FPAC: Future Prototype Assessment for Cockpit 2.0

Michael Scherer*, Sabrina Billinghurst‡, Heather Justice†, and Scott Davidoff§
NASA Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
Email: { michael.scherer*, sabrina.billinghurst‡, heather.justice†, scott.davidoff§ }@jpl.nasa.gov

***Abstract*—There is a steep learning curve for sequencers and mission planners to create mission plans for landing and driving robotic vehicles on current text-driven interfaces. Touch screen interfaces are quickly growing in popularity as a means to effectively and intuitively interact with computers and robotic systems. Future Prototype Assessment for Cockpit 2.0 (FPAC) is designed as a system for assessing the effectiveness of various touch-based paradigms for mission planning and sequence development. Operations for approach, landing and investigation of near-earth objects (NEOs) with the ATHLETE platform are chosen as the use case for this system. Scientists, engineers, and other mission-planning roles are the primary audience for FPAC. Usability and training minimalization are the primary metrics for this system, in order to optimize the workflow of sequencers and mission planners. Flexible touch-based camera controls, interactive timeline, sequence comparison options, and 3DConnexion SpaceBall integration are all included features in the assessment package.**

I. Introduction

Touch-based interfaces have been growing as a means to control and operate computers. Such interfaces are more intuitive, reducing the training time for new users [1]. These traits make touch interfaces desirable as a new means for planning, sequencing, and controlling remotely operated vehicles. This system was designed to operate on the Samsung SUR40/Microsoft PixelSense platform, formerly referred to as the Microsoft Surface 2.0. Additionally, the 3dConnexion SpaceBall was integrated. The experimental setup can be found in Figure 1. With these tools and innovative interface design, we hope to evaluate technologies to benefit and streamline the planning and sequencing operations of future NASA spacecraft and robots, which is seen as a growing challenge with rapidly advancing technologies [2].

A. Background

Current rover planning operations are completed with a variety of software packages that are generally mission specific. In order to collaborate, plan, and sequence a mission or drive, many tools are used even within a given project. For the Mars Exploration Rovers (MER) [3], planning and scheduling is done through a software package called MAPGEN [4], [5]. A newer software package, Ensemble, was created as the next iteration for planning mission operations for the Mars Phoenix Lander and the Mars Science Laboratory [6]. Ensemble is framed within the Eclipse IDE as a collection of plugins to be interoperable with different missions.

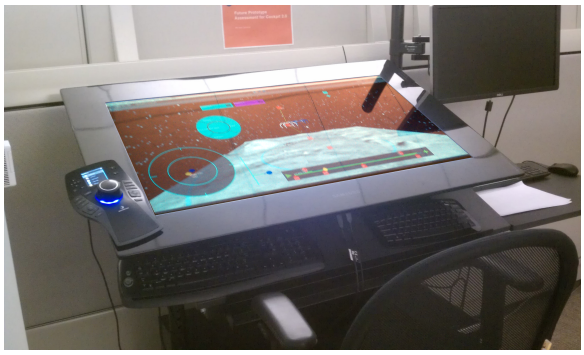

Fig. 1. Experimental setup

II. Objectives

The primary objective is to create a set of controls and a prototype interface for the evaluation of large touch interfaces as a means of creating sequences and mission plans for the ATHLETE [7] vehicle to land on, travel across, and interact with the surface of a Near-Earth Object (NEO). Using this prototype, user needs can be assessed in tandem with tests of the usability of various touch-based controls and interaction schemes.

III. System Design

Low fidelity prototypes were created initially to get a feel for how the interface might be presented. This was accomplished with photos taken of a white board that a user was ”interacting” with. Afterwards, the photos

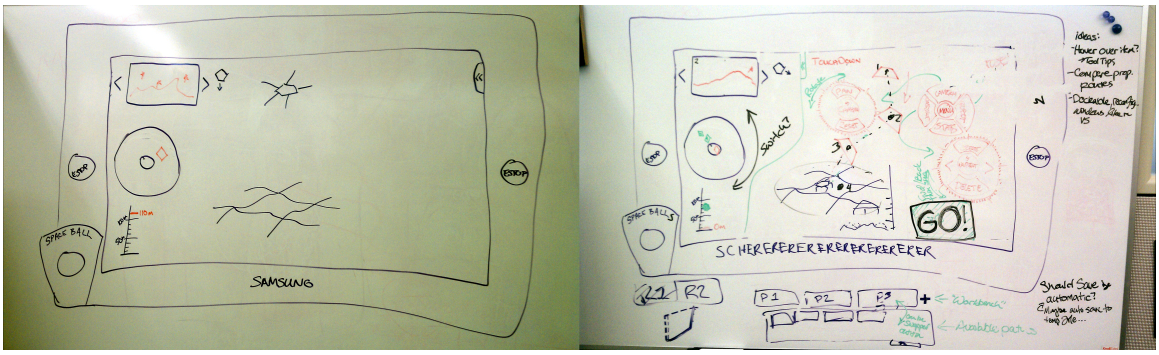

Fig. 2. Low-fidelity prototypes, before (left), and after (right)

were looked at in sequence to give an impression for the interface by members of the research group to sort out and get an understanding of technical requirements. Over time, these prototypes evolved to help establish a better understanding of the system being designed, as shown in Figure 2.

The system is split into two major components, the Input Server and the Client User Interface (UI), as shown in Figure 3. Communication between the two interfaces occurs over named pipes with a custom messaging scheme. This separation was created originally due to technical limitations of using Unity [8] as the game engine for the UI. However, the decoupling has a side effect of improving the modularity of the components, so that future work with the Surface and SpaceBall can be implemented much more quickly.

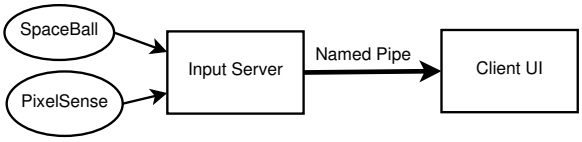

Fig. 3. System overview

A. Input Server

The input server asynchronously accepts data from the SpaceBall and touch points from the Microsoft Surface SDK, and then relays that information to the Client UI. The information is relayed over a Named Pipe. The Named Pipe was chosen because of its simplicity and reliability for interprocess communication on the same workstation [9].

B. Message Format

Messages have a fixed width of 12 bytes. The first byte is the message code, followed by the payload, followed by the tag. The tag is a single byte that is incremented with each sent message. This is not to maintain order, as named pipes transmit all information in order, however it is to ensure that duplicate messages are not received, which is the case when a connection becomes interrupted. The message length is always padded to the full 12 bytes.

1) HeartBeatMessage (0x00): The HeartBeatMessage is the simplest of the messages, with a message code of 0x00. It does not contain any payload, and is sent periodically if no other input is being transmitted so that the connection stays alive.

2) TouchPointMessage (0x01): Touch point messages contain the x and y coordinates, stored as two 4-byte IEEE 754 floating-point values. It is followed by a 4-byte signed integer value representing the id of the touch point. Each touch point id is unique and allows the user to see the changes in a single tracked point over time. A single-byte operation code is also present, which indicates whether the point is new, existing, or a point which has just been removed. If the point has just been removed then the x and y values have no meaning.

3) SpaceBallMessage (0x02): The last of the message types, the SpaceBallMessage contains the x-y-z translation or the yaw-pitch-roll rotation of the SpaceBall as 3 floating point values. An opcode is given to determine whether the message contains translation or rotation values.

C. Client UI

The Client UI is written in C# in the Unity3d development environment. Unity was chosen for its ease of use in rapidly prototyping new graphic software. The SUR40 is essentially a large, 40” multi-touch capable screen with an integrated computer. The screen has a resolution of 1920x1080. The UI allows for the user to create, edit, and delete waypoints that the ATHLETE robot may drive to, along with adding leg poses, scrubbing through

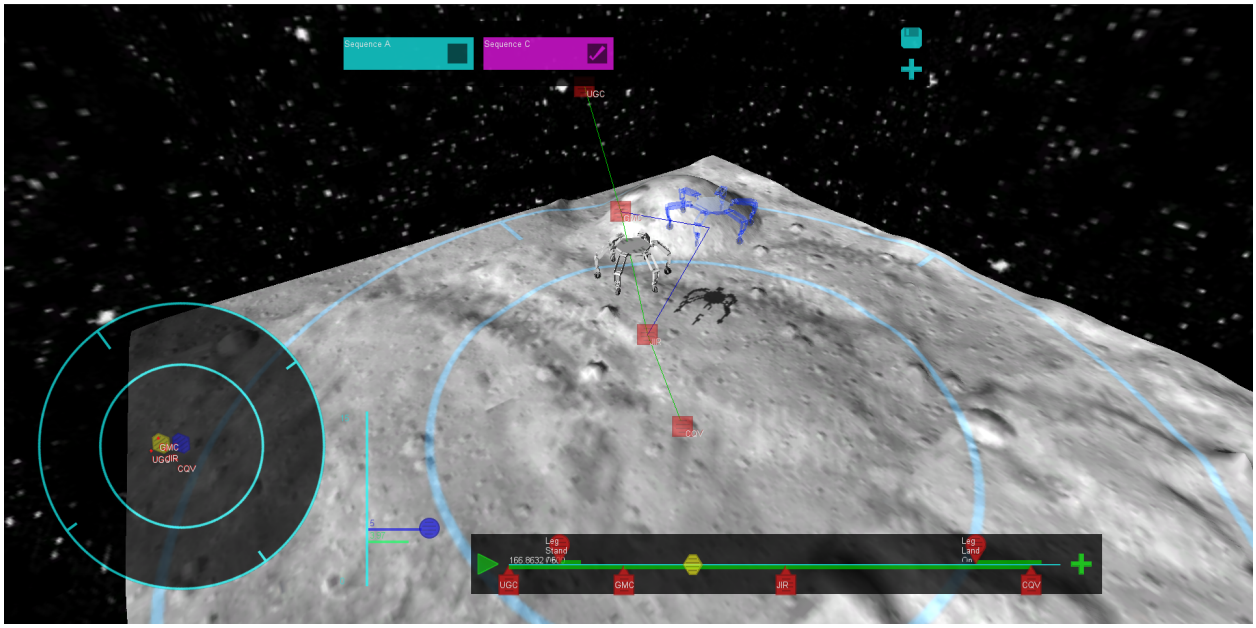

Fig. 4. GUI overview

the sequence on a timeline, and finally save and load collections of sequences.

The workspace supports various multi-touch gestures. A dragging gesture rotates the view about the vertical axis or the horizontal axis by dragging left/right or up/down, respectively. A pinch gesture is used for zooming. Waypoints and the cursor can also be dragged on the screen. When dragged, their motion is constrained to the 2D coordinate space of the screen in the 3D scene.

1) Top Down View: This control constrains the viewing space of the environment to the surface plane. It shows the vehicle location, cursor location, and waypoint locations. The view rotates such that the top matches the heading of the main workspace view. From this view, the user can drag the cursor along the plane with one finger to get constrained motions along x-y.

2) Elevation Control: The elevation control shows the current vehicle altitude along with the cursor altitude. The cursor altitude (z) can be changed by dragging the handle up and down with one finger.

3) Timeline: The timeline control visualizes waypoints and robot actions that occur throughout the simulation as events. It allows the user to play back, and scrub through, a simulation of the ATHLETE robot. Waypoints are indicated as square icons below the timeline bar, while robot actions are indicated as circular icons above the bar. All waypoints and robot actions may be dragged independently of one another back and forth across the timeline. Additionally, they may be deleted by dragging them up off of the timeline bar. A red "X" appears beneath the finger of the user for a delete gesture as feedforward to indicate what the system sees as the user's intent. Feedforward is used because it has been shown to improve users' learning of the capabilities of a system [10].

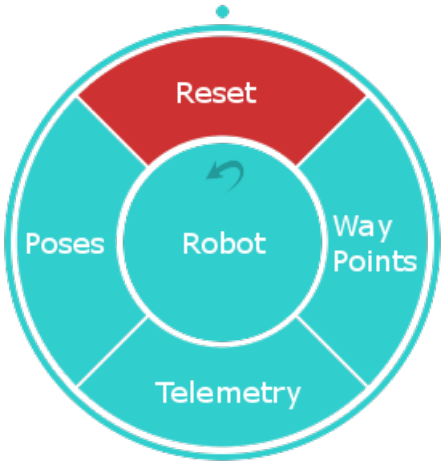

Fig. 5. Example touch menu

4) Touch Menus: The touch menus are versatile hierarchical circular menus for accessing additional functionality on the system. They are movable in the workspace, and extendable in code. The circular form factor of the menu was chosen because of its high end-user performance [11]. A blue color was chosen to indicate

menu items which traverse down the hierarchy, while a red color was chosen to indicate menu items which perform an action.

To create a touch menu, the user presses his or her finger on an empty spot in the workspace. While he or she presses, a circle is drawn around the finger to indicate that a menu create gesture is being performed. Once the circle closes, then the menu will appear centered in the location of the users finger. In order to traverse back up the hierarchy, the center circle can be pressed. There is also an optional dial which surrounds the menu which may be rotated. Values generated by this dial are accessible to all the active menu items. An example of a touch menu is shown in Figure 5.

5) Sequence Bar: Sequences can be created, selected, deleted, and saved with the sequence bar. Selecting a sequence is done with a touch gesture. Sequences can be rearranged by dragging them to the left or to the right. Dragging a sequence down off of the bar will delete the sequence, with feedforward in the same manner as in the timeline waypoint deletion gesture. Sequences may be added by pressing the "plus" button, or saved by pressing the "save" icon, which takes the form of a floppy disk. Adjacent to the name of the sequence is a check box, which may be selected or deselected through a touch gesture for comparing the sequences. When a sequence is selected for comparison in this way, its path appears as a yellow trajectory in the main workspace so that it can be visualized against the current active sequence.

6) Space Ball: The SpaceBall, also sometimes referred to as a "3D Mouse", allows for full 6 degree of freedom control by translating or twisting the control knob. The space ball serves as a supplemental input method for FPAC. The primary advantage a space ball has is its ability to take full six degree-of-freedom input from the user. In this implementation, cursor movements and free-form camera control can be performed with the space ball.

IV. Test Approach

The chosen participants for the study are primarily rover planners and drivers for MER and Mars Science Laboratory (MSL) [12]. Participants are first given a pre-survey to collect demographic information, and then introduced to the context of the system. Participants are asked to use the talk aloud protocol, which is to have the participant speak their current thoughts and intentions while using the system. A "training" session is then done with the participant, where different controls are indicated to on the screen by the proctor, who then asks what the participant thinks the functions of said control are. The participant is then asked to attempt to perform the actions they expect, and report whether or not the outcome of such actions were expected.

Next, participants are asked to complete three tasks:
1. a. Create a sequence that follows the green line, such that the three waypoints lie in the beginning, middle, and end of the timeline, respectively. All lines should be green.
1. b. Make the robot go into a landing position which completes at the last waypoint.
2. Create a new sequence where the robot starts off not moving on the surface, then stands, and then drives across the surface. All lines should be green.

Finally, the participants are asked to complete a post-survey where they list three things they liked about the interface, three things they didn't like, and lastly, whether or not they could see the interface as being used for rover planning.

V. Results

The cursor appears to be one of the largest sources of confusion for the users when starting with the system. After explanation of the cursor's function, users are able to complete the assigned tasks with minimal problems. The users appear to want to manipulate the simulation model of ATHLETE directly rather than making use of the cursor.

The users also express a desire to have the ability to copy and paste sequences and waypoints. Additionally, they want the ability to select multiple objects which can be manipulated or deleted as a group.

There was only one frame of reference given with the spaceball. Users state in some situations, they want a local frame of reference, and in other situations, they want a global frame of reference. The users become frustrated when using the spaceball at times because they inadvertently manipulate more axis than intended, e.g. rotating the cursor while they only want to translate the cursor.

The camera view at times is disorienting to users because it tracked the simulated vehicle (a "chase cam"). When shown the static/free fly camera, they greatly prefer it. Users indicate that there are times when they like the "chase cam", however they mostly work using a static camera. Additionally, users want a way to reset the camera to pre-set views, such as a top down view where the top of the camera has a heading of 0° (North).

VI. Discussion and Future Work

Because the cursor was the source of much of the initial confusion, there is evidence to suggest it should be removed. Direct manipulation of the simulation model may serve as a replacement to the functionality of the cursor. Additionally, this could resolve issues surrounding the user's understanding of how to add waypoints, as they might be automatically created to match the manipulations made by the user.

Another important change may be to have the ability to "snap" the vehicle to the terrain surface, because many users desired such functionality. This idea may be extended such that the vehicle is always constrained to the terrain surface. This extension is inspired by a better understanding of the problem domain, which is less focused on entry, descent, and landing (EDL), and more focused on daily operations of driving across a terrain surface. When driving, users also wanted to be able to specify one or many "keep out" zones, which are areas that appear dangerous for vehicle operation, and thus should be avoided explicitly by any automated or commanded function of the rover.

There are many limits and bounds that the planners use to prevent existing rovers from performing dangerous maneuvers. Any future system would need to incorporate the ability to edit and view such parameters. It could be conjectured that such parameters would be best as a side menu that is not always visible, based on input from the users that much of the information is copied between missions and does not always need to be modified. There are, however, other, more frequently used parameters for specific functions, such as in the case of driving. For a given drive, important editable parameters include maximum drive speed, drive distance, heading, and drive mode. The drive modes in current systems, MER and MSL, includes an autonomous "AutoNav" mode [13].

VII. Conclusion

Recognizing FPAC as a prototype, the users overall seemed satisfied with the system as a starting point for future sequencing and planning operations. FPAC is far from being ready to replace existing systems, however valuable information on the needs of rover planners along with the usability of various touch-based controls was gained. This prototype and subsequent experiment was successful in attaining information which can be used to design future systems which will incorporate these technologies.

Acknowledgment

The authors would like to thank the entire Conductor leadership and team, including Jeff Norris and Victor Luo. This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the Jet Propulsion Laboratory Summer Internship Program and the National Aeronautics and Space Administration.

References

[1] A. Holzinger, "Finger instead of mouse: Touch screens as a means of enhancing universal access," *User Interfaces for All, LNCS 2615*, pp. 387–397, 2003.

[2] D. Billman, M. Feary, and J. R. Zumbado, "Evidence report: Risk of inadequate design of human and automation/robotic integration," August 2011.

[3] R. E. Arvidson, S. W. Squyres, R. C. Anderson, and J. F. B. III, "Overview of the spirit mars exploration rover mission to gusev crater: Landing site to backstay rock in the columbia hills," p. 22, 2006.

[4] M. Ai-Chang, J. Bresina, L. Charest, A. Chase, J. Cheng-jung Hsu, A. Jonsson, B. Kanefsky, P. Morris, K. Rajan, J. Yglesias, B. G. Chafin, W. C. Dias, and P. F. Maldague, "Mapgen: Mixed-initiative planning and scheduling for the mars exploration rover mission," *IEEE Intelligent Systems*, vol. 19(1), pp. 8–12, 2004.

[5] J. L. Bresina, A. K. Jonsson, P. H. Morris, and K. Rajan, "Activity planning for the mars exploration rovers," *15th International Conference on Automated Planning and Scheduling*, 2005.

[6] A. Aghevli, A. Bachmann, J. Bresina, and K. Greene, "Planning applications for three mars missions with ensemble."

[7] B. H. Wilcox, T. Litwin, J. Biesiadecki, J. Matthews, M. Heverly, J. Morrison, J. Townsend, N. Ahmad, A. Sirota, and B. Cooper, "Athlete: A cargo handling and manipulation robot for the moon," *Journal of Field Robotics*, vol. 24(5), pp. 421–434, May 2007.

[8] "Unity3d," 2012. [Online]. Available: www.unity3d.com

[9] Microsoft, "Interprocess communications." [Online]. Available: http://msdn.microsoft.com/en-us/library/windows/desktop/aa365574(v=vs.85).aspx

[10] D. Freeman, H. Benko, M. R. Morris, and D. Wigdor, "Shadowguides: visualizations for in-situ learning of multi-touch and whole-hand gestures," pp. 165–172, 2009.

[11] J. Callahan, D. Hopkins, M. Weiser, and B. Shneiderman, "An empirical comparison of pie vs. linear menus." [Online]. Available: http://www.donhopkins.com/drupal/node/100

[12] NASA Jet Propulsion Laboratory, "Mars curiosity rover," August 2012. [Online]. Available: http://www.jpl.nasa.gov/msl/

[13] D. M. Helmick, A. Angelova, M. Livianu, and L. H. Matthies, "Terrain adaptive navigation for mars rovers," *2007 IEEE Aerospace Conference*, pp. 1–11, March 2007.