails provide translation paths for extravehicular activity (EVA) crew members. Pull-down menus in the Dynamic Onboard Ubiquitous Graphics (DOUG) software allow the user to highlight and locate each handrail. Entire translation paths can be highlighted and displayed for review by crew members prior to performing an EVA.

More than 620 work interface sockets are located on the external structure of the ISS, and nine articulating portable foot restraints can be relocated to any of the work interface sockets. Each articulating portable foot restraint has three articulating joints and a rotating base that produce 33,264 different orientations for an EVA crew member standing in that particular foot restraint. Each work interface socket can be located in the software package, and each articulating portable foot restraint can be configured to show all potential worksites and worksite configurations to support EVA planning.

The DOUG software package also contains and can highlight the locations of externally mounted orbital replacement units on the ISS, thruster and antenna keep-out zones that affect EVA crew member positioning, and articulating antennas, radiators, and solar arrays—all of which are configurable.



Articulated portable foot restraints configuration (top) and highlighted translation path (bottom).



These two views show the effect of level-of-detail control. The left view is a high-resolution image compared to the low-resolution image on the right.

training simulation. The Simplified Aid for EVA Rescue (SAFER) on-board trainer required software that would run on the original IBM 760 laptop computers on board the ISS and thus required the UNIX-based code to be ported to a Windows-based operating system. The limited graphics capability of those computers also required additional model database artifacts that provided level-of-detail manipulation to make the simulation adequate for its intended purpose. This additional level-of-detail capability allowed the same high-fidelity model database developed for EVA training in the virtual reality facility to be used on the laptop computers on the ISS.

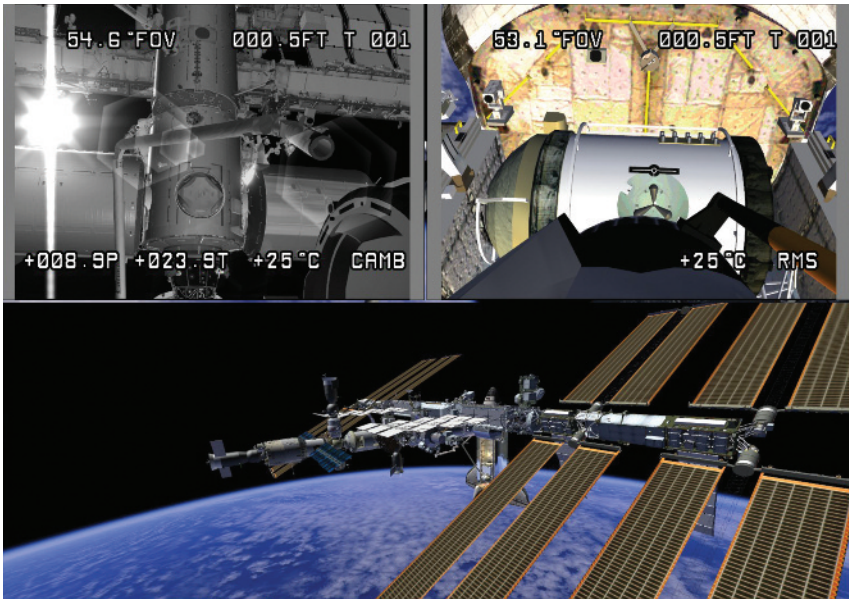
To obtain adequate graphics performance and screen update rates for simulating SAFER flying, crew members could select a low level-of-detail scene, which still displayed enough detail for the recognition of station landmarks and motion cues.

The DOUG software package, when not in use as a trainer, also provided a highly detailed, interactive 3-D model of the ISS that was viewable from any vantage point via keyboard inputs. The software first flew on board both shuttle and station in March 2001, and during Space Transportation System (STS)-102, and was on all subsequent shuttle and station flights with the exception of STS-107 (2003). That flight did not carry a robotic arm, had no planned EVAs, and did not dock with the ISS.

Benefits for Robotic Arm Operations

The DOUG software package supported SAFER training. The software was also capable of providing the situational awareness function during Space Station Robotic Arm operations by connecting to the on-board payload general support computer and using the telemetry from the arm to update the graphic representation in the program display.

The same software was compatible with laptop computers flown on the shuttle, and the graphical Shuttle Robotic Arm could be similarly driven with shuttle arm telemetry. Different viewpoints



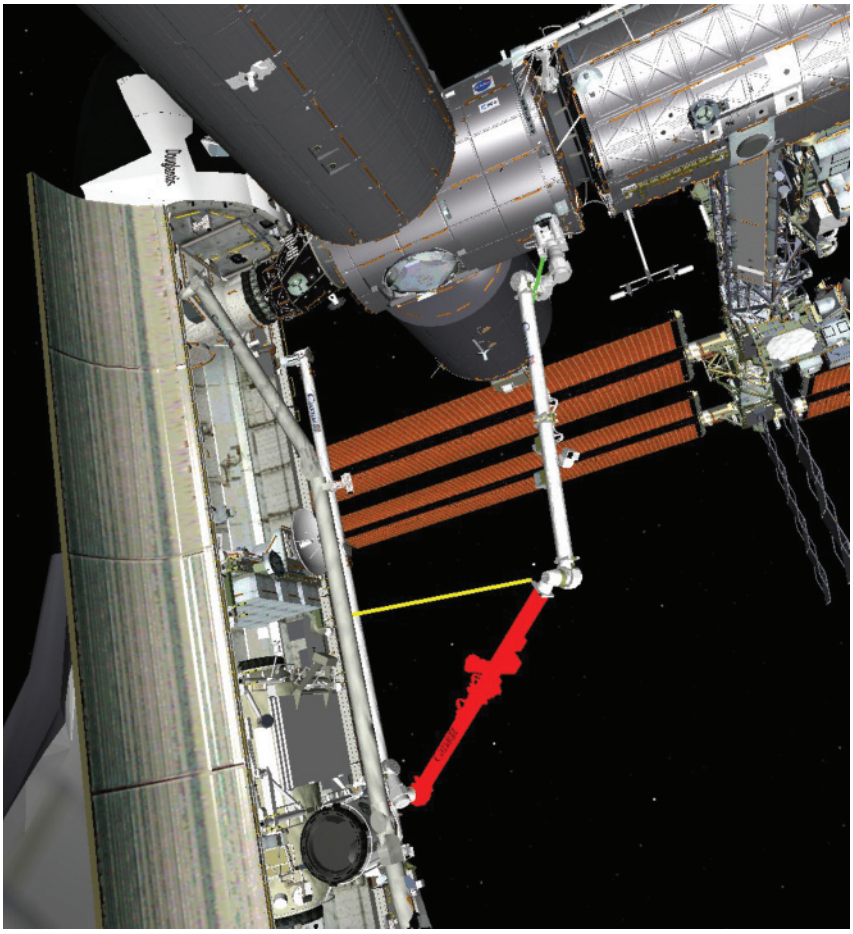
Dynamic Onboard Ubiquitous Graphics displays multiple simulated camera and synthetic eye-point views on the same screen. The simulated camera views show the Japanese Experiment Module and the Columbus Laboratory in the top left image, the Mini Research Module-1 in the top right image, and the International Space Station in the bottom image.

could be defined in the software to represent the locations of various television cameras located around station and shuttle. The various camera parameters were defined in the software to display the actual field of view, based on the pan and tilt capabilities as well as the zoom characteristics of each camera.

The second ISS crew (2001) used these initial capabilities to practice for upcoming station assembly tasks with the Space Station Robotic Arm prior to the actual components arriving on a shuttle flight. The crew accomplished this by operating the real robotic arm using the real hand controllers and configuring a "DOUG laptop" to receive remote manipulator joint angle telemetry.

The graphics contained the station configuration with the shuttle docked and the station airlock component located in the shuttle's payload bay. The arm operator could see synthetic end-effector camera views produced in the program. These views showed the airlock with its grapple fixture in the payload bay of the Orbiter even though no Orbiter actually existed. The operator practiced maneuvering the real arm end-effector onto an imaginary grapple fixture and then maneuvering the real arm with the imaginary airlock attached, through the prescribed trajectory to berth the imaginary airlock onto the real common berthing mechanism on the ISS Unity Node.

Through DOUG the arm operator also had access to synthetic views from all the shuttle cameras, as well as the Space Station Robotic Arm cameras that would be used during the actual assembly operations. This made training much more effective than simply driving the robotic arm around in open space.



The colors displayed in Dynamic Onboard Ubiquitous Graphics indicate direction of approach of the robotic arm booms with respect to the closest object: green = opening; yellow = closing; and red = envelope violation.

Proximity Detection

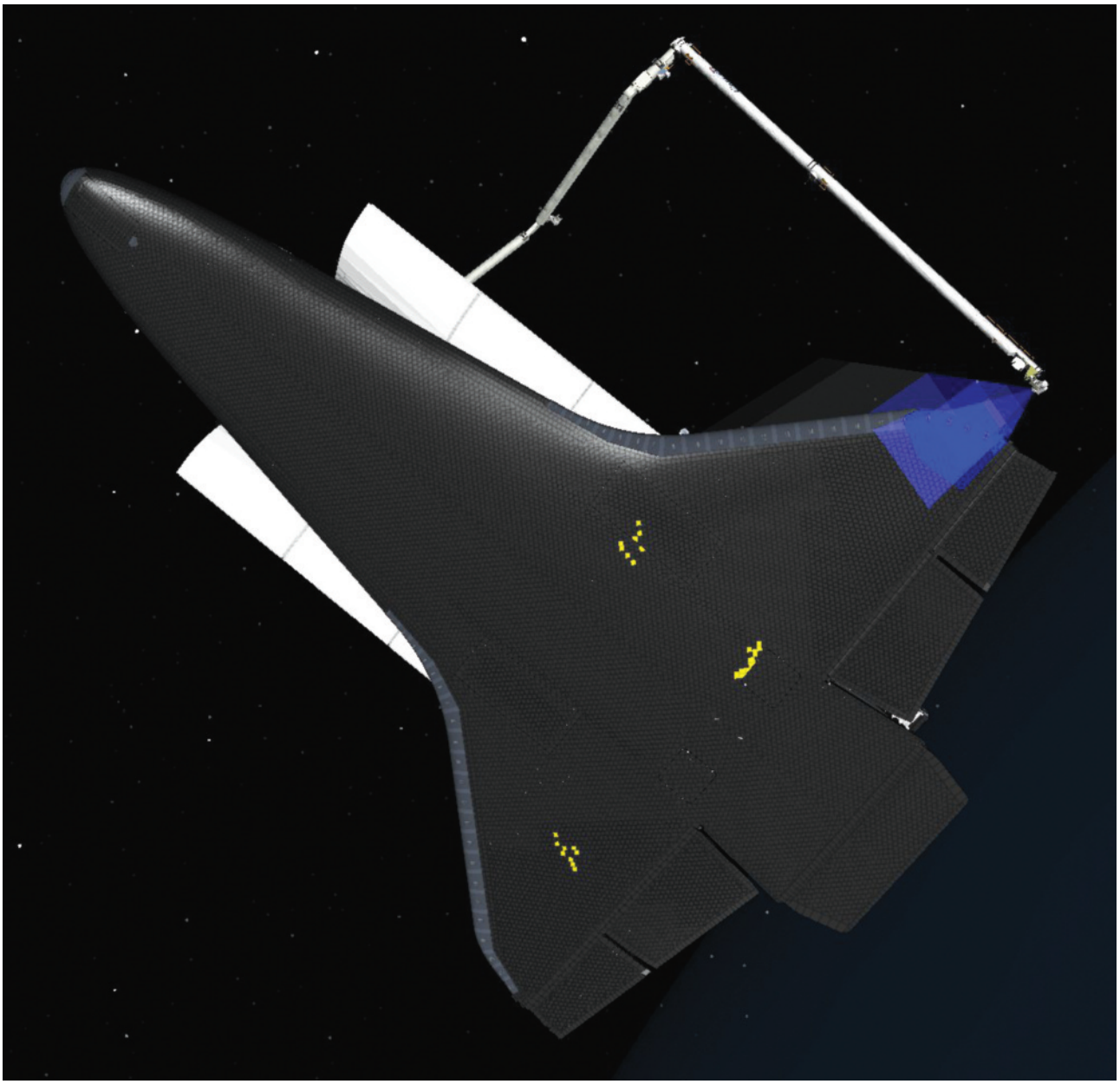
As the ISS grew in complexity, NASA added capabilities to the DOUG software. Following a near collision between the Space Station Robotic Arm and one of the antennas located on the laboratory module of the ISS, the space agency added the ability to detect objects close to one another—i.e., proximity detection. The software calculated and displayed the point of closest approach for the main robotic arm booms and the elbow joint to any station or shuttle component displayed in the model database.

A vector was drawn between each of the three robotic arm components and the nearest structure. When DOUG received robotic arm telemetry data and was being used for situational awareness during robotic arm operations, the color of these vectors indicated whether measured distance was increasing or decreasing. It also indicated whether the relative distance was within a user-defined, keep-out envelope around the robotic arm. Both audible and graphical warnings were selectable to indicate when a keep-out envelope was breached.

Thermal Protection System Evaluation

During the preparation for Return to Flight following the Columbia accident in 2003, NASA incorporated the entire shuttle Thermal Protection System database and a "painting" feature into the DOUG software package. The database consisted of all 25,000+ tiles, thermal blankets, reinforced carbon-carbon wing leading edge panels, and nose cap.

The software was used preflight to develop the trajectories of the Shuttle Robotic Arm and Orbiter Boom Sensor System used to perform in-flight Orbiter inspections. The software allowed engineers to "paint" the areas that were within the specifications



An example of the tile highlighting and painting feature in Dynamic Onboard Ubiquitous Graphics.

of various sensors on the Orbiter Boom Sensor System (e.g., range, field of view, incidence angle) to make sure the Thermal Protection System was completely covered during on-orbit surveys.

The same configuration models and tile database used on the ground were also loaded on the on-board laptop computers. This allowed the areas of interest found during the survey data analysis to be highlighted and uplinked to the shuttle and station crews for further review using the DOUG program.

Inspection of the STS-114 (2005) survey data showed protruding gap fillers between tiles on the Orbiter. These protrusions were of concern for re-entry into Earth's atmosphere. Ground controllers were able to highlight the surrounding tiles in the database, develop a Space Station Robotic Arm configuration with an EVA crew member in a foot restraint on the end, and uplink that configuration file to the station laptop computers. The crew members were then able to use the software to view the area of concern, understand how they would need to be positioned underneath the Orbiter, get a feel for the types of clearances they had with the structure around the robotic arm, and evaluate camera views that would be available during the operation.

Having the 3-D, interactive viewing capability allowed crew members to become comfortable with their understanding of the procedure in much less time than would have been required with just "words" from ground control. A key aspect to the success of this scenario was the software and configuration database commonality that DOUG provided to all participants—station and shuttle crews, ground analysis groups, procedure developers, mission controllers, and simulation facilities.

DOUG was loaded on more than 1,500 machines following the Columbia accident and was used as a tool to support preflight planning and procedures development as well as on-orbit reviews of all robotic and EVA operations. In addition to its basic capabilities, the software possessed many other features that made it a powerful planning and visualization tool.

Expansion of Capabilities

DOUG has also been repackaged into a more user-friendly application referred to as Engineering DOUG Graphics for Exploration (EDGE). This application is a collection of utilities, documentation, development tools, and visualization tools wrapped around the original renderer. DOUG is basically the kernel of the repackaged version, which includes the addition of various plug-ins, models, scripts, simulation interface code, graphical user interface add-ons, overlays, and development interfaces to create a visualization package. The project allows groups to quickly visualize their simulations in 3-D and provides common visuals for future program cockpits and training facilities. It also allows customers to expand the capabilities of the original software package while being able to leverage off the development and commonality achieved by that software in the Space Shuttle and ISS Programs.

EDGE is now publicly available. To request a copy, call or email:

Technology Transfer and Commercialization Office
NASA Johnson Space Center
Phone: 281-483-3809
Email: jsc-techtran@mail.nasa.gov

Summary

The graphics-rendering software developed by NASA to support astronaut training and engineering simulation visualization during the shuttle era provided the cornerstone for commonality among ground-based training facilities for both the Space Shuttle and the ISS. The software has evolved over the years to take advantage of ever-advancing computer graphics technology to keep NASA training simulators state of the art and to provide a valuable resource for future programs and missions.