HCII | NASA

Building an Interface for Robotic In-Situ Re-Tasking

Client
NASA Ames

Team Members
Marina Kobayashi
Shannon O'Brien
Guy Pyrzak
Christian Ratterman
Karen Wong
Greg Vassallo

HCII | NASA

Contents

HCII | NASA

Executive Summary

Executive Summary

Since the 1960s, NASA has focused on exploring mars. Initially sending flybys, NASA then moved on to launching orbiters and more recently has put efforts into developing landers and rovers. The success of the two rover missions has proved that the work and determination of NASA has paid off. As technology improves, the capabilities of the rovers and landers are greatly increasing and the activity schedules of spacecraft are becoming more complex and dynamic. Because of this increase in complexity, scientists and rover operators are requiring improved interaction with mission schedules. The challenge of facilitating the interaction between human operators and the robotic rovers and landers has been gaining importance.

In January of 2005, our team, consisting of six Masters of HCI students from Carnegie Mellon University, was formed. For our Masters capstone project, the Human Computer Interaction (HCI) group at NASA Ames Research Center presented our team with the task of designing and developing an interface to facilitate in-situ re-tasking of generalizable robotic platforms. The goal of this interface design is to address the new challenges of human-mission interaction.

As a way to test the functionality of our interface, the HCI group at Ames developed a game that was analogous to scientific data collection on Mars. The game has users acquire points by taking pictures of science targets. Each target provides points and a clue to the value of adjacent targets. The game board consists of an area of the floor with target blocks placed within a grid. The game pieces are two working robots: a Personal Exploration Rover (PER) and a Lego Mindstorm rover.

The team approached the problem by basing our research and core design ideas on space exploration as well as the game. The game provided simple activity plans and an easily understood context, which faciliated idea generation and the design process. The game also helped us produce simple tasks for our interface to support during usability studies.

At its core, the design supports all stages of the planning process within a unified system. This allows the user to build a plan from the ground up, becoming more detailed as he or she proceeds. The user starts with an initial strategic drive path using the map, then adds key activities such as photos within the map, and finally adds precise detail to each activity through the use of a unique graphical text editor. From pre-planning through data analysis, the interface supports the user in advanced and unique ways.

HCII | NASA

Design Themes

Mulitple Command Ability

Situational Awareness

Simultaneous Status Update

Correction for Uncertainty and Error

Design Themes

Based on our user research over the past 8 months of the Human Computer Interaction Capstone, we have selected 4 primary design themes to direct our new design.

> *Multiple Command Ability*

In-situ retasking's primary focus is giving new commands to a robot. The importance of commanding a robot was paramount in all of our contextual inquiries. However, the amount and complexity of the commanding varied. This data showed that the ability to command a rover on many levels was extremely important to robot operators. With this in mind, the design theme of multiple command abilities focuses on making sure that any single command can be given in many ways. It is the hope that this affordance will ensure any type of user, from the advanced roboticist to the scientist, is equally able to retask a robot. Design Aspects: 2, 3, 7, 8, 10, 13 16.

> *Situational Awareness*

In order to command a robot the user must understand the surroundings of the robot. Our user research shows that prior to commanding a robot a user would look at pictures, or if collocated, look at the robot itself before issuing commands. The user would also attempt to visualize the actions of the robot prior to sending commands. In-situ retasking asks the user to issue new commands based on new situational awareness and therefore, the design of the interface must allow for the new information to be quickly absorbed and understood by the user prior to issuing new commands. Design Aspects: 1, 4, 5, 6, 9, 15, 17, 18, 19, 20.

> *Simultaneous Status Update*

User data showed that the process of in-situ retasking has multiple cyclical steps, strategize, detailed tactical commands, receive new data, and then repeat. These steps are separate, however, they are also interrelated due to their cyclical nature. Each step affects both the previous and next step. Our design reflects this by showing the effects of one interface area on the others. This allows the robotic operator to better understand the consequences of his actions on multiple levels. Design Aspects 7, 10, 18, 17.

> *Correction for Uncertainty and Error*

Unfortunately, errors and uncertainty, in general and in robotics is unavoidable. This is especially important when the user is not co-located with the robot because they are not able to physically intervene if necessary. We also observed that users did not completely trust the situational awareness of the robot and often desired to see the situation with their own eyes, this was due to the uncertainty associated with the robot's perception of its surroundings. Our design reflects this by allowing the user to correct the robot during communication points. The ability to correct and adjust for uncertainty gives the user the assurance that even if the robot makes a mistake the robotic operator will be able to make appropriate adjustments, thereby lowering the anxiety caused by the possibility of errors or uncertainty. Design Aspects 1, 6, 7, 12, 14.

HCII | NASA

Task Scenario

Create a Plan

Adjust Camera Angle

Check Time until Downlink

View Data

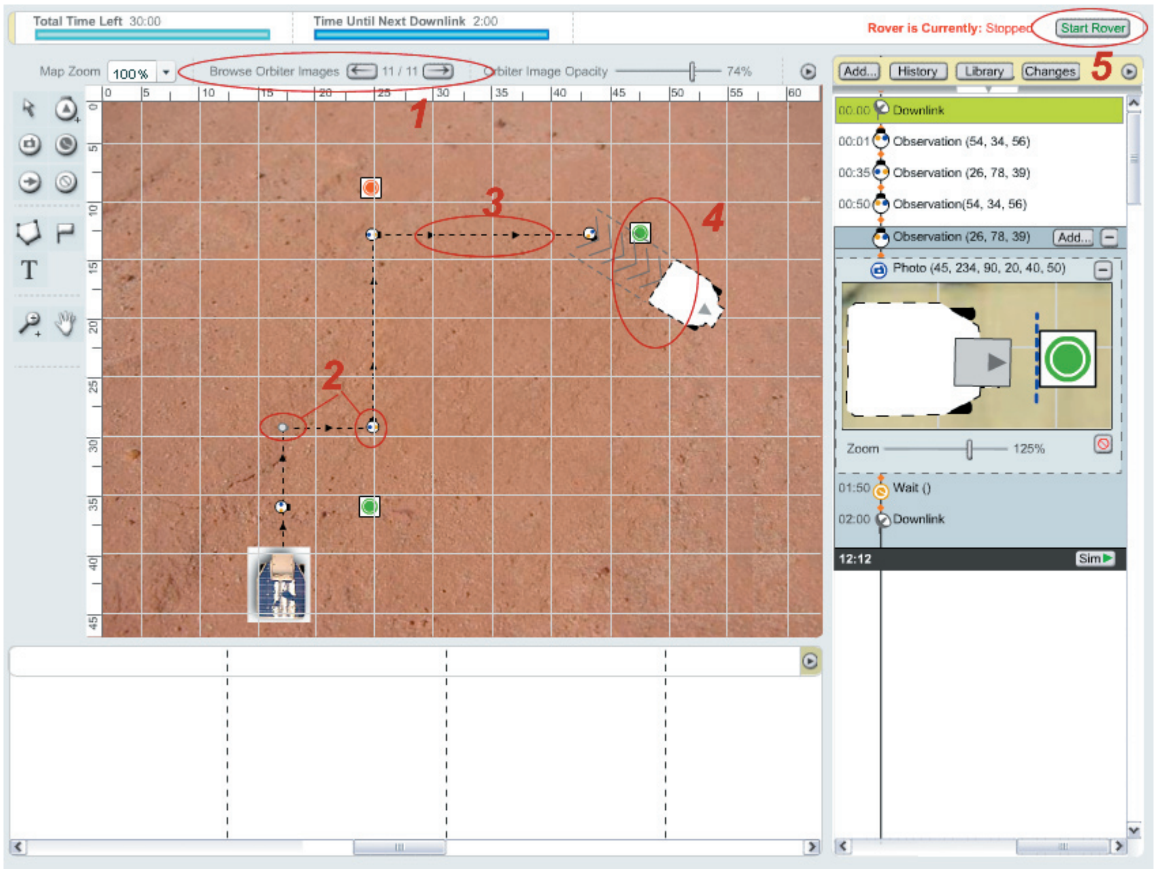
Task Scenario Description

Dr. Jane D. is a scientist at the Jet Propulsion Laboratory currently working on a Mars exploration mission. She is responsible for controlling one of the many rovers involved in the mission. Jane's background is in marine biology and physics, and her goal for the rover is to discover more deposits of hematite in rocks, which will strongly support her hypothesis of a lake once existing in one of the large craters on mars.

It is sol 115, and Jane has just settled down at work. She is about to start controlling her rover, which she dubbed H2Rover. Jane has had the responsibility of controlling H2Rover since the beginning of the mission. Jane is very comfortable using graphical user interfaces, but only has some knowledge of using command line application control.

Task 1: Create Initial Plan

Jane looks over the orbiter image and starts mapping out where she wants H2Rover to go. Since yesterday, her goal has been to visit three of the rocks lying to the right of the rover. She places waypoints on the map to specify where she wants H2Rover to go. By using the camera tool, she adds 'take picture commands' to the waypoints next to the targets. When she is done creating this initial plan, she clicks the 'start rover' button to begin execution.


Total Time Left 30:00
Time Until Next Downlink 2:00
Rover is Currently: Stopped
Start Rover
Map Zoom 100%
Browse Orbiter Images 11 / 11
Orbiter Image Opacity 74%
Add...
History
Library
Changes
1
2
3
4
5
Zoom 125%
12:12
Sim

00:00 Downlink
00:01 Observation (54, 34, 56)
00:35 Observation (26, 78, 39)
00:50 Observation(54, 34, 56)
Observation (26, 78, 39)
Photo (45, 234, 90, 20, 40, 50)
01:50 Wait ()
02:00 Downlink

Design Aspects

1 *Ability to see orbiter image overlaid with visualizations*

In our design we overlaid the computers model of the plan and what its understanding of the rovers status was on top of an actual orbiter image taken at the last downlink time. By doing this the user is able to cross check their understanding of the situation. They are also able to better plan out their actions as they can work within the physical space of the terrain and avoid the hazards which might come up. At the downlink point, a new orbiter image comes in and automatically updates the screen. If the user would like to see previous orbiter images, he or she can select the image by using the arrows at the top of the screen (1).

Data

One of the important things that came out of our research was the importance of both the raw image and the computer model. Each provides valuable data and either can be flawed.

1.1. CI-RS-U1 > Work Flow Model > Time Stamp 1:00
1.2. NASA-MER-Driver > Note 362
1.3. CI_TR_U1_Follow > Note 547

2 *Ability to command based on waypoints*

Within the context of Mars exploration the different activities which the rover is capable of completing center around coordinate positions. We needed a system that would allow a plan to be built up from a strategic to tactical level while also minimizing the complexity of integrating all of the tasks. The solution was to allow the user to draw out a map on the screen. With each click they could place a waypoint down. The image above shows the two styles of waypoints: one with no sub activities and the other with two types of activities occurring. Each waypoint automatically sets the drive and turn information for the robot. By focusing this strategic level of tasking around the selection of locations, the user is able to avoid the hassle and confusion that comes with correlating his or her perspective, the rover's perspective and the robots surroundings for each start location. Although the robots themselves take commands in regards to duration of drives and direction of turns (right or left) the user is able to simply provide start and end locations and have the backend software provide the commands to navigate the path.

Data

2.1. CI_TR_U1_Follow > Note 523
2.2. TA on Paper Prototype Spring

3 *Waypoint Representation*

We originally had the direction arrows inside the waypoint icons. But when there was a need to place a Reverse waypoint (telling rover to go in reverse), the arrow was pointing in a different direction as the nose, and users were really confused by this. (2 – Both are forward drives) So we changed the waypoints to just represent the 'nose' of the rover, and have the direction it's going be represented on the drive path through the user of arrows along the path. The nose of the waypoint icon shows the rover will be facing in that direction when it pulls away from that location.

Data

3.1. TA at beginning of semester

4 *Separation between yet to run and already executed*

The path that the rover already drove is displayed in a solid black line, where as the future path is displayed in a dotted line.

Data

In our visit and observation at JPL, the rover drivers of MER told us that often between communication downlinks they were uncertain of which activities the rover had completed and which were yet to be executed. In the operating room contextual inquiry this was also a major finding that once the agent is tasked and acting in the environment, the OR manager cannot tell exactly which activities are yet to run and which are already executed, so this person must track down the information. In Trestle's robotic construction task it also came up that it was hard to tell how far along the robot was in its task versus its commands that it was executing.

4.1. JPL - Interviewh
4.2. MDRS – Day 2 observations
4.3. Trestle – Artifact Model, UI
4.4. OR – Interview

5 *Currently selected rover path is as wide as the rover*

Data

In our initial design all the drive paths for the rover were really lines. But a HE revealed that this could result in the user unknowingly running into or over objects in its environment by clipping them on the way to a target. (4 – An example of the interface showing a damaging drive). We changed the 'selected' representations of the rover icon and its path to be representative of the rover's actual size.

5.1. Heuristic Eval
5.2. TA_GS_U1

6 *E-Stop easily accessible*

The same button that is used to start the rover is used to stop it in an emergency. (5 - The button displayed in its start state). The button switches modes depending on the current state of the rover.

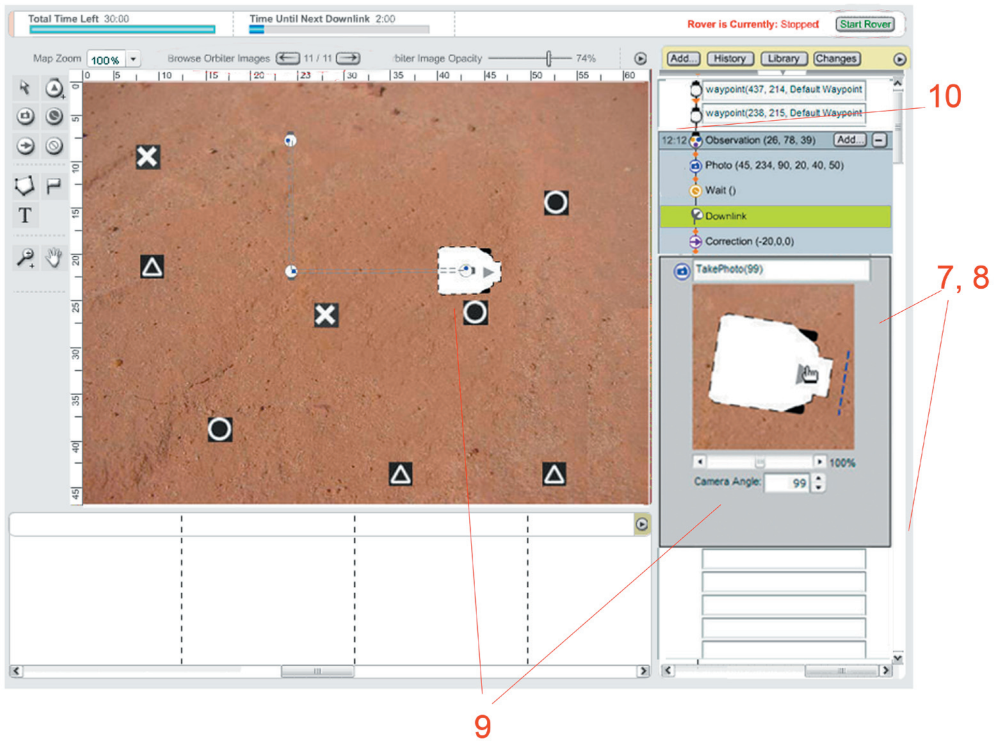
Data

Many of our CI's showed a need for a way to stop the rover without processing any further commands. This was valuable in both collocated and distant commanding. One question that comes up is if it is necessary in the Mars context since data transmission takes so long. The emergency stop is still important without real-time knowledge of the rovers location. With this system the user is aware of an estimated rover location. If the user has the running through a variety of activities and then realizes that they forgot to cover an area that they will soon pass by, it might be worth it to them to sacrifice further actions within the current commanding cycle to allow them to change their path.

6.1. MDRS > Breakdown: E-Stop needs to be quickly accessible.
6.2. Importance of e-stop also articulated in lit review and other robotic platform investigations, including Automated Turf Management, Robo Soccer, etc.

Task 2: Adjust Camera Angle

Jane sees that H2Rover has just left the second target and is approaching the third target. She wants to make sure that the science camera is aimed to take a clear picture of the third target. She makes detailed adjustments to the camera by using the activity planner. She finds the corresponding take photo activity, expands the view, and adjusts the angle.


Total Time Left 30:00
Time Until Next Downlink 2:00
Rover is Currently: Stopped
Start Rover
Map Zoom 100%
Browse Orbiter Images 11 / 11
Orbiter Image Opacity 74%
Add...
History
Library
Changes
waypoint(437, 214, Default Waypoint
waypoint(238, 215, Default Waypoint
12:12 Observation (26, 78, 39)
Photo (45, 234, 90, 20, 40, 50)
Wait ()
Downlink
Correction (-20,0,0)
TakePhoto(99)
100%
Camera Angle: 99
10
7, 8
9

Design Aspects

7 *Breakdown of Science Observations into Sub-Activities*

Data

The contextual inquiry with the robo-soccer team and tekkotsu provided us with data that showed operators of robots wanting to divide activities with greater granularity, or even to give the operator access to the most atomic level of activities within a grouping. This finding was also supported by the contextual inquiry at the Jet Propulsion Laboratory where members of the Mars Exploration Rover teams using the tool SAP (science activity planner) needed to be able to interact with the commands in a hierarchical breakdown.

"To me, it is very natural to look at big picture and break it down." – JPL MER TAP

8 *Enhanced visual representation of relationship between activities/sub-activities*

Data

In the think-aloud study we ran with our first interactive prototypes we found evidence that users required distinct and unambiguous visual representation on screen in order for them to understand the relationships of on-screen objects such as sub-activities that are grouped into an activity, or contingencies to a specific plan. Users had difficulty understanding how activities and sub-activities were related, so we modified the design to make it more obvious. We added a dark bar around the observation, a light box below that with the observation's sub-activities, and animation to have the sub-activities actually grow out of the observation.

9 *Tight coupling between the Strategic (Map) and the Tactical (Activity Planner)*

Data

Areas of the interface causes corresponding changes to occur in the map (strategic) area of the interface due to a tight coupling of these areas. Making a change to a specific sequence could affect the strategic science goals specified in another application, so our design is intended to help the user understand the relationship and implications between each. We also supported this in our design by allowing both graphical and code-based edits at the same time because of users at different levels being more comfortable and efficient with different interaction methods.

10 *Display Activity Start Times and/or Durations*

Data

The contextual inquiry at JPL gave evidence for the need to know start times and durations of each activity. This helps planners develop a schedule that will obtain maximum data during the given number of hours the rover will be in operation for that sol.

Extras

11 *Need for calibration*

Data

We saw that it was important to update the system's model of reality with what they called 'ground truth'. Robotic error WILL happen, and if you don't tell the system where the Rover actually is, the system won't know how to properly interpret your commands, or it will proceed with you commands in spite of them being inappropriate for the situation. This calibration would also include a 'Now' icon with the last known position based on the orbiter image, and where we next observed the rover and corrected the system's understanding of that position.

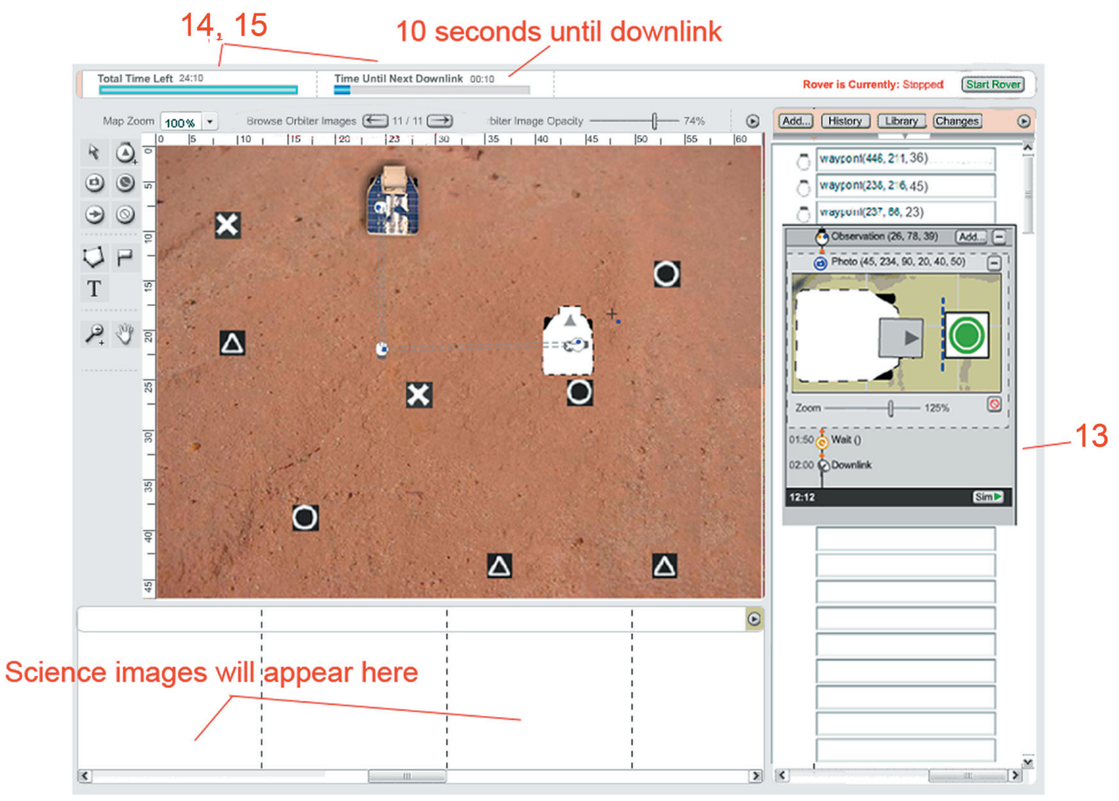
12 *Need for options*

Data

It was important for our interface to support options because a breakdown we observed at MDRS and in many of the other contexts was the fact that they weren't able to specify branching options ahead of time, thus they had to re-task 'in-situ' every time. Another issue that needed to be supported by optioning was that with more advanced robotic platforms they had ability specify on-board contingencies

Task 3: Check Time until Downlink

Jane knows she can only get data back from the rover every two minutes. She notices that there are only ten seconds left until downlink, and decides to wait until the new orbiter and science camera images appear before she continues planning.



Design Aspects

13 *More obvious representation of Downlink Point*

In the current design, downlinks are represented in the same way as other activities: there's an icon with a text description next to it.

Data

Participants in the critique at JPL indicated that they need a quick way to see where the downlink times occur; our current design may not be adequate.

14 *Status in Prominent Position*

The status bar is given a prominent and very visible position at the top of the screen.

Data

Evidence supporting this prominent position came from the CHI Tutorial and one of the contextual inquiries with a charge nurse at the OR. The tutorial urged that warnings can't go ignored. Furthermore, it stated that critical or urgent information has to be visible and prominent in order for it to be attended to. The charge nurse stressed that she needed to know the status of each doctor and nurse in order to quickly staff rooms during a trauma alert.

15 *Time Left and Time until Downlink Feedback*

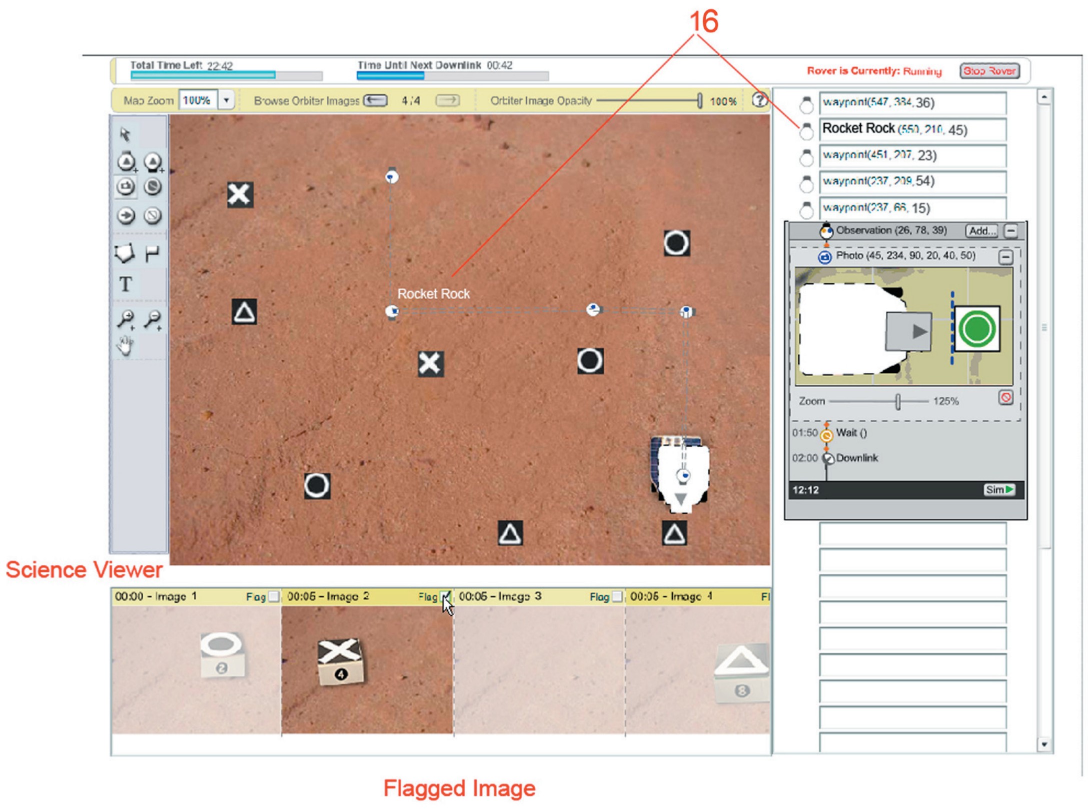
Located within the status area are two bars: one that provides feedback of how much time is left in the game, and the other that tells the user how much time before data from the rover is received (downlink time).

Data

This type of status is based on the crucial need for situational awareness (CHI Tutorial and literature review). In addition, one of the OR contextual inquiries gave evidence for the need to know how much time is left until the operation is completed. User 2 from the OR contextual inquiry exclaimed that she depends on the constant feedback from the doctors and nurses in each operating room in order to know their progress.

Task 4: View Data

To see the photos that the rover took, Jane references the science viewer. These photos help her make more informed decisions as to where the rover should go and whether or not she needs to re-task it. Based on the data collected, Jane decides to re-visit the second target and take more pictures of the other side of it. She flags the image.



Design Aspects

16 *Ability to Name Waypoints & Observations*

Data

We specified that in the next iteration of the prototype that the waypoints and observations could be named or labeled for easier searching and comprehension of where data came from. This was due to observations at MDRS and JPL showing that an enormous amount of time was spent to gain shared understanding or to map waypoints to what was visually represented. Without this labeling feature in the science data area of the science data a breakdown occurs when people try to refer to waypoints, targets, and observations that are not commonly and consistently named or labeled. At the OR as well, task cases were processed and tracked with names and other unique labels, so we determined that this was a good way to manage the data.

HCII | NASA

Conclusion

Conclusion

The design we have created is based on user data and iterativeanalysis. Our design allows the user to command therover on multiple levels through the map and activity planner. The interface affords for a high level of contextual awareness through the use of robotic plan overlay of the orbiter image in the map pane. It also assists with contextual awareness by using a zoomed-in and zoomed-out view of the orbiter image through the activity planner and map panes. The error correctionand rover calibration features which are both accessed through the map or activity planner enable the operator to quickly make corrections that will save time which could be better spent used in executing the plan.

While we have made great strides in our design, this is only the beginning of a larger exploration of a complex field. Future work on this design consists of implementing other features of our design, to allow for a more thorough user test. It is also important to continue testing the generalizability of the design by trying to use the interface with different types of rovers.

The design aspects highlighted in this report have been selected based on their viability in both the game context as well as the interplanetary context. It is our hope that these design aspects will be further examined and eventually implemented with interplanetary rovers.

Appendix

Table of Contents

1 > Team Members

Marina Kobayashi

- Carnegie Mellon University
    - Masters Program
    - Human-Computer Interaction Institute
- Carnegie Mellon University
    - BS, Policy & Management, Decision Science
    - Business Administration Minor
    - Skill Focus
    - Behavioral Science, CSCW, Large-Scale Collaboration

Marina grew up in Morgan Hill, California and originally came to Carnegie Mellon University for her undergraduate studies. As an undergraduate research assistant in the Human-Computer Interaction Institute's User Studies Lab she became interested in usability and how her background in behavioral science could be applied. Since that initial research assistantship she's become interested in looking at people working together and how multiple-person collaboration can be improved through innovative technology.

Marina has planned, designed, and conducted laboratory research to test multiple-person collaboration, schedule planning, and content management systems being developed for DARPA. The tests compared currently available applications with AI-enhanced applications, mocked-up interfaces, and multi-agent coordination, which involved several different components of an overarching intelligent agent system. Marina's NSF-funded research in the medical context involved field observations and interviews to assess the situation awareness and coordination, of people adapting to changing schedules, workloads, and resources. Her long-term research goals are to explore work coordination and planning in many complex and varied contexts, and to look at how human-machine interaction can extend the capabilities of technology that is not fully autonomous and vice-versa.

In her spare time Marina enjoys travel, movies, reading, and international cuisine.

Shannon O’Brien

- Carnegie Mellon University
    - Masters Program
    - Human-Computer Interaction Institute
- Carroll College
    - BA in Mathematics and Computer Science
- Work Experience
    - Two Internships at JPL
- Skill Focus
    - Programming (Java, C++), Writing

Shannon graduated in May 2004 from Carroll College, a private liberal arts college in Helena, Montana. She spent most of her life in rural Montana, and has had many unique experiences such as raising animals for 4H and attending a one-room school.

For the past two summers, Shannon had the opportunity to intern at JPL. During her first summer, she worked on automated sequence software, editing error messages. Last summer, she worked in the AI lab, coding an interface designed at Ames. Both internships have further led to her interest in the NASA program. In her spare time, Shannon enjoys basketball and outdoor activities such as running, hiking, and fishing.

Guy Pyrzak

- Carnegie Mellon University
    - Masters Program
    - Human-Computer Interaction Institute
- LSU
    - BS in CS
    - Chemistry Minor
- Work Experience
    - User Analyst for LSU's Office of Computing Services
- Skill Focus
    - Web Design/Programming, Application Programming
    - Document Design/Layout

Guy Pyrzak started playing with computers at a young age, and was quickly enamored with the Internet and its capabilities. Guy spent most of his younger years in Atlanta, Georgia, but moved to Mandeville, a small town north of New Orleans in high school. In high school, Guy became involved in web page design, and dynamic web application programming.

Guy graduated from Louisiana State University in May 2003 with a BS degree in Computer Science and Chemistry minor. After graduation from LSU, Guy took a position with the Information Technology Department at LSU and worked on several web applications. Guy's academic interests include ubiquitous computing, organizational communication, electronic collaboration, and game design.

In a former life Guy was an avid soccer player, scuba diver, skier and hiker. In his free time he enjoys watching Buffy on his TiVo, playing video games, and indulging in other geeky activities. Guy wanted to work on the NASA project because of his interest in translating complex and difficult tasks into a clear and simplistic interface. He hopes that in the future he will be able to apply his experiences from the NASA project to designing interfaces for complex ubiquitous devices.

Christian Ratterman

- Carnegie Mellon University
  - Masters Program
  - Human-Computer Interaction Institute
- Miami University of Ohio
  - Bachelor of Philosophy
  - School of Interdisciplinary Studies
  - Degree focus: Interactive Information Systems
  - Marketing Minor
- Skill Focus
  - Web Application Development, Interactive System Design & Media Development

Clearly focused on developing and managing various types of application systems, Christian Ratterman generated a personalized undergraduate degree in Interactive Information Systems, which combined such disciplines as Computer Science, Business, Communications, Interactive Media, Design, as well as Social and Physical Sciences. Another significant component of this degree was pragmatic experience; therefore, Christian assumed core and leadership roles in designing software systems for Convergys, The Circuit, and Miami University.

Now working toward his Masters degree in Human-Computer Interaction at Carnegie Mellon University, Christian has expanded his programming and development expertise in a number of areas, including server and client side programming, database management, graphic design, video editing, and project management. In preparation for his involvement with NASA, Christian will take additional courses and attend seminars dealing with Artificial Intelligence and Robotics. His long-standing interest in the history of NASA and space research makes the opportunity to work with the agency not only intellectually and professionally satisfying but also personally exciting.

Greg Vassallo

- Carnegie Mellon University
  - Masters Program
  - Human-Computer Interaction Institute
- Syracuse University
  - S.I. Newhouse School of Public Communications
  - Major: Visual & Interactive Communications
  - Concentration in Photography
  - Minor: Information Management & Technology
- Work Experience
  - Lead Web Developer; ACC Web Services; Syracuse, NY; 2001-2004
- Skill Focus
  - Interface and Interaction Design, Prototyping & Web
  - Development Technologies

To Greg Vassallo, NASA's recent missions to Mars and Titan represent the limitless ingenuity of the human spirit, a pinnacle achievement in the curve of human history, and a positive force in a world filled with negativity. Greg is extremely excited about the opportunity to apply his passion for interface and interaction design to this project, and contribute - if even in a small way - to the advancement of NASA's mission.

Greg came to the interaction design field with a strong background in graphic design and photography. While he still loves the visual arts, he had a need to use his creativity to design and build things that people could use. He taught himself web design and programming, and within one year was promoted to Lead Web Developer at a development firm in his hometown, Syracuse, NY. After a few years of developing web applications for medium-sized clients, Greg was ready for a new and bigger challenge, which brought him to CMU and now the NASA project.

Greg has been married to his wife Manal for a year and a half, and she and his huge Italian family are the most important things in his life. When he' not working or hanging out with Manal, chances are he's listening to independent music, working on a personal photography or web project, reading, meditating, or watching a movie.

Karen Wong

- Carnegie Mellon University
  - Masters Program
  - Human-Computer Interaction Institute
- University of British Columbia
  - Major: Computer Science
  - Minor: Psychology
- Skill Focus
  - Educational Technology, Cognitive Tutors, Collaboration

Karen was first introduced to HCI while on an internship in Japan, where she worked on a 3D emotion-recognizing dance system. After that enlightening experience, she has continued to search for fun and exciting ways for humans to interact effectively with computers.

Upon graduating from the University of British Columbia with a major in Computer Science and a minor in Psychology, she spent 2 years at NGRAIN, a 3D graphics start-up company, as a software engineer and HCI lead. Her passion for HCI only continued to grow, so she returned to school for further specialization in the field. Since then, for the last year and a half, she has been working part-time towards her Masters of HCI at Carnegie Mellon University, as well as working fulltime on different projects with a focus on educational technology. She began with Project LISTEN's Reading Tutor: a tutor that teaches kids to read by listening to them. More recently she has joined the Kiva team, conducting user research on collaborative tools.

During her free time, Karen can be found swing dancing, snowboarding, or stargazing.

2 > Research

2.1 > Literature Review

Purpose

In order to establish a basis of understanding in the robotic domain, the team conducted a literature review. The literature review was focused on familiarizing the team with robotic terminology, current robotic projects, and robotic research. The literature review process consisted of each team member reading a few papers a week, and summarizing and presenting the findings at a weekly group meeting.

Understanding and using the terminology was crucial for the team when communicating with the various robotic teams during the usability studies. For example, the following lists and defines some domain-specific terminology that was utilized during the research and development process.

Activity: An action that the robot takes. Example: ‘turn left wheel.’

Sequence: Group of activities. Example: ‘drive forward’ is a sequence made up of low level actions such as ‘turn left wheel.’

Goal: Higher level activity of the robot, made up of low level actions. Example: ‘inspect upper region of worksite’ is a goal made up of many sequences of activities.

Mixed Initiative: Type of robotic operation in which the robot is controlled either by the operator or by onboard artificial intelligence software.

Waypoint: Position on the rover’s path. Waypoint is used to describe where the robot is traversing to.

Not only did the literature review help establish a domain-specific vocabulary, it also brought the team up to date on current robotic platforms. These current research projects included robotic search and rescue teams, past and current NASA missions, and robotic systems deployed by the United States Army and Navy. The team also became familiar with current robotic software systems. The team discussed planning software used by the Artificial Intelligence Lab and JPL such as ASPEN and CASPER. Different control techniques were also reviewed, such as Internet-based operation for future Mars missions. Outside of the NASA domain, the team looked at the Army and Navy’s Detection Assessment and Response System (MDRS) that controls autonomous guard robots.

The third focus of the literature review was for the team to become familiar with current robotic research. The research discussions centered around publications on human-robotic coordination, mission planning for multiple robots, utilizing current planning and scheduling software for autonomous spacecraft, and collaboration amongst scientists.

An entire separate paper could be written summarizing the research findings from our literature review. Here, however, is a brief summary of the points most salient to our specific challenge:

Summary

Need for Better Support for In-Situ Re-Tasking

Our literature review supported our client’s request for us to address the challenge of in-situ re-tasking in remote robotic planning. Sequencing a remote robot is an extremely time-consuming and therefore costly activity for current mission operation teams. Without a flexible commanding method, or a more intelligent on-board autonomous system, valuable unforeseen science opportunities are lost. Many research projects are based on this concern for allowing mission operation teams to spend more time gathering useful science data and less time dealing with low-level activity planning.

A typical approach to this problem found in the literature involved increased and improved on-board-autonomy levels. For example, various current robotic research projects are looking at allowing operators to define and rank high-level science goals, and then allowing the robot, or an off-board AI system, to formulate the detailed activity-level plan to achieve these goals. Then, when the robot is carrying out the plan and communication with the robot is not possible, the robot would make real-time decisions to adjust its plan if and when opportunistic science elements were encountered. One example of this type of on-board system with dynamic sequence generation is NASA’s CASPER (Continuous Activity Scheduling, Planning, Execution, and Replanning).

While this increased-autonomy approach is certainly promising, it seems that it is many years away from being fully flight-ready. Based on the current state of robotic research, a ‘human in the loop’ will be required for dynamic re-tasking for at least the next few Rover missions.

Human-Robot Interaction Taxonom*y*

Before tackling the problem of human-in-the-loop dynamic re-tasking, we needed to understand the important issues involved in 'tasking' in the first place. You can't re-task if you haven't already 'tasked'! Here are some important take-away suggestions and guidelines found in our literature review:

- The interface should provide the necessary information for the human to determine that an intervention – such as the human taking over a robot currently in auonomous mode - is needed.
- The interface should provide the necessary information for the human to gain a solid understanding of situation awareness: where the robot is and what it's doing.
- The interface should fuse information whenever possible in order to decrease the cognitive load of the human operator. For example, if there is one area of the interface that displays the environment map and another area of the interface that displays the robot's position within the environment, the user will be forced to fuse those pieces of information together in their heads.
- All interactions with the interface should be efficient and effective. Time is a critical resource when planning and re-planning remote robots.
- The interface design should be extensible enough to allow for additional robotic sensors and tools.
- The interface should provide the user with information about where the robot has already been.
- A map, or some sort of representation of the robot's environment, should be given a prominent size and position in the interface.
- A key consideration is that even if the user is familiar with the GUI and it's widgets, they still need to understand exactly how these input mechanisms will affect the actions of the robot.
- The operator should be able to rapidly identify the robot's current position, state, and health.

All of these guidelines helped us focus our first-hand research and formulate our interaction and interface design ideas.

Literature Review References

- "About Face: Rover Engineers Change the Rules for Driving", NASA Web Site, July 16, 2004. Author unknown.
- "Activities of the NASA Exploration Team Human-Robotics Working Group", AIAA Space 2003 Conference. Author Unknown.

- Backes et al., “Internet-based Operations for the Mars Polar Lander Mission”, Proceedings of IEEE, April 2000.
- Baker, M., et al., “Improved Interfaces for Human-Robot Interaction in Urban Search and Rescue”, Proceedings of the IEEE Conference on Systems, Man and Cybernetics, October 2004.
- Balaram, J., et al., “Enhanced Mars Rover Navigation Techniques “, Proceedings of the IEEE International Conference on Robotics and Automation, pages 926-931, 2000.
- Chien, S., et al., "ASPEN - Automating Space Mission Operations using Automated Planning and Scheduling," SpaceOps 2000, Toulouse, France, June 2000.
- Chien, S., et al., “Using Iterative Repair to Increase the Responsiveness of Planning and Scheduling for Autonomous Spacecraft”, Jet Propulsion Laboratory. Publication unknown.
- Clement, B.J. and Barrett, A., "Continual Coordination through Shared Activities," Proceedings of AAMAS 2003, ACM Press 2003.
- Cohen, P., Firoiu, L., “Abstracting from Robot Sensor Data using Hidden Markov Models”, University of Massachusetts at Amherst, February 1, 1999. Publication unknown.
- Croft, E.A., Kulić, D., “Estimating Intent for Human-Robot Interaction”, Proceedings of the International Conference on Advanced Robotics, Coimbra, Portugal, June 29 - July 3, 2003.
- Drury, J.L., and Yanco, H.A., “Classifying Human-Robot Interaction: An Updated Taxonomy”, Proceedings of the IEEE Conference on Systems, Man and Cybernetics, October 2004.
- Drury, J.L., Yanco, H.A., and Scholtz, J., “Awareness in Human-Robot Interactions”, Proceedings of the IEEE Conference on Systems, Man and Cybernetics, Washington, DC, October 2003.
- Drury, J.L., Yanco, H.A., and Scholtz, J., “Beyond Usability Evaluation: Analysis of Human-Robot Interaction at a Major Robotics Competition”, Journal of Human-Computer Interaction, Volume 19, Numbers 1 and 2, pp. 117 - 149.
- Frenger, “Robot Control Techniques”, ACM SIGPLAN Forth Report, 1997.
- Inderieden, R.S., et al., "Overview of the Mobile Detection Assessment and Response System," DND/CSA Robotics and KBS Workshop, St. Hubert, Quebec, October 1995.
- Kaelbling, L., Shatkay, H., “Learning Geometrically-Constrained Hidden Markov Models for Robot Navigation: Bridging the Topological-Geometrical Gap”, Journal of Artificial Intelligence Research, March 2002.
- Laird, R.T., et al., “MDARS Multiple Robot Host Architecture”, Naval Command Control and Ocean Surveillance Center. Publication unknown.

- McCurdy, M., Tollinger, I., Tollinger, P., and Vera, A.H., "Collaborative Knowledge Management Supporting Mars Mission Scientists," Proceedings of CSCW 2004, ACM Press 2004.
- Nesnas et al., “Toward Developing Reusable Software Components for Robotic Applications”, JPL. Publication unknown.
- Steinfeld, A., “Interface Lessons for Fully and Semi-Autonomous Mobile Robots”, Proceedings of the 2004 IEEE international Conference on Robotics & Automation, April 2004.

2.2 > Contextual Inquiry Process

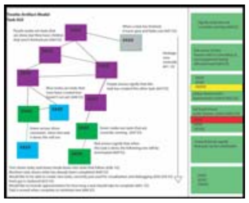
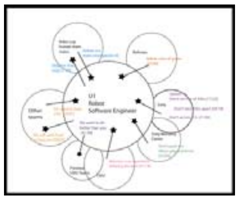
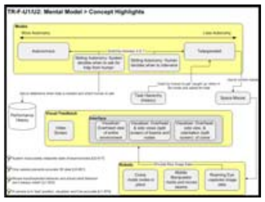
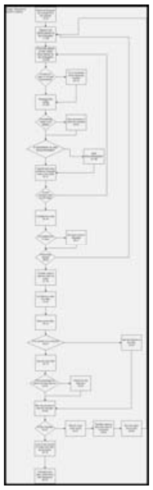
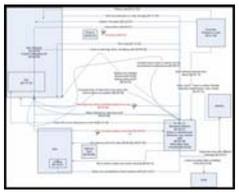
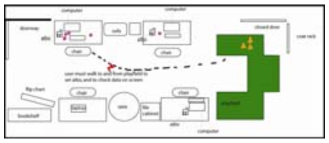
The research process that the team followed consisted of five steps:

- Collect
- Review
- Model
- Summarize
- Present

First, we collected data by either conducting a contextual inquiry or a post observation interview. We recorded the data with video, screen capture, audio recording, and written notes. Second, the team reviewed the data with analyzing software internally developed for the project. This software allowed us to watch and listen to video and audio recordings and easily attach notes to specific parts of the recordings. The software streamlined the note-taking process and greatly facilitated the modeling of the data. Third, the team modeled the data by creating one or more of the following models:

- Artifact model
- Cultural model
- Mental model
- Physical model
- Sequence model
- Work flow model

We determined which models to create depending on the data collected. For each set of data, we summarized the data by creating a list of key-findings. We then presented the data to both our faculty and to our client, who was non-collocated at the time.

2.3 > Contextual Inquiry Approach

Before embarking on our Contextual Inquiries, we wanted to make sure we knew what we'd be getting out of them. We knew that few, if not none, of the robotic domains we'd be exploring would be exactly analogous to the MER domain. Some robots would be fully autonomous with no human-in-the-loop; some would be commanded in real time and by a collocated operator; some would serve a purpose other than collecting science data. In order to overcome these discrepancies, we developed a matrix to organize the possible domains to explore and the features they shared with the MER domain. This would allow us to focus our CIs and know what elements we were looking for ahead of time.

We first researched robotic taxonomy in order to gain an understanding of what kinds of attributes we should be using to establish congruency with our domain. We also did some initial investigation into robotic research in the Pittsburgh area that we might have access to. The matrix, which we continued to refer to and update throughout the semester, is shown below.

GROUP	Functionality			Sensors			Autonomous			Data		Comm.		Users			Criticality			Collaboration (people:robots)			Time		Space	
	locomotion	artifact collection	altering environment	cameras	environmental	spectrometers	"fully" autonomous	mixed-initiative	teleoperated	scientific	human-robot awareness	sending	receiving	scientists	roboticists	engineers	high	medium	low	M:1	1:M	M:M	synchronous	asynch.	collocated	non-collocated
ROVER explore	x		x	x		x		x		x	x	x	x	x	x	x	x			x				x		x
Mindstorm Our game	x			x					x	x		x	x	x			x			x				x		x
RedTeam complete/win race	x			x	x		x				x	x	x		x	x		x		x			x	x		x
TRESTLE assembly	x		x	x			x	x	x		x	x	x	x	x	x			x		x		x		x	
Robocup win soccer game	x		x	x	x		x					x	x	x	x				x			x			x	
Subterranean explore	x		x	x				x		x	x	x	x	x				x		x			x			x
RedZone unclogs sewer pipes	x		x	x				x			x	x	x		x			x		x			x			x
Tekkotsu Programming Aibo	x	x		x	x		x	x	x		x	x	x		x	x			x		x				x	
Campus Police Campus Protection		x		x	x		x	x			x	x	x				x	x				x	x	x	x	x
Automated Turf Mar Mowing lawns	x		x	x		x	x		x		x	x	x		x			x		x			x		x	
Operating Room Managing a Hospital		x	x	x	x		x	x			x	x	x	x			x	x	x		x		x	x	x	x
MDRS Mobile Agents	x			x	x		x	x	x	x	x	x	x	x	x			x		x		x	x	x	x	x
Baja Fresh Managing People	x	x	x	x				x			x	x	x					x			x		x		x	
Life in the Atacama Exploring Earth	x		x	x	x	x		x		x	x	x	x	x	x			x		x				x		x

After reviewing our initial matrix, we recognized a need to look outside the robotic domain to gain a more fundamental understanding of in-situ re-tasking in general. We brainstormed potential areas where we could gain access to this type of work, and decided on nurses in an operating room, security dispatchers, and a fast food restaurant manager. Stepping outside the robotic domain also prevented us from developing ‘robotic blinders’, which would have had the potential detrimental effect of restricting our design ideas to those that are established and standard in the robotics field.

2.4 > Automated Turf Management

Summary

Automated Turf Management (ATM) is a project being developed at Carnegie Mellon University’s Robotics Institute in collaboration with the National Robotics Engineering Consortium. The goal of the project is to develop a fully autonomous vehicle that will mow a golf course with little or no intervention from humans, thus greatly reducing labor costs and increasing a golf course’s playable availability. The primary challenges facing this project are:

- The vehicle must achieve autonomous obstacle avoidance at a scale as small as a golf ball.
- The vehicle must achieve centimeter-level mowing precision in order to create the detailed hash patterns found on golf courses.

Three members of our team spent two hours interviewing the ATM team. We discussed key challenges they face and how they overcome those challenges. The discussion also went into other, similar robotic projects they were working on.

Primary Investigator: Sanjiv Singh
More Information: http://www.rec.ri.cmu.edu/projects/toro/

Key Findings

- Even with today’s best on-board autonomous driving and obstacle avoidance technology, there still needs to be a human in the loop. An autonomous robot will not always execute a plan exactly as expected, and a human is often required to intervene and make corrections to the plan. As such, E-Stop functionality is crucial.

- Even with the aid of real-time video telemetry, it is very difficult to tele-operate a non-co-located robot.
- The graphical representation of the robot in an interface used to command a non-co-located robot must explicitly indicate the robot's orientation.
- An aerial map (in the form of an orbiter image) is important for determining the robot's position.
- The interface should visually distinguish between elements 'as planned by the operator' and 'as executed by the robot'.
- One option for tracking a robot along a path is to have the robot's representation remain static and have the background in motion.
- On-board obstacle avoidance (and other automatic safety measures) should be able to be overridden by the human operator.
- The human operator needs to be able to investigate why the robot did what it did. This is typically achieved by reviewing detailed log files.
- The robot should ask for help when it needs it. This paradigm is referred to as 'assisted autonomy'.

2.5 > Baja Fresh

Summary

In order to gain more data on re-tasking, our team stepped outside the robotic domain, and interviewed a manager of a local fast food restaurant, Baja Fresh. Because the manager's work was time critical and involved interactions with customers, we conducted an interview instead of a contextual inquiry. The interview was conducted at Baja Fresh. Our questions focused on defining the manager's interactions with the other employees. We learned that re-tasking of employees occurred when either there was an unexpected increase in customers or decrease in customers. When there was a surplus of customers, the manager would make sure each employee was performing the tasks that he or she did best. During slower times of the day, the manager would allow the employees to switch tasks, in order to gain experience in other areas. If projected sales were low, the manager would send employees home. If more employees were needed than were currently working, the manager would call in an employee with a low hours count. It was very important for the manager to know what each employer was able to do in order to know which task to assign him or her.

"I need to look at skills, look at what needs to be done. Thus, when rushed, I know employees' skills and can delegate tasks to those who do it best."
-Baja Fresh manager

Key Findings

- Important to know the skills of workers (robots) to be able to quickly optimize when rushed.

2.6 > JPL – Mars Exploration Rovers

Summary

In 2003 NASA landed two exploration rovers on Mars. The rovers are controled from the Jet Propolution Laboritory in Pasadena California. The rovers, named Spirit and Opportunity, are capable of taking precise instructions or accepting rough instructions and then performing obstical avoidance on their own. Planners must take into account the safety of the rover while attempting to get as much scientific data as possible. Some of the primary challenges facing the project are:

- Coordinating the scientific needs of the international community to be executed by two robots.
- Communicate with and control a rover between 50 and 235 million miles away.
- Must be completely solar powered which causes fluctuations in power due to weather and rover placement.

Through two trips, five members of our team observed and interviewed planners and drivers working directly with the rovers.

Contact: Mike McCurdy
More Information: http://marsrovers.jpl.nasa.gov/home/index.html

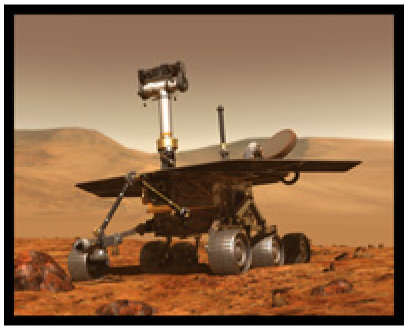
Key Findings

- Building the plans from a skeleton up can allow for a continual understanding of overall goals. This can help guide decisions made throughout the process.
- Although different mental models used in planning software can avoid early ambiguity since the model can be customized for the individual user, however this causes greater difficulty when trying to discuss the underlying plan with the larger group. In long term, high risk, expert systems a single mental model is vital.
- Although computer simulations can help planners bring out important problems with the plan they must be carefully monitored since missing or damaged data can be poorly interpreted by the computer.
- Critical information should be calculated automatically or within integrated interactions.

- Historical information is vital to robotic planning, however accessing and using the historical information can be limited and time consuming if not well presented and organized.

2.7 > NASA Seminar

Summary



Chris Ledger is rover driver on MER who received his PhD in Robotics from Carnegie Mellon University. He returned to CMU in the spring of 2005 to give a talk on his experiences on the MER mission. Three of our group members went to his talk and were able to ask him a few questions. His presentation provided a lot of insight into how the planning process works on MER, and the challenges they face. We were able to complete a Contextual Inquiry Sequence Model based on the data we obtained.

Speaker: Chris Ledger
More Information: **http://marsrovers.jpl.nasa.gov/home/**

Key Findings

- Planning a rover's mission is a detailed and collaborate effort in which many scientists, robotic engineers, mission managers, and activity planners must balance many needs. Conflicts between the needs of various scientists, between the need to obtain science data and the abilities/well-being of the rovers, and between differing opinions of team members in general must be reconciled in a short amount of time.
- The plans are made in a telescopic fashion. The first manifestation of the plan is a very high level overview, and it then gets refined and the details filled in over a series of pre-defined meetings, until finally the low-level commanding sequence is approved.
- The well-being of the rover most often trumps the desires of the scientists.
- Templates, or modules, are very important to the rover drivers. They will often use 'canned' sequences or commands in order to save time, and then adjust the parameters of the module to fit the current situation.
- The general paradigm is that there are 'drive days', in which the goal is to move the rover to a specific area, and 'science days', in which the rover has arrived at the desired location, where it then uses it's on-board scientific instruments to collect data.
- The team uses RSVP (Rover Sequencing Visualization Program) to simulate the commands before sending them to the rover.

- At all stages in the planning process, there are detailed approval processes set up so that everyone on the team is completely aware of and okay with the plan.
- Depending on the characteristics of the terrain, the rover drivers will put the rover in either Auto-Nav mode (on-board obstacle avoidance, slower), Blind Drive mode (no obstacle avoidance, faster), Semi-Autonomous mode (the rover has veto power to stop if it encounters an unsafe situation), or a non-autonomous mode (the rover drivers specify the exact path the rover should take a very fine-grain detailed level).

2.8 > Operating Room

Summary

Hospital operating room management has some parallels with a human-robotic teaming, so we explored this context to gain and understanding of how an operator or manager of multiple agents coordinates individual schedules, activities, and resources. We conducted two contextual inquiries with charge nurses at the University of Pittsburgh Medical Center's Montefiore and Presbyterian Operating Rooms (ORs). Three members of our team spent three hours conducting the in-context inquiry in the hospital's operating room front desk or "command central". We discussed key challenges they face and how they overcome those challenges.

Key Findings

The analogy we abstracted from the context was that the OR can be viewed as a semi-autonomous robot with the charge nurse acting as the operator to process the patients' cases which are tasks, and the OR personnel and equipment are resources or parts of the robot that must be coordinated. The ability to find the parallels in these contexts allowed us to find valuable data about the needs of a system operator.

The Information needs of the OR manager:

- When interpreting data about a situation, the OR manager needed access to raw data in addition to the abstracted data in order to trust the information they are getting on the situation.
- The OR manager expressed the need to be able to override security procedures in extreme situations.
- The OR manager needed to know understand the information about the situation and the capabilities of robot/personnel in order to appropriately task them.
- Having access to history of events was very important to the OR manager. In particular it was useful to see how similar events were handled when unexpected events came up, and they wanted to see a recent history of events so they could estimate the status of

resources and tasks. They indicated that in addition to logging events in a standard protocol they made annotations of events and off-nominal behaviors not captured by the system.

- In addition to seeing a history of what had been done with similar situations, the managers indicated that generalizable help and basic protocols for events were useful, especially suggestions and guidance for first-time users. They provided example artifacts of resources they used to maintain operations knowledge and described its usage.
- The OR manager needed to see the status of multiple events or keep multiple event status data peripheral but still accessible.

Re-stasking Recommendations

Based on the CI data we collected on in-situ re-tasking performed by operators we came to several generalizable rules for re-tasking that are summarized below.

Re-tasking heuristics:

- If the original goal is to keep the OR running, the manager must manage and coordinate physical resources of equipment and space, human resources, and time.
- They knew when to re-task because of known triggers to re-tasking that included emergencies and alarms being sounded, cancellations or deferment of tasks, and non-emergency additional tasks.
- The OR manager knows the options and alternatives available because of experience and knowledge of the resources, the situation, and how these variables will factor together.
- The OR manager acquires such knowledge and becomes capable at handling re-tasking through practice and experience in the context.
- The OR manager averages the data from different sources to get the best feedback on the situation and status of tasks and resources. They prioritize this situational data by factoring in knowledge of past cases, necessity of the tasks, others' opinions and use this to process the information and make a choice between the alternatives.
- In making re-tasking decisions, the OR manager considered that it may be necessary to task personnel into overtime, or call in "on-call" personnel, but they were also aware of the costs and how that related to the original goal of keeping the OR running. This seemed to be parallel to the concept that an operator of a robot may overtask the robot beyond its stated capability, locate new resources, or reconfigure plans for the robot.
- Some ORs are being equipped with real-time video feed from the individual OR suites, but it was still very difficult to tell exactly what the personnel were doing in the room and to communicate tasks with them.

- The OR manager would frequently walk through the entire OR to see who was in what OR suite, and what they were doing. From this we are making the recommendation that some kind of overall map and data would be needed for determining the robot's position.

2.9 > MDRS - Mars Desert Research Station

Summary

MDRS 2005 involved the investigation of Mobile Agents, a wireless EVA data management system. This software, Mobile Agents, runs on moving computers carried by the astronauts and rovers, and enables astronauts to ask questions, and the rovers to carry out their commands. The investigation occurred during a Mars simulation, out in the Moab desert, where astronauts lived in a hab, and worked with two special robots.

We had the unique opportunity to observe consecutive half-days of planning, and even an EVA! We saw first-hand the difficulties scientists had with specifying contingencies in the planning tools that they had, achieving the same shared mental model, coming to a consensus with all the stakeholders, and interpreting a 3D world on a 2D image or map.

During the EVA, the astronauts used speech to communicate with their personal agents, and to command the robots. Of note, their being collocated with the robots was different from our project scenario, however, the planning, and the need to re-task was very real, and offered us many opportunities to learn from their experiences.

More Information: http://www.marssociety.org/MDRS/fs04/

Key Findings

- Method of specifying a re-tasking instant a priori. For example, go 50 feet, analyze rock, if the rock is red, re-task; otherwise continue with the plan.

- Organization of data is essential and important to astronauts/scientists.
- Importance of labeling areas (worksites, waypoints). Have common language for reference.
- Consistent labeling and naming across images and maps.
- Importance of modules, templates and reusable data.
- Templates are very important. They must be sterile, i.e. can't have information for previous days, etc.
- Access to history for future planning.
- E-Stop needs to be quickly accessible. (Had problem with robot running over rock and astronaut unable to stop it)
- Method of asking questions and making comments about the plan.
- Use of parameters to define an action instead of programming, easier for high level planning .
- When using variables or parameters:
  a) indicate which are required vs. optional
  b) indicate questions and comments for parameters
- Separate a name for an action from the actual activity
  -- ie, name: Drive to Waypoint 32. Action is drive
- Interface accommodates for different skill levels, beginner vs expert vs debug > Braums Composer.
  - Indicate abilities/limitations of different assets both in text and visually.
- Allow for quick drastic changes if needed, as well as allow for undo.
- Need consistent time frame! Especially if collaborating in different time zones. (Resultant loss of data from cameras)
- Again structure in planning interface: high level first, click for more detail.
- Key breakdowns:
- There was a lot of difficulty in judging distance. (Example: depth perception for cliffs, as well as distance between two points where the robot must travel).
- Planning interface is unable to support branching (alternate plans) .
- Again, an enormous amount of time was spent to gain shared understanding. (Shared mental model)

2.10 > Robo Soccer

Summary

We observed the Robo Soccer team at Carnegie Mellon University. The Robo Soccer team participates in the yearly RoboCup soccer competition in which teams of Sony Aibos play soccer against each other. RoboCup is an international joint project to promote AI and robotics research.

We conducted a contextual inquiry with a graduate student who is highly involved with the back-end programming of the Aibo robots. We focused on how the Aibos were re-tasked. The re-tasking was a result of testing and de-bugging the behavior software. For the Robo Soccer team, re-tasking only occurs during the development stage since the robots are fully autonomous during competition.

We observed the operator testing the Aibo's behavior software in the development and testing environment in the Robotic Institute at CMU. The operator re-tasked the robot by turning off certain behaviors. Switching behaviors on and off let the operator concentrate on the output of one or two behaviors at a time. The operator would show us the code she was changing and then execute the code. We would then observe the Aibo moving on the playing field, which was located next to the operator's work station. The operator was able to view system feedback from the testing either as a simulation of the robot's view or as text-based print lines. We noted that the operator was more comfortable looking at the textual feedback than the visual feedback.

Key Findings

- Important for operator to see software changes made from co-workers
- Important for operator to be in same room with Aibos during testing in order to quickly see results
- Operator mainly referenced textual feedback from the testing
- Important for operator to have quick communication with co-workers

2.11 > Subterranean

Summary

Hazardous underground activities such as mining and cave mapping have in the past been conducted by humans, but with autonomous robotic navigation, environmental data collection, and computer generated maps it is unnecessary to take such risks anymore. Three members of our team spent two hours conducting the in-context inquiry of the Subterranean Robotics project in the Robotics Institute's high-bay area of the Field Robotics Center. We observed the typical work of one member of the team, and discussed their efforts to improve the reliability and robustness of their cave-crawling through enhanced algorithms for navigation and obstacle avoidance, and innovative use of 2-dimensional sensors to produced 3d maps and learned about the data fusion process. In our contextual inquiry we uncovered breakdowns and needs of operators that command autonomous robots navigating through remote environments that do not have detailed charts or knowledge of the terrain available.

Key Findings

Our key findings from the Subterranean Robotics group were that the following areas needed improvement and would have to provide the user with greater support:

- Similar to the context of Mars exploration rovers, the remote and hazardous environment that these cave crawling robots are traveling through puts constraints on the ability of human operators to retrieve the robots in case of an emergencies or accidents. This imposes the need for help, error prevention, and error recovery onboard the robot to be very reliable, and for our interface to provide as much support for that as possible.
- Systems for providing feedback and informing the user of the status of the robot are especially useful when sending the robot into unknown environments. The operator may want to know information such as the current battery level and an estimate of how much further the robot can go before it can safely return.

- The communication with the robot is in a raw and unprocessed format whether it be commands being sent to the robot or data being retrieved from the robot. Labeling of this information as it comes and goes could help the operator to more efficiently and effectively command the robot.
- The cave crawling robots being developed here record sensor data of the environment they travel through. The data is not processed onboard the robot, but once it is downloaded software is used for fusing the data together into a visualization of the 3-dimensional space. This visualization is the purpose of sending the robot into the cave.
- The visualization representation of the space is good for understanding objects and dimensions of the space, but another representation that is needed is a map of the area. The operator needs to know the possible paths that the robot may take, and navigation on a more 2-dimensional level is more useful for supporting this.
- A final finding from this CI was that easy data manipulation is valuable to the operator of the robot particularly when the task of the robot is to gather information about the environment it is deployed in. The process of retrieving the data from the robot and producing something useful and easy to understand required a set of complex and confusing interactions with several applications including command-line shells for accessing the data files.

2.12 > Tekkotsu

Summary

Tekkotsu is an open source project at Carnegie Mellon University to develop an application development framework for intelligent robotics. Throughout the development process programmers check the code and the robots to make sure bug fixes and new features work with the robot correctly. The primary challenge of this project is to develop a framework for cognitive robotics.

We observed this project because of the close similarities between the coding process and the in-situ re-tasking task. While coding in general is not similar to re-tasking, Feature implementation and bug fixes require changing of a list of sequence. Running the sequence, and then based on the results of the sequence, changing the code again. It was this sequence of events that was the team's primary focus in this observation.

Three members of our team observed a feature implementation session for one hour. During analysis of the contextual inquiry the team created a workflow, cultural, sequence, artifact model and physical models. We also created a mental model diagram to document the way that the developers approached this unique problem. After creating these diagrams we summarized the results with the following take away points.

More Information: http://www-2.cs.cmu.edu/~tekkotsu/

Key Findings

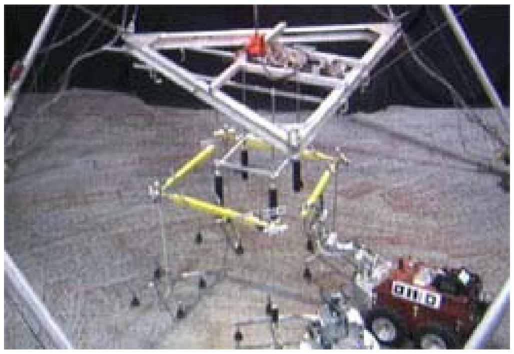
- Based on the models and observations made, the following list of key findings was compiled. Many of these findings are illustrated in the sequence model.
- Support of past documentation & see status of co-workers are doing in relation to your work.
- Show information at different levels of details (verbosity).
- Importance of good IDE features, auto-complete, syntax checking, pre-compile syntax checking.
- Relaying the system's state and the robot's state.
- Show length of time to complete task – lots of waiting.
- Importance of accurate feedback – debugging.

- Use of simulations to help predict results without risking/needing robot & understanding of the physical constraints of the robot.

- Allow functions and features to be turned on and off.
- Importance of reusable code & sanitize data before it is used again – During the contextual inquiry code snippets were often copied and pasted from one part of the code to the other. However, this would often cause compile problems

2.13 > TRESTLE: Autonomous Assembly by Teams of Coordinated Robots

Summary



TRESTLE focuses on developing architectural tools to coordinate actions performed by multiple robots, in the context of complex assembly projects. Since the number of possible failure modes over many robots working on long sequences of actions is high, TRESTLE incorporates the ability of humans to cooperatively manage tasks. Thus, with sliding autonomy, the system can help the operator to find the optimal mixed-initiative to manage assembly.

We observed three robots working together to assemble 4 beams. The operator was non-collocated, and for the trials we were observing, the operator was completing the assembly task by pure teleoperation. We later watched the video of the session again with the project team members, and interviewed them at the same time. While we could not observe every type of trial using sliding autonomy, we were able to ask the researchers many questions to gain a better understanding. In general, we were able to learn what were the difficulties in being non-collocated, and what types of aids helped, or could be useful.

More Information: http://www.frc.ri.cmu.edu/projects/trestle/index.html

Key Findings

- Raw imagery from camera is important to user in lieu of inaccurate visualization…can't rely solely on visualization.
- User's accurate understanding of orientation is vital to accurate commanding. Perspective plays a big role, and needs to be consistent.
- Since the operator is non-collocated, initially becoming familiar with the robots and space arrangement helps a great deal. Expert users can make assumptions.
- Since there is an overwhelming source of information, there needs to be careful filtering of information. Too many windows, or excessive feedback makes it hard to focus attention on what is important.

- Somehow have to deal with the fact that commanding / expected behavior and actual robot behavior don't always match. One idea: provide different degrees of granular control for different accuracy needs…ex: long traversal vs. fine motor scientific extraction.
- Converting a 3D space to a 2D display and back is extremely challenging. 2D displays have difficulties with depth perception.
- Performance history is crucial for robot determining
- if it can do something
- how long it will take
- if a human is better suited to control robot (as opposed to autonomous)
- which human is best suited for control, etc.
- Consider use of one type of input mechanism/widgets to control different robots -- system re-interprets input per robot.
- Human needs access to a representation of task hierarchy history (what the robot has been doing) in order to command it after period of autonomy (independent robotic activity).
- For mixed-initiative, there were several key concepts that the Task Hierarchy represented:
- When nodes have been called but haven't run
- What tasks are currently running
- What tasks are switchable (having different choices)
- Hierarchical relationship between tasks
- Specific details of tasks: name, equipment, control
- Minimizing what is shown
- Using colors to signify meaning
- Showing tasks that has uncompleted sub-tasks (children)
- Showing task constraints with color
- Showing that tasks might cause others to be triggered

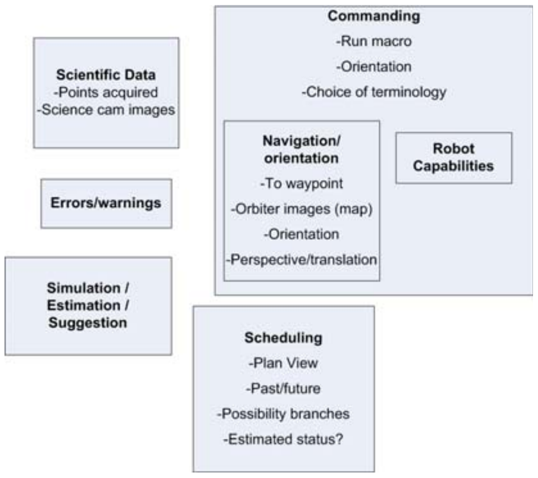
4 > Design

4.1 > Summary

After conducting and analyzing multiple contextual inquiries, the team entered the design phase with a rich understanding of what is needed in a re-tasking interface. One of the main goals of the design was to ensure every aspect of the interface had sufficient data supporting it. The following goes into detail about the seven interface areas and references the observations that support them.

4.2 > Interface Features

We first approached the design phase by brainstorming primary features and functions. We individually brainstormed features, and then as a group and created an affinity diagram. The items from the affinity diagram were categorized into the following interface areas.



Interface Features

- Commanding
- Scientific Data
- Errors/Warnings
- Navigation/Orientation
- Robot Capabilities
- Scheduling
- Simulation / Estimation / Suggestion

Commanding

Ideas for the commanding area consisted of the need for multiple modes, including an advanced mode for troubleshooting and viewing of low level data and controls as well as a novice mode. One of the research papers recommended the ability to access all levels of control so that an operator could drop down to low-level control during unusual or anomaly situations. The idea of commanding the robot to traverse to waypoints originated from the NASA seminar. In addition, the need for an emergency stop was supported by multiple robotic teams.

Supporting Data

- *Robo-Soccer*
  - CI_RC_U1 > mental model > operator manages code at all levels of granularity
- *Red Team*
  - Has e-stop button for emergency situations
- *NASA Seminar*
  - NASA_Sem > sequence > [6:30] > using waypoints to specify where to go
- *Tekotsu*
  - CI_TK1_U1 > mental model > operator manages code at all levels of granularity
- *Jet Propulsion Laboratory*
  - use built-in flight rules
- *Research Papers*
  - *"Interface Lessons for Fully and Semi-Autonomous Mobile Robots"*

Scientific Data / Situational Awareness

Scientific data was defined as the data collected by the robot during the mission. This data would include the pictures taken by the onboard camera as well as maps generated by the onboard pictures and orbiter pictures. Along the lines of scientific data, the interface needs to communicate situational awareness as well. Situational awareness is of utmost importance because it helps the operator visualize the location and environment of the rover. To ensure sufficient awareness, the interface needs to communicate a correct description of the robot's environment. Needed details of the robot's environment include: temperature, amount of visible light, terrain description, time of day, distance to targets, and description of current weather.

The interface would ensure that the data being given to the user is accurate. Because of a reported difference between what the robot's and orbiter's perception, it was suggested that the interface be able to calculate and display this difference in perception to ensure optimal operator performance. In Robo-Soccer, an overlay camera image was paired with the robot's computer generated 'world view,' this ensured that there was consistency.

Scientific data would also include awareness data, such as information displaying what tasks the robot is currently performing. Other important information that the operator needs to know include connection status and battery and equipment status. In respect to the given task, the operator in the game situation will need to know point values and game score.

Supporting Data

- *Robo-Soccer*
  - RS_U1 > [1:00] > Ground Truth vs. World View
- *CHI Tutorial*
  - Guidelines for HRI
- *Trestle*
  - Interface shows what task is being performed
- *NASA Seminar*
  - Need to know what terrain to avoid
  - Need to know equipment status

Errors/Warnings

It is essential that the interface display errors and warnings. A warning could be given when the battery is running low or equipment isn't properly working. A warning could also be given when the robot is about to run into an obstacle. Data supporting the display of errors and warnings included contextual inquiry data from Robo-Soccer, interview data from the NASA seminar, and background papers. In the Robo-Soccer contextual inquiry, the operator made sure an error didn't occur by ensuring that the robot didn't run off the course and into an obstacle. Evidence also came from the NASA seminar. The Mars rover driver described problems when the onboard AutoNav (auto-navigation) unexpectedly caused the robot to drive randomly in circles. In addition, the paper “Improved Interfaces for Human-Robot Interaction in Urban Search and Rescue” shed light on the importance of enhanced awareness of the robot.

Supporting Data

- *Robo-Soccer*
  - CI_RC_U1 > Work Flow > [1:00]
  - CI_RC_U1 > Work Flow
- *NASA Seminar*
  - NASA_Sem > Sequence > [31:02]
- *Research Papers*
  - *“Improved Interfaces for Human-Robot Interaction in Urban Search and Rescue”*

Navigation/Orientation

In order to navigate, the controller needs to know the layout of the robot’s environment. In addition, in order to correctly formulate commands, the user must be aware of the orientation of the robot. The pre-defined task allows for two different views: the orbiter’s view and the robot’s view. It was suggested that a map of the environment be shown at all time.

Supporting Data

- *CHI Tutorial*
  - HRI Guidelines
- *Robot-Soccer*
  - RS_U1 > [25:02] > Map Overlay
  - RS_U1 > [1:00] > Ground Truth vs. World View
  - RS_U1 > [25:02] > Overlay Map & Robot view tasks based on skill and experience
- *Subterranean*
  - ST_U1 > [23:40] > Visualization of 3D view
  - ST_U1 > [49:20] > 3D Map
  - ST_U1 > [20:15] > Controller needs maps

- *Trestle*
    - TR_U1 > [11:32] > Object Orientation Maps
    - TR_U1 > [47:40] > Object Orientation Maps

Robot Capabilities

In order to effectively control a robot, the operator must know what the robot can and cannot do. In the case of the Mars rovers, the rovers run off of solar energy. The drivers are keenly aware of this, and so always know that the robot must stop at locations that will allow it to get enough sunlight. Even when scientists want to stop the rover to do observations, the scientists will override the scientists' requests to ensure the robot will have enough battery power.

Not only is knowing a robot's capabilities important when deciding what commands to send, it is also important when deciding how the robot will execute commands. Robot command execution can happen two different ways. One, the execution is continuous, meaning that if new commands are sent while the robot is performing a task, the robot will automatically do those commands and then go back to the original plan. The second way is to tell the robot to stop after execution and wait for more commands at the next communication uplink. This ensures that the operator can determine where the robot is located. The mode of operation chosen generally depends on the robot's level of precision when executing commands. Since the Lego Mindstorm responds to commands with low precision, it has been decided that the robot will stop after execution and wait for more commands.

Supporting Data

- *NASA*
    - *NASA_Sem > Sequence > [5:25], [6:10] > need to know robot capabilities*
    - *NASA_Sem > Cultural > [45:26] > must know duration of task in order to plan*
- *Operating Room*
    - *CI_HS_U2 > Workflow > [0:27] > doctor and nurses given tasks based on skill and experience*
    - *CI_HS_U1 > Sequence > [12:40] > nurse's task depends on abilities and experience*
- *Baja Fresh*

        - Important to know abilities of employees

Scheduling

In order to re-task the robot, the interface has to support scheduling of activities. In the brainstorming session, ideas for the scheduling aspect of the interface included

- ability to rank each task
- graphical manipulation of activities in a queue
- automatic optimization of the plan based on priorities
- pre-planning of different options
- graphical manipulation of activities in a graph
- gaant chart view of plan
- spatial visualization of the plan overlaying a map

Based on evidence, we also wanted our interface to be able to show an activity log of past events. We found that history and documentation would be important features based on the contextual inquiry with Robo-Soccer. During the contextual inquiry, the robot operator needed to know who made a particular change to the code, and when. Other supporting evidence came from the NASA seminar, in which the rover driver stated that events that happened in the past affect the current plan. The knowledge of past events helped the NASA planning team create the schedule. For instance, when one of the rovers got stuck in the sand, the rover planners new to stay away from certain areas in the future. Along the lines of providing history and documentation, it was decided that the interface should have an archive of photos taken which could be searched by the operator.

Supporting Data

- *Robo-Soccer*
    - RS_U1 > [25:02] > Map Overlay
    - Important to have documentation of changes
- *JPL*
    - Scientists prioritize activities
    - Planners use a gaant chart to visualize plan
- *Nasa Seminar*
    - Knowledge of past performance affects plan

Simulation / Estimation / Suggestion

We discovered that simulation of the activity was important because it allowed robot operators to visualize the activity and to foresee if any problems might occur. The Robo-Soccer team, the NASA Mars Rover planners, and Red Team all referred to a simulation of the task when planning. Simulations of both the current plan and the different plan options would be valuable when the operator had to choose between options. We decided that the interface would also have an estimation ability, which would include an estimation of the amount of time needed to complete each task, and the amount of time needed to finish the plan. Further supporting the operator would be a suggestion area that would allow other scientists and the backend AI to make suggestions for the plan.

Supporting Data

- *Robo-Soccer*
  - Used simulation of performance
- *NASA Seminar*
  - *Rover drivers use simulation tool when visualizing results of commands*
- *Red Teamr*
  - Used simulation software to see how the vehicle would handle different types of terrain and different obstacles

Basic Design Requirements

The underlying focus of the interface design is to follow basic usability design specifications. These specifications include the integration of shortcut keys to create, edit, and delete items. To facilitate the commanding of the robot, we chose to design for the use of macros. Macros would allow the user to input a high-level command that the backend software would break down into robot-recognizable commands. We also decided to implement customizable placement of pallets and windows.

- Shortcuts
- Macros

- Customizable pallets/windows
-

“I want an interface so anybody can use it… I don’t want to have to rely on the one person staying in the desert for months at a time to be the only person who can operate it.”
Mike Wagner, Life in the Atacama

4.3 > Scenarios

In order to transition into the creative process of the design phase, each team member wrote up scenarios analyzing the interaction between the operator (s) and robot (s). The scenarios spread across the spectrum: some were very realistic while others were very extreme. By having extreme usage scenarios, we not only were able to come up with creative ideas, but it helped us to get comfortable with brainstorming.

Realistic Scenarios

- Operator playing the game in our test site
- Driver of MER A is waiting to uplink the plan for the upcoming sol
- Astronaut beginning fourth day on Mars
- Scientist evaluating a rock

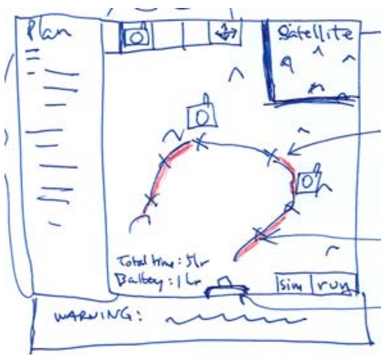
Extreme Scenarios

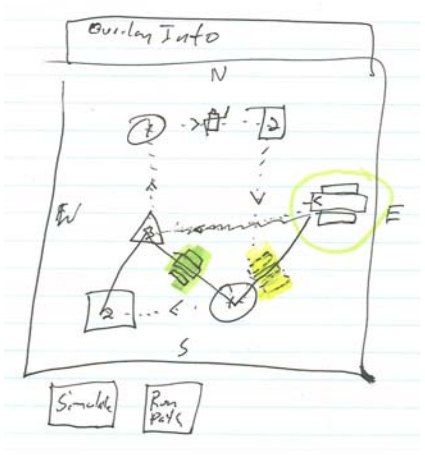
- Really smart robot
- 1 billion robots
- Really cheap robots
- Human on planet operator is controlling
- Human standing behind robot
- 1 Robot, a million scientists with 1 day each to control it
- 1 Robot, a million scientists at one time
- Voice only commanding (hands tied behind back)

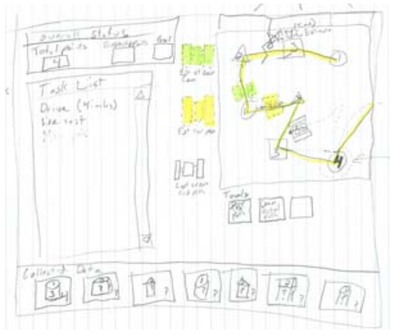
4.4 > Low-Fidelity Stage

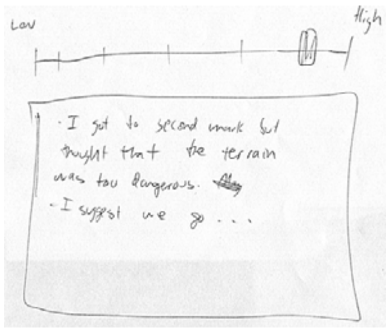
4.4.1 > Designed paper-prototypes

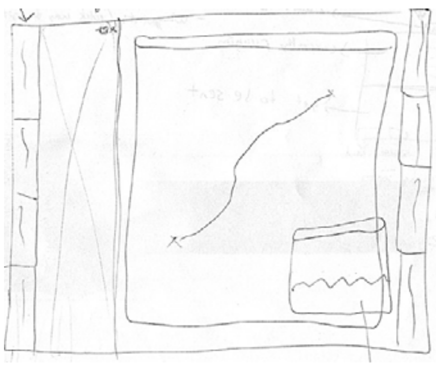
The following shows the results of our first round of prototyping. We created these paper mock-ups individually and in groups of three. By putting our ideas on paper, we were able to smoothly communicate them to the rest of the group and to conduct a critique with our professors and client.

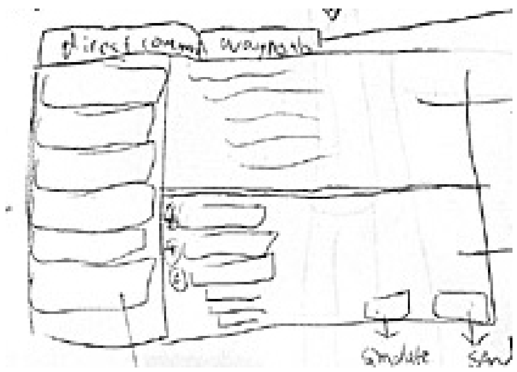






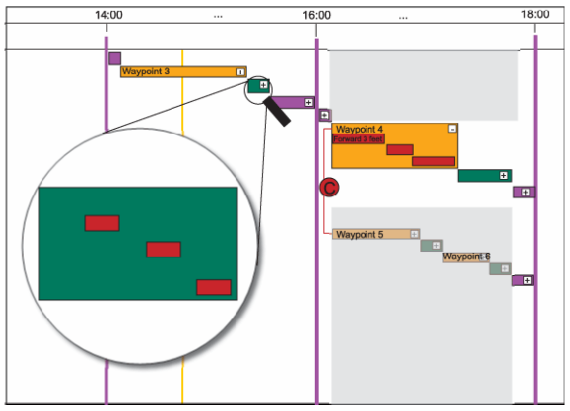

Low
High
· I got to second mark but thought that the terrain was too dangerous.
· I suggest we go . . .

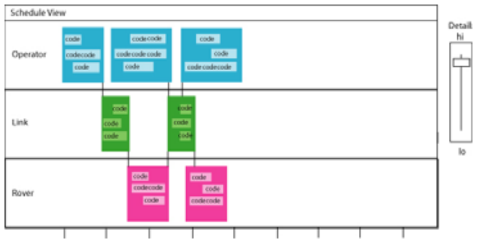


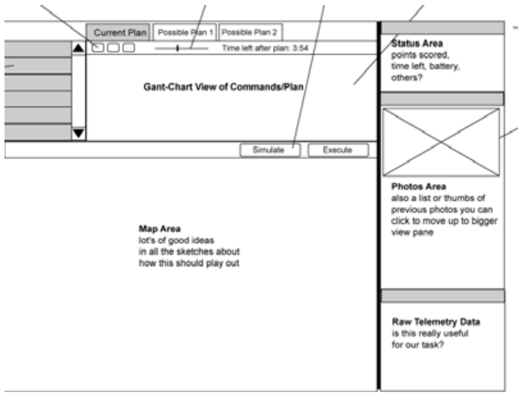


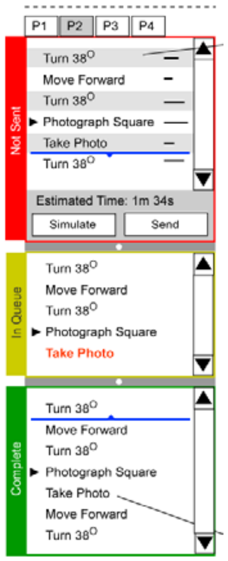
4.4.2 > Designed screen-shot-prototypes

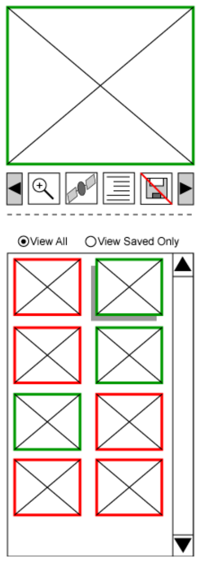
After critiquing the paper prototypes amongst ourselves and our professors, we took another step in the low-fidelity design phase. More detail was added to the paper prototypes, and the new designs were made into digital mock-ups.

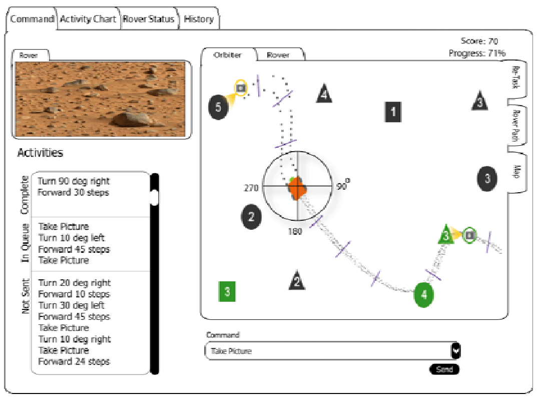





Current Plan
Possible Plan 1
Possible Plan 2
Time left after plan: 3:54
Gant-Chart View of Commands/Plan
Simulate
Execute
Status Area
points scored, time left, battery, others?
Photos Area
also a list or thumbs of previous photos you can click to move up to bigger view pane
Map Area
lot's of good ideas in all the sketches about how this should play out
Raw Telemetry Data
is this really useful for our task?







4.4.3 > Improvisation of Task

In order to fully understand the task of commanding a non-collocated robot, our team acted out a scenario. This improvisation helped us detect the questions and demands the interface would have to support.

We set up an environment that simulated the game. Instead of blocks, we used tables, and instead of a robot, we used one of our team members. We also had a score keeper, an operator, and a note taker.

The game play was straight forward. The operator had an orbiter's view by standing on a balcony above the playing field. After looking at the position of the robot, the operator would determine the best path to the next target. The operator would then cover his eyes and verbalize the commands to the robot. This instantly brought up the question of how the commands should be stated. We told the operator to structure the commands in whatever way felt most comfortable. Robot commands were delivered in this form:

> *"Turn to Right 90 degrees, take 3 strides forward, 90 degrees left turn,*
> *3 strides forward, take picture."*

After receiving the command, the robot would move. The human robot simulated errors by doing the following:

- Attempted to take a literal direct path through obstacles
- Missed the waypoint by some amount
- Moved with different stride length
- Moved at different speeds
- Sometimes didn't move at all

When a ‘take picture’ command was given, the score keeper would determine if the robot was at the right angle and close enough to take the picture. The score keeper would then determine the number of points obtained. During the entire process, the note keeper took note of observations.

As the improvisation unfolded, we identified tools and resources needed in the environment. We noted the need for a grid system in order to command the robot to traverse a certain distance. The improvisation helped us to understand the limitations, frustrations, and uncertainty involved. We took note of the takeaways from the improvisation and referred to them later as we progressed through the design process.

Important Takeaways

- IMPORTANT TO KNOW ROBOT CAPABILITIES
- Data: Operator missed target since he didn’t know Robot’s stride

- IMPORTANT TO KNOW ROBOT ORIENTATION AND RESULT OF COMMANDS
  Data: Operator kept turning his body to imagine he was the robot. He needed this physical visualization of what his commands would make the robot do.

- NEED TO SEE ROBOT VIEW
  Data: Because of operator angle, it was hard for him to see what was in front of robot.

- NEED TO KNOW SCORE
  Data: Operator kept asking, “What did I end up getting?”

- NEED TO KNOW ROBOT CURRENT CAPABILITIES
  Data: Robot changed stride, this simulated robot changes in capabilities (results of wear and tear)

- NEED TO USE UNIFORM LANGUAGE
  Data: Operator commanded the robot to take “steps” and “paces”

4.4.4 > Defined critical areas

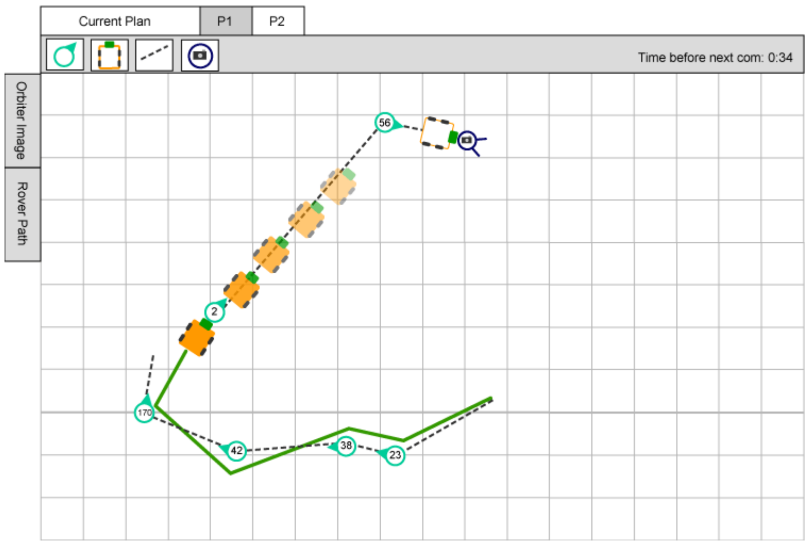
Three of the seven interface areas were classified as "critical areas," meaning they were defined as vital to the controlling of the robot. These three areas included:

- Activity Planner
- Map
- Gannt Chart

4.4.5 > Conducted Think-Aloud User Studies on screen-shots

In order to receive quick feedback on widget design before continuing with the design process, we presented paper prototypes of the three critical areas to four participants in a think-aloud study. The focus of the study was to discover whether or not the widgets afforded the correct use and whether the visual design was comprehensive. Two different representations of the map were tested in order to detect which design elements could be added to the next iteration. The following displays the designs presented to the participants, and a summary of the results.

Map 1

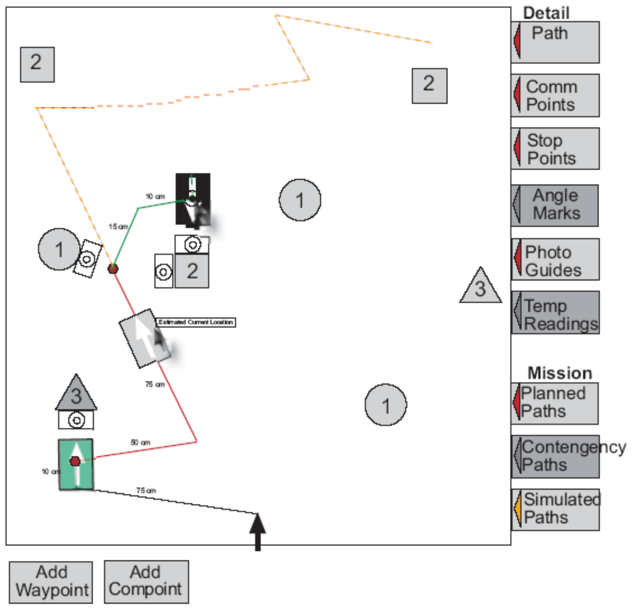


Results

Results from the think-aloud of the paper prototype of Map 1 showed that participants liked the overall design, but found some of the widgets unclear. The direction of the rover was not clear, which was caused by the participants viewing the translucent rovers as the past and not the future. The angle widgets were designed to afford entering in a specific angle for the rover to turn. The participants misunderstood the angle widgets, and instead of viewing the widgets at angles, three of the four participants viewed them as numbered waypoints.

- Faded rover resembles motion blur
  Data: (SKO –TA-U3, KNW-TA-U1, KNW-TA-U2)
- Angle widget resembles waypoint
  Data: (SKO –TA-U3, KNW-TA-U1, KNW-TA-U2)
- Rover path representation unclear
  Data: (SKO-TA-U3, KNW-TA-U1)
- Number inside angle widget unclear.
  Data: (SKO –TA-U3, KNW-TA-U1, KNW-TA-U2)

Map 2

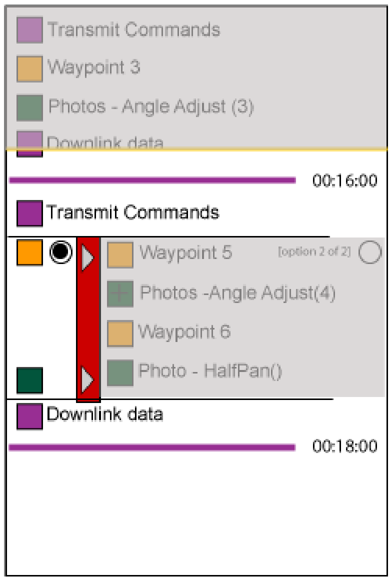


Results

After explaining the meaning behind the widgets, and how they foresaw the interaction between the user and the interface, the participants noted that they would like to see a legend for the widgets and interactions. The results of the study showed that participants favored the design of Map1 over the design of Map 2. However, our team noted that the design of Map 2 offered more functionality, and that a balance between simplicity and functionality needed to be made.

- Camera representation unclear
  Data: (SKO-TA-U3, KNW-TA-U1)
- Rover position unclear
  Data: (SKO –TA-U3, KNW-TA-U1, KNW-TA-U2)
- Unclear how to manipulate path
  Data: (SKO-TA-U3, KNW-TA-U1)
- Layer labels are unclear
  Data: (KNW-TA-U1)
- Units missing
  Data: (KNW-TA-U1)

Activity Planner

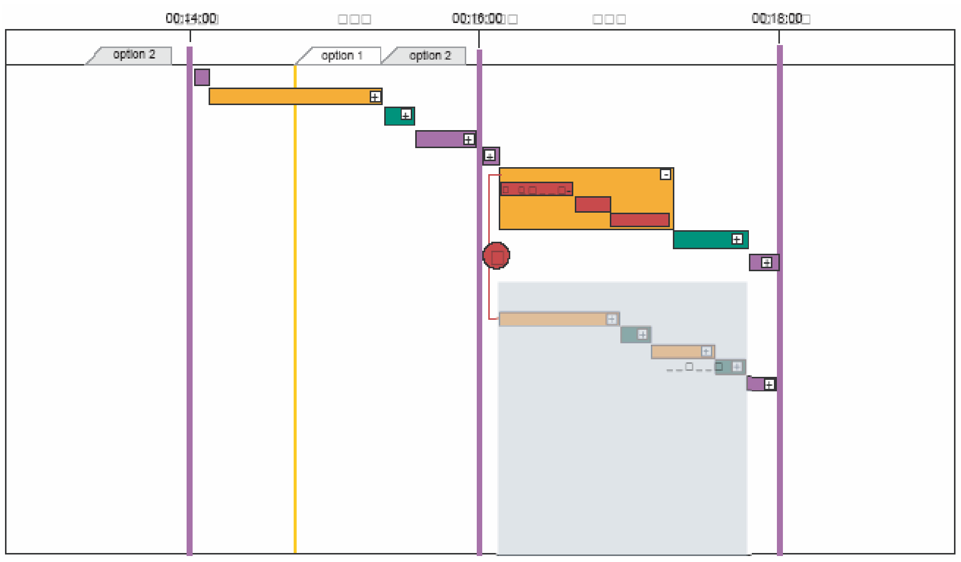


Results

The study conducted with the Activity Planner showed that the notion of contingencies was understandable, but that the representation in this design was difficult to understand. Our team set out to re-think and re-design contingencies, and the design process that contingencies went through is apparent throughout the rest of the design stages. We also noted that unlike the map designs, the Activity Planner was difficult to test in the paper-prototype stage because of the lack in interactivity.

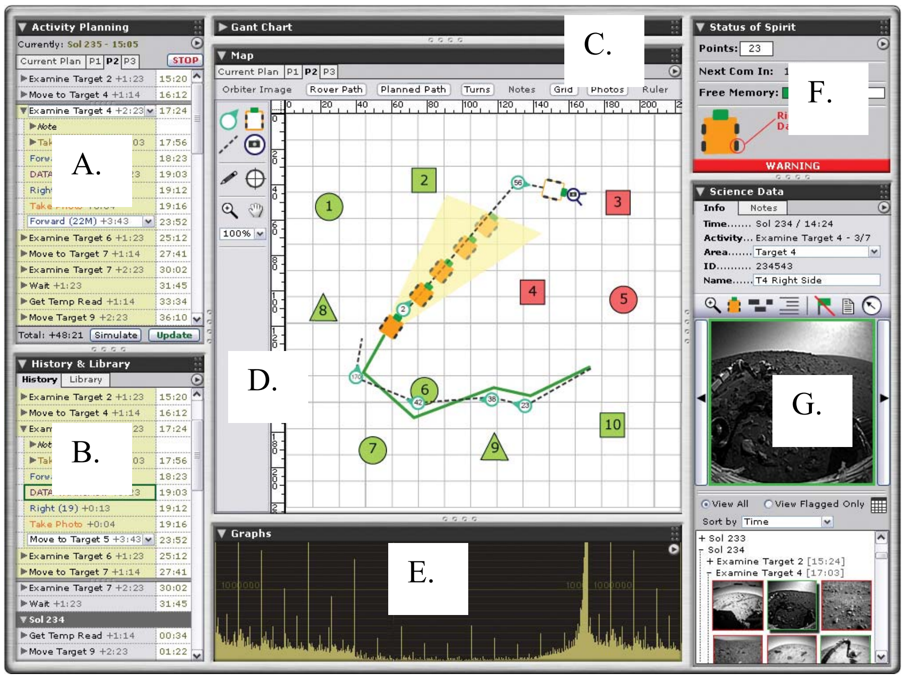
- Contingency representation unclear
  Data: (SKO-TA-U3)
- Tab labels unclear
  Data: (KNW-TA-U1)
- Layers are unclear.
  Data: (KNW-TA-U1, KNW-TA-U2)
- Contingency divider is unclear.
  Data: (KNW-TA-U1)

Gant Chart



4.4.6 > Designed Consolidated Prototype

Following the think-aloud user studies on the paper prototypes of the Map, Activity Planner, and Gant Chart, a consolidated prototype was developed. The consolidated prototype featured a consistent design amongst the seven sections. It also illustrated where these sections would be positioned within the interface. The seven sections were enclosed in scalable panels, with the intent of allowing the user to minimize and maximize sections. In order to thoroughly test each section of the interface, our objective was to divide the interface into the seven sections and test them separately.



Sections

A. Activity Planner
B. Map
C. Gant Chart
D. Science Data
E. History & Library
F. Status of Robot
G. Graph

Section A: Activity Planner

Purpose

The Activity Planner was developed to update and create robotic plans. Its initial design resembled a text editor, such as Eclipse. The user manipulates the plan by adding and deleting the textual commands and start times. An activity drop-down list would be available.

Section B: Map

Purpose

The Map section was designed to graphically represent the plan as well as the robot's environment. The Map section allows the user to change the plan by editing the map's widgets.

Section C: Gant Chart

Purpose

The Gant Chart section was created in order to graphically represent the plan.

Section D: Science Data

Purpose

The Science Data section was designed in order to keep track of the information that the robot collected. For instance, in or scenario, the robot is taking pictures of its environment. The Science Data section allows the user to organize the images taken. Also, this section allows the user to flag images for further analysis.

Section E: History and Library

Purpose

The History & Library section was developed in order to store past plans as well as let the user store past sequences of activities.

Section F: Status of Robot

Purpose

The Robot Status section was designed in order to update the user on the robotic physical state.

Section G: Graph

Purpose

The Graph section was created in order to graphically show the change in a chosen variable over time. The intent is the have the user choose from a list of items including robotic battery status, robotic distance traveled, etc, and the graph will graphically show the change in the selected item over time.

4.5 > Interactive Stage

4.5.1 > Created interactive prototypes

After applying the results from the think-aloud study of the paper prototypes, the critical areas were developed in Macromedia Flash and tested. Heuristic Evaluations were conducted amongst the group members. After implementing necessary changes indicated from the Heuristic Evaluation, the flash prototypes were presented to participants in a Think-Aloud study.

4.5.2 > Conducted analytical tests

Before we presented the interactive prototypes to participants in a user-study, each member of the team conducted a heuristic evaluation of all three prototypes. Jacob Nielson's criteria for critical incidences were used. The following lists out the problems individual team members discovered in each of the prototypes. Some problems were found by multiple members. For more information about this list of problems, such as how many members found them, refer to the Appendix. Because a think aloud study quickly followed the heuristic evaluation, only a few of the problems found from the heuristic evaluation were fixed.

Jacob Neilson's Criteria

- Visibility of System Status
- Match between System and the Real World
- User Control and Freedom
- Consistency and Standards
- Error Prevention
- Recognition Rather than Recall
- Flexibility and Efficiency of Use
- Aesthetic and Minimalist Design
- Help and Documentation

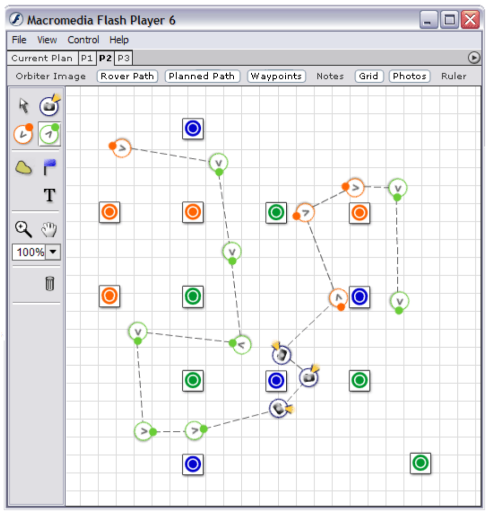
4.5.2.1 > Map

The Heuristic Evaluation pointed out various problems with the map design. Because of time constraints, only five of these problems were addressed in the next iteration. However, they original list of problems was kept in mind throughout the rest of the design process.

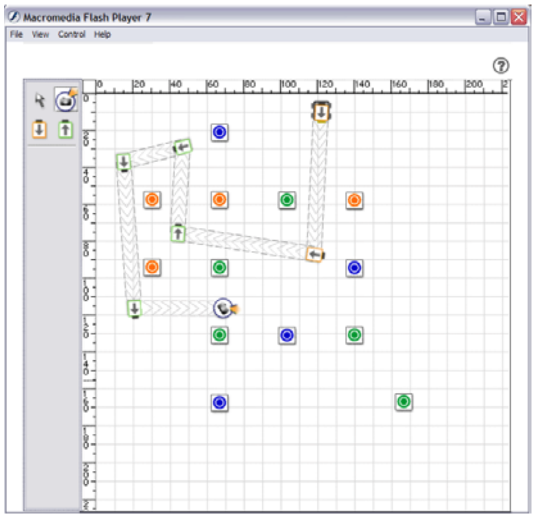
The following design changes were made as a result of the heuristic evaluation.

- Improved detection of rover direction (waypoints redesigned)
- Visualization of rover width (rover path redesigned)
- Meaning behind tools (added tool tips)
- Starting position of the rover marked (image of rover added)

Before Heuristic Evaluation


Macromedia Flash Player 6
File View Control Help
Current Plan P1 P2 P3
Orbiter Image Rover Path Planned Path Waypoints Notes Grid Photos Ruler
100%

After Implementing Changes



The following is the complete list of consolidated problems found from the heuristic evaluation. More detail about each problem, including the severity rating and number of people finding each problem is shown in the Appendix.

Problems

ID	**Problem**
HE-01:	Difficult to know position of last waypoint

HE-02:	Difficult to see where rover starts
HE-03:	Compass is missing
HE-04:	User can overlap waypoints
HE-05:	Difficult to see which targets are or will be visited
HE-06:	User can make the robot run over the targets
HE-07:	Awkward that camera icon is also a waypoint
HE-08:	User has to remember meaning behind red and green waypoints
HE-09:	Map doesn’t tell number of points robot could obtain
HE-10:	Difficult to visually compare plans
HE-11:	Map doesn’t state point value of each block
HE-12:	Estimate time of each plan is not shown
HE-13:	Map missing coordinates/units
HE-14:	No cancel or undo widget for last waypoint
HE-15:	No help in case of problem in plan
HE-16:	Rotate widget only accessed through MOVE cursor
HE-17:	No tool tips
HE-18:	No shortcuts for switching between tools
HE-19:	No Zoom out too available
HE-20:	Can’t interact with the trash can
HE-21:	No indication of which waypoints can rotate
HE-22:	No indication of distance values
HE-23:	Camera icon not rotating at center of camera
HE-24:	Rotate arrows in cursor move strangely

Updates

ID	Problem	Solution
HE-02:	Difficult to see where rover starts	Add icon showing rover’s position
HE-06:	User can make the robot run over the targets	Redesign path to show width of rover
HE-08:	User has to remember meaning behind red and green waypoints	Redesign waypoints

HE-15:	No help in case of problem in plan	Add Help icon
HE-17:	No tool tip	Add tool tips

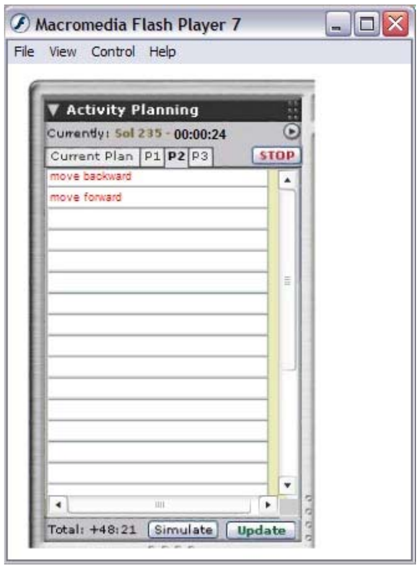
4.5.2.2 > Activity Planner

The Heuristic Evaluation pointed out various problems with the activity planner design. Because of time constraints, only five of these problems were addressed in the next iteration. However, they original list of problems was kept in mind throughout the rest of the design process.

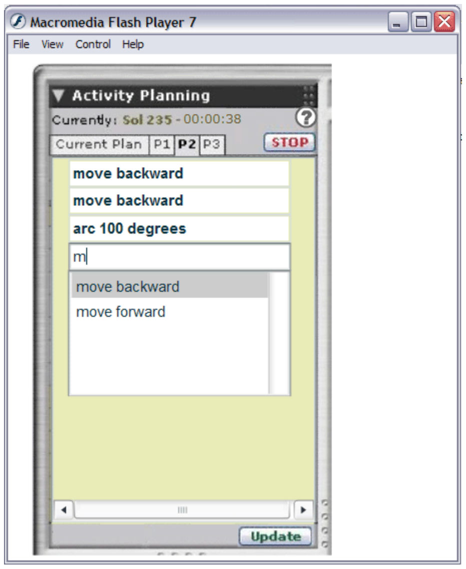
The following design changes were made as a result of the heuristic evaluation.

- Improved traversal (added tab functionality)
- Increased speed of feedback (decreased time until pop-up appeared)
- Clarified meaning behind interface colors (established that red meant emergency by changing command color to black)
- Improved interaction (made the auto-fill box appear only when a word was recognized)
- Clarified where user needed to type (made interface only have one blank text box, and more would appear as the user typed)

Before Heuristic Evaluation


Macromedia Flash Player 7
File View Control Help
Activity Planning
Currently: Sol 235 - 00:00:24
Current Plan P1 P2 P3 STOP
move backward
move forward
Total: +48:21 Simulate Update

After Implementing Changes



The following is the complete list of consolidated problems found from the heuristic evaluation. More detail about each problem, including the severity rating and number of people finding each problem is shown in the Appendix.

Problems

ID	Problem
HE-25:	Button interaction unclear
HE-26:	Tab traversal needed (and shift Tab)
HE-27:	Impossible to determine when plan is updated
HE-28:	Difficult to determine different between command and note
HE-29:	Activity list pops up too slow
HE-30:	User has to remember command layout
HE-31:	Auto-fill box should not appear when no match found
HE-32:	Awkward that next items to run are at bottom
HE-33:	Stop button may be interpreted as stopping robot, not stopping the sending of the plan
HE-34:	Text boxes are always present
HE-35:	There is no way to add a command line between two existing command lines
HE-36:	Different types of commands are not distinguished by color
HE-37:	No way to delete a command line
HE-38:	No way to copy/cut/paste a collection of lines
HE-39:	Lines with sub-lines are indented further than lines without sub-tasks
HE-40:	Contingency plan unclear
HE-41:	Indication of note is not clear
HE-42:	STOP button wipes out entire plan
HE-43:	Red color indicating keyword looks erroneous

Updates

ID	Problem	Solution
HE-25:	Button interaction unclear	Change appearance of buttons
HE-26:	Tab traversal needed (and shift Tab)	Add tab functionality
HE-29:	Activity list pops up too slow	Decrease time until pop-up
HE-43:	Red color indicating keyword looks erroneous	Change color of commands to black

HE-31:	Auto-fill box should not appear when No match found	Make auto-fill box only appear when words recognized
HE-34:	Text boxes are always present	Make text boxes appear as user types

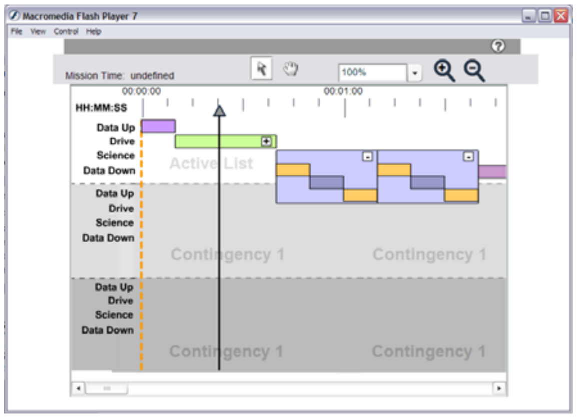
4.5.2.3 > Gannt Chart

The Heuristic Evaluation pointed out various problems with the gannt chart design. Because of time constraints, just two of these problems were addressed in the next iteration. However, the original list of problems was kept in mind throughout the rest of the design process.

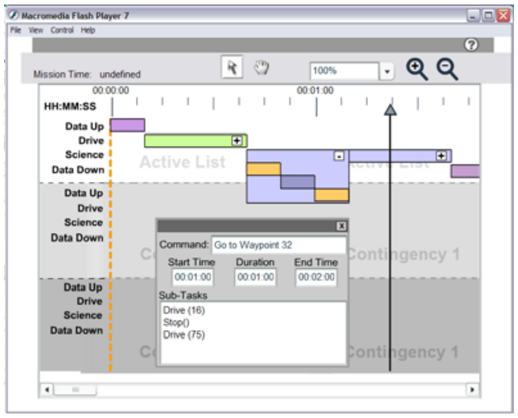
The following design changes were made as a result of the heuristic evaluation.

- Improved navigation (Added 'hand' cursor)
- Increased amount of information about activities (Added an edit box)

Before Heuristic Evaluation



After Implementing Changes



Command: Go to Waypoint 32

Start Time	Duration	End Time
00:01:00	00:01:00	00:02:00

Sub-Tasks
Drive (16)
Stop()
Drive (75)

The following is the complete list of consolidated problems found from the heuristic evaluation. More detail about each problem, including the severity rating and number of people finding each problem is shown in the Appendix.

Problems

ID	**Problem**
HE-44:	Inconvenient to use scroll bars
HE-45:	Activities can exist without names
HE-46:	'Label' on side doesn't match up all the time
HE-47:	Active items can overlap with contingency area
HE-48:	Meaning behind staggered tasks unclear

HE-49:	Contingency label confusing
HE-50:	Difficult to see information about activities
HE-51:	No indication when activity name is longer than shown
HE-52:	Meanings behind different colors unclear
HE-53:	Current zoom level not displayed
HE-54:	No visual indication of edit mode
HE-55:	Expand and collapse widgets differ in activity planner and gannt view
HE-56:	Background color of each contingency plan is the same
HE-57:	Current time not always viewable
HE-58:	No marking of the units in timeline
HE-59:	No help available
HE-60:	How do you add contingencies?
HE-61:	No affordance for interaction
HE-62:	Window focus (scroll) does not move with needle
HE-63:	Needle below actions
HE-64:	Magnifying glass not a cursor icon

Updates

ID	Problem	Solution
HE-44:	Inconvenient to use scroll bars	Add ‘hand’ cursor
HE-50:	Difficult to see info about activities	Add an edit box

4.5.4 > Conducted a think-aloud study

Quickly following the Heuristic Evaluation, a think-aloud study was conducted with the flash prototypes of the three critical areas. The purpose of the study was to detect how easily the participants interacted with the three interface designs. Three college-aged interns from Apple were chosen as participants. Each study involved all three prototypes and lasted about an hour long. During the study, audio and screen movement were captured. Afterwards, the data was analyzed and the following criteria were used to write-up UAR's of the critical incidences. Refer to the Appendix for the protocol used.

Criteria

- Misunderstanding of orientation
- Misunderstanding of interaction
- Misunderstanding of graphic
- Misunderstanding of label
- Not yet available in version
- Lacking ability in design
- More instruction or training for task needed
- Bug
- Unexpected behavior of interface
- User frustration
- User gives up
- Unable to complete task
- Task completed in suboptimal way
- Stated incorrect assumption/understanding (mental model)

4.5.4.1 > Results of Map

This version of the map tested a new concept of waypoints, as encouraged from Think-Aloud Study 01 with the paper prototypes. It also presented the concept of tools as way-points as well. For example, positioning the camera tool caused the rover to automatically move to the specified location to take a picture. The widgets in the prototype were redesigned based on results from Think-Aloud 01 and the Heuristic Evaluation.

Problems

- TA-02: Still confusion with camera as a waypoint
  Data: (KNW-06, KNW-11, SO-10)

- TA-14: Did not know point of forward or backward
  Data: (MK-06)

- TA-07: Expected use of keyboard
  Data: (SO-07, MK-03, MK-04)

4.5.4.1 > Results of Activity Planner

The results from the user study on the Activity Planner gave ideas for contingency design. Results showed that a possible design for contingencies would be to mimic the way the sub-task interaction was designed. We came across this result on accident. In the task presented, the contingency was between two pre-defined sub-plans. Instead of simply inserting the two names of the sub-plans, the users wanted to plan out the different sub-plans and be able to view them both at the same time. The prototype tested did not allow the users to do this.

The idea of automation came up. A design idea would be to offer more automation, and simply have users fill in the missing blanks. This would aid in reducing errors. Unexpected feedback included the uncertainty of command order. The user wasn't sure if he was supposed to tell the rover to turn first, or give the orientation first. One way this problem would be solved is through more instruction.

Problems

- TA_02_30: Thought arrows made more sense for contingency, because are grouping things and hiding them.
  Data: (KNW-40, SO-13)

- TA_02_32: No idea what subtasks was for
  Data: (MK-15)

- TA_02_36: Once sub-task created, had trouble going back to main task
  Data: (MK-18)

- TA_02_37: Not sure which part of contingency is automatic
  Data: (MK-20)

- TA_02_34: Not sure arc or move first
  Data: (MK-21)

4.5.4.3 > Results of Gannt Chart

This test was designed to explore whether a gannt chart layout of a rover's activity plan would assist the user in gaining information and understanding of both high and low level activities, temporal flow, and contingencies. The task involved exploring the interface the interface and explaining what was happening for several two minute intervals. In order to do this they had to examine the overall plan, sub-plans, properties, and contingencies for the active plan. We asked for a written description of what was represented to give us insight to their conceptual model of activity flow.

Problems

- TA-17: Contingency layout not clear. Thought active path would drop down when an activity failed
  Data: (KNW-18, KNW-19, KNW-22, MK-13)

- TA-18: Thought contingency auto-resumed default active path when finished with contingency
  Data: (KNW-20, MK-25)

4.5.4.4 > Tips for Think Aloud Studies

In addition to detecting problems with each of the interface, the think aloud study shed light on a few problems with the team's current process of preparation before the study. Problems were mainly caused by the lack of sufficient time between the heuristic evaluation and the think aloud study. But, because tight time constraints are very much a part of all projects, we came up with the following recommendations on how to ensure that the usability study runs smoothly.

- Ensure tasks have enough instruction by practicing study on fellow team mates
- Establish a design implementation freezing point. This will allow for more time for practice and preparation of the usability study
- Have team members become experts of one design, and all usability studies on that design will be ran by the expert team member. As is true with quick and constant design changes, it is hard for all team members to keep mental note of the latest changes and be able to run usability studies on those changes.

4.6 > Redesign Phase

4.6.1 > Rethought the concept of contingencies

In presenting the first interactive versions of our front-end prototype to new users we observed a difficulty with the concept of contingencies across each trial. We had think-aloud tasks requiring the user to first explain and manipulate a plan with the visual map aspect of the interface, second explain activities as represented in the animated gannt chart area of the interface, and third was to use the the command editor area of the interface to enter a sequence of activities that were already shown on a planning map. While our interface had ways to support contingencies being built by the user and then selected for execution, the real difficulty with that interaction was that users did not fully understand what was intended by "contingency", so they could not properly use or interpret what it was for. This posed a severe problem for supporting pre-planning and fast in-situ re-tasking which is a vital functionality of our interface, so to overcome this we changed the terminology from contingency to 'option'. In addition to modifying the label we changed the interaction so that it would be less detailed low-level planning, but more high-level option identification.

4.6.2 > Conducted a think-aloud study on game

We had conducted contextual inquiries in several different contexts, all of which had some aspects that made it close to or similar to our end-user's context. Through our user research and takeaway matrices we fused our data sets into a conglomeration that gave us insight into a context that doesn't yet exist. However, with that data we still did not have a complete understanding of how a person would behave in the future context posed in our challenge. Prior to designing an interface based solely on the needs found in our mixed-context user research we chose to simulate the end-user's task and context as it was described in the challenge. With a game simulation upfront in the process we could observe how a user would behave in-context and learn about the needs we would have to support.

The challenge described the game as a simplified robotic control task that would be analogous to controlling a rover on Mars to visit science targets and geological features on the surface of the planet. The game simulation used symbols and icons to represent targets, the robotic components, and the control interface. The best approach to exploring how a person would re-task 'in-situ' was to have them think-aloud while playing in the simulation prior to having them do so with a fully developed prototype. One of the reasons for this is that would be much more costly to change the prototype once development had plugged into the backend and produced interactions that could actually control the Lego Mindstorm. This allowed us to test a high-fidelity version of the game without an interface ready to go.

The results of the game showed that players were in fact thinking about 3 moves or increments in advance of their current position, and they considered an average of 3 possible options at a time for their immediate next moves. They developed interesting strategies within the hour of testing that we conducted.
Following game play we asked them to do a card sort of “plan modification types”, and from that we were able to come up with names for interactions that users would typically want to use, as well as prioritize the interactions and strategies they would be using most frequently while re-tasking in the actual field test.

4.6.3 > Narrowed focus of interface

After gaining a more solid understanding of how the game might play out and the strategies involved, we re-evaluated the time we had left and the design ideas we’d have to cut. We decided that the gant chart view of the plan, while important, was not crucial to our investigation, and would have to be side-lined in favor of focusing more attention to the map and the activity planner.

Of course, gant chart views of rover plans have proved to be a valuable tool in visualizing a plan and helping the user quickly gain an accurate mental model of the temporal relationship between activities. We are not suggesting that gant chart plan visualizations be abandoned from all future design pursuits. However, in the context of our specific domain, we felt that such a representation of the plan was not crucial enough to warrant the time designing such a representation would take away from exploring other areas of the interface.

4.7 > High-Fidelity Stage

4.7.1 > Conducted study with participants at JPL

After going through the final redesign phase, we set up a usability study with five participants at JPL. The five participants were each highly involved with the current Mars Exploration Rover missions. We recruited a Tactical Uplink Lead for Opportunity, a Tactical Activity Planner for Opportunity, a software engineer manager for rover uplink software, a strategical planner for Spirit, and a rover driver for Spirit.

The method performed during the user study resembled a mix between a cognitive walkthrough and a think aloud. This type of study was done in order to get an expert opinion while conducting a cognitive walkthrough. After walking through a task and answering the cognitive walkthrough questions outlined below, we directed questions towards widget interaction and overall visual design of the interface.

Cognitive Walkthrough Questions

- **Does that make sense to you?**
- **Does this happen on MER?**
- **What do you do when it happens on MER?**
- **How do your current systems represent and handle such events?**
- **How do you react to something like this happening in the MER context?**

We showed each participant an interface design of various scenarios, each portraying one of the different option types of rush, delay, skip, delete, detour, and fork. We aimed at observing whether or not a particular activity occurred with the mars rovers, and if the feedback presented by the interface was sufficient. Our cognitive walkthrough questions aimed at directing the participant to evaluate the interface design as if it were incorporated into the current mars missions. By answering whether or not a particular activity occurred, the participant was able to better visualize their day to day activities. They usually answered our questions with a specific example. Since the team wanted feedback before implementing the design, the prototype presented to the participants was in paper form, showing designs done with Adobe Illustrator.

After the study, we analyzed the data and produced a list of problems and recommendations. Because of time constraints, the recommendations were prioritized. The following lists the recommendations with highest priority. For a complete list of problems and recommendations, refer to the Appendix.

4.7.1.1 > Recommendations (priorities: 0 = high)

Represent all option types in the activity planner as a unique branch.
Priority: 0

Rather than trying to represent different types of options (Delay, Rush, Insert, etc.) with separate graphical representations for each, show them all as a separate branch going off the main current activity line, similar to how we have the Fork Option type represented in the current designs. Users indicated that this would be easier for them to quickly see and understand (as opposed to trying to represent, for example, a Rush in a special way), and that this representation would allow them to easily see the ramifications of an option, such as time.

It's more important for us to focus on making the interactions of quickly modifying the plan in real-time easy and efficient than to focus on the implementation of pre-planning options.
Priority: 0
User 5 noted, "Make it easy to build a path, put emphasis on that... instead of having to plan alternative paths, just allow the operator to be able to quickly build a path… this is where macros would come in handy."

Add a distance scale to the map.
Priority: 0
This is an obvious but overlooked feature that we should add to the map.

When mousing-over a waypoint on the map, highlight this waypoint in the activity planner (and visa versa).
Priority: 0
Users commented that they need a way to connect the waypoints on the map to the waypoints in the activity planner. One suggestion was to label the waypoints on the map, but we feel that this would introduce too much clutter. Another suggestion was to have a tool-tip, so that when the user moused over a waypoint on the map, a tool-tip was displayed with the name of the waypoint. We feel that this method requires too much cognitive effort: the user has to glance at the tool-tip,

store the label name in their short-term memory, move their visual focus over to the activity planner, and then find the label name. We suggest a solution where the user mouses over a waypoint on the map, and this waypoint is visually highlighted in the activity planner, and visa versa.

Represent the rover's executed drive path with more detail.
Priority: 1
This may not be possible given the constraints of the game and the available technology. One possibility is to have the orbiter take images every 10 seconds, and then at each 2 minute downlink point the user would receive each of the orbiter images. This would give the user a more clear understanding of how the rover got from point A to point B, as opposed to the current design, where the user is only shown point A and point B, but nothing in between. This of course would require a change to the game's rules.

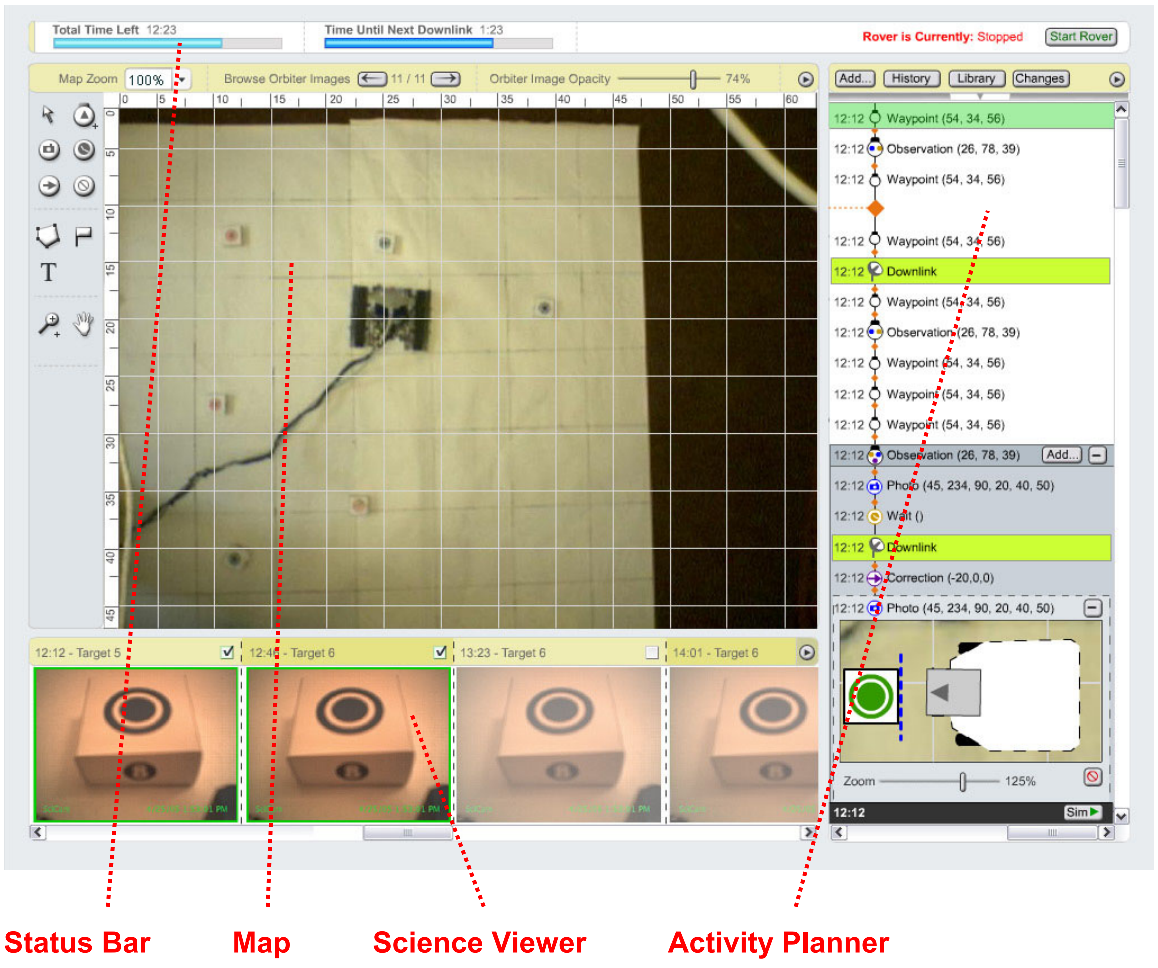
Make the downlink times stand out more in the activity planner, possibly using a special dark line.
Priority: 1
In the current design, we have downlinks represented in the same way as other activities: there's an icon with a text description next to it. Users indicated that they need a quick way to see where the downlink times occur; our current design may not be adequate.

CMU MHCII NASA Project > Summer 2005

Rover Planning and In-Situ Re-Tasking Interface

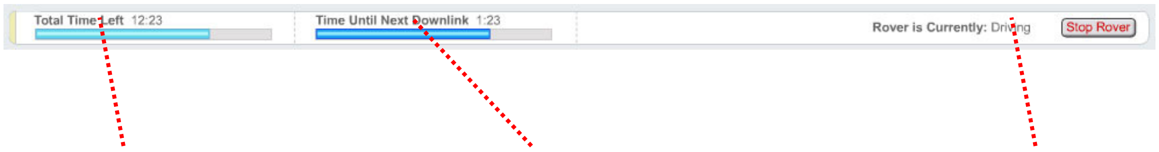
Alpha Prototype Functionality Specification > V 1.0

1 > Overview



12:12 Waypoint (54, 34, 56)
12:12 Observation (26, 78, 39)
12:12 Waypoint (54, 34, 56)
12:12 Waypoint (54, 34, 56)
12:12 Downlink
12:12 Waypoint (54, 34, 56)
12:12 Observation (26, 78, 39)
12:12 Waypoint (54, 34, 56)
12:12 Waypoint (54, 34, 56)
12:12 Waypoint (54, 34, 56)
12:12 Observation (26, 78, 39)
12:12 Photo (45, 234, 90, 20, 40, 50)
12:12 Wait ()
12:12 Downlink
12:12 Correction (-20,0,0)
12:12 Photo (45, 234, 90, 20, 40, 50)
12:12

2 > Status Bar

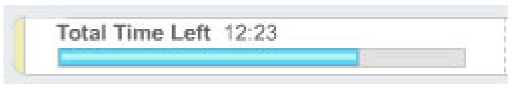


Total Time Left Indicator **Time Until Next Downlink Indicator** **Rover Status**

2.1 > Purpose

The Status Bar gives the user important feedback about the status of the game context and any information about the Rover that isn't appropriately represented in other areas of the interface. It is placed prominently at the top of the screen because it contains crucial information about the user's task.

2.2 > Total Time Left Indicator



2.2.1 > Purpose

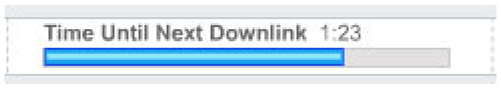
The Total Time Left Indicator gives the user feedback about how much time is left in the game.

2.2.2 > Functionality & Interactions

Before the game begins, the Total Time Left text reads "30:00" (thirty minutes), or however long a game is. The blue indicator bar is 'full', which means it is completely covering up the light grey background behind it. Once the game starts, the time ticks down backwards, and the progress bar gets smaller relative to how much time is left.

The Total Time Left Indicator is a purely system feedback widget. The user does not interact with it directly, as it is only meant to provide feedback.

2.3 > Time Until Next Downlink Indicator



2.3.1 > Purpose

The Time Until Next Downlink Indicator gives the user feedback about how much time is left before the next downlink will occur.

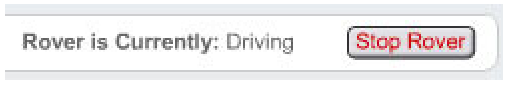
2.3.2 > Functionality & Interactions

The Time Until Next Downlink text starts out at 2:00 (two minutes), with the progress bar 'full' (completely covering up the light grey background behind it). The time ticks down backwards, and the progress bar gets smaller relative to how much time is left. At the end of two minutes, the indicator resets to 2:00, with the progress bar 'full', and begins it's countdown again.

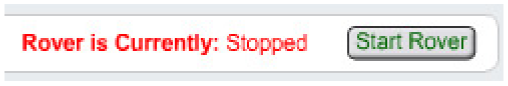
The Time Until Next Downlink Indicator is a purely system feedback widget. The user does not interact with it directly, as it is only meant to provide feedback.

2.4 > Rover Status

Active State:



Stopped State:



2.4.1 > Purpose

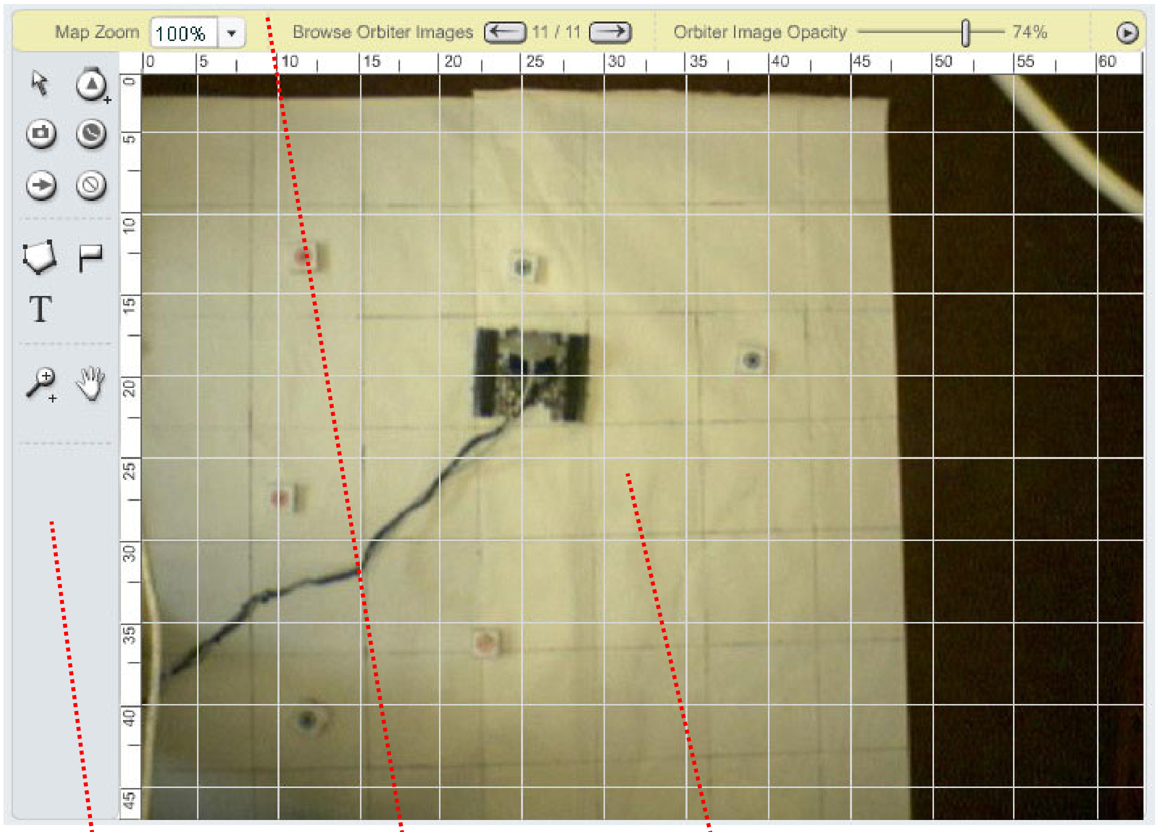
The Rover Status area gives the user feedback about what the system thinks the Rover is currently doing, as well as a way to start and stop the Rover.

2.4.2 > Functionality & Interactions

The text label tells the user what the system thinks the Rover is currently doing. If it is currently executing a drive, the 'Rover is Currently' text label reads 'Driving'. Other possibilities include 'Taking a Picture', 'Turning', 'Waiting', or 'Stopped'. If the Rover status is stopped, the text appears in red to draw the user's attention.

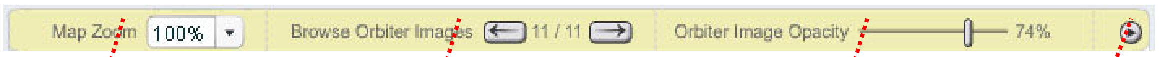
The button toggles between ‘Stop Rover’ and ‘Start Rover.’ If the Rover is currently active, the button reads ‘Stop Rover’, and pressing it will tell the Rover to stop executing the plan, but don’t delete the rest of the plan (like a Pause button on a CD player). If the Rover is currently stopped, the button reads ‘Start Rover’, and pressing it will tell the Rover to resume executing the plan (like a Play button on a CD player).

3 > Map



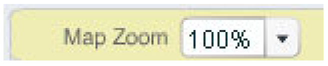
Map Toolbar Map Options Bar Map Main Area

3.1 > Map Options Bar



Map Zoom Orbiter Image Browser Orbiter Image Opacity
Map Options Button

3.1.1 > Map Zoom



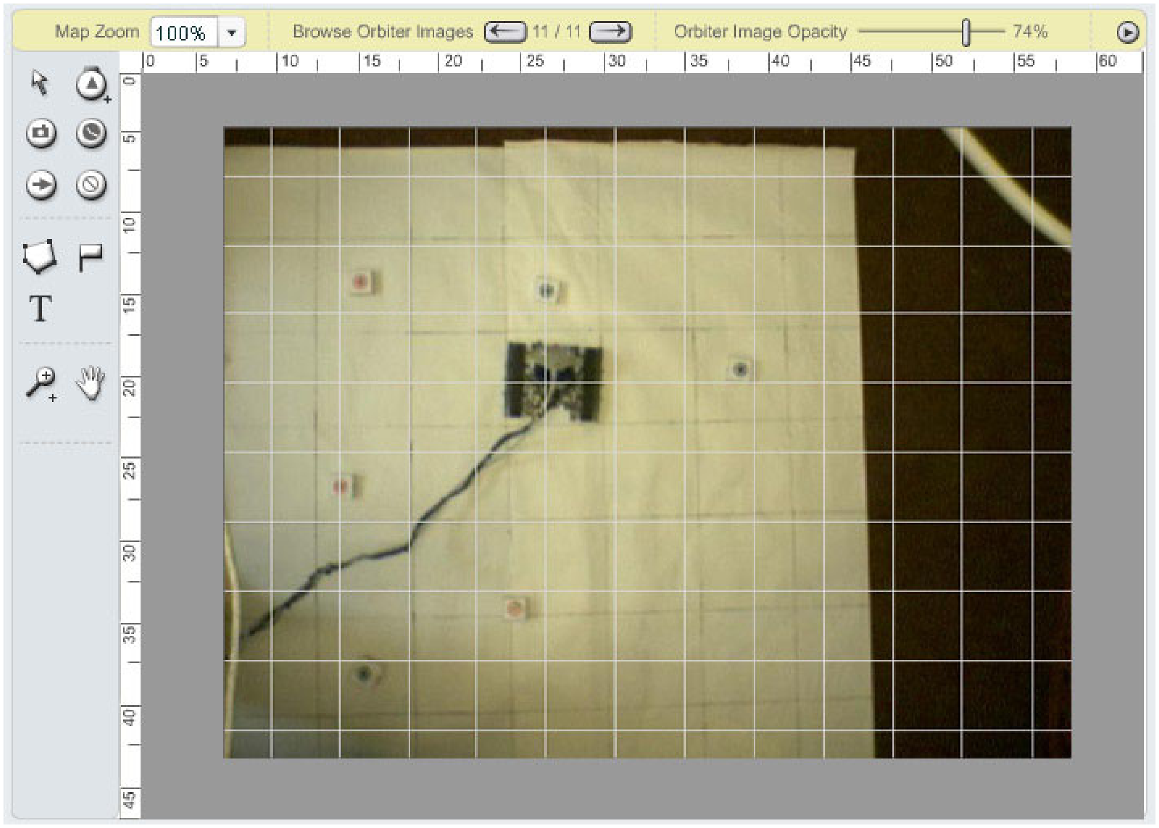
3.1.1.1 > Purpose

The Map Zoom tool allows the user to zoom in and out on the Map Main Area in order to establish the most appropriate map size for the current situation.

3.1.1.2 > Functionality & Interactions

The user clicks on the Map Zoom combo box and selects from one of the following options: 10%, 25%, 50%, 75%, 100%, 150%, 200%, 500%. The Map Main Area then zooms in or out based on the user's selection. If the Map Main Area Ruler is currently visible, the scale of the ruler adjusts accordingly, so that for example, when zooming in, the same number of pixels on the screen will represent less physical space, and the ruler scale will be 'stretched out.'

If the Zoom level is set to less than 100%, the Map is centered in the Main Map Area, and is surrounded by a grey background color (#999999), such as represented in the following screen shot:



3.1.2 > Orbiter Image Browser

3.1.2.1 > Purpose

The Orbiter Image Browser tool allows the user to quickly access past orbiter images. By quickly browsing through each of the orbiter images, the user should be able to establish a solid mental model of the Rover's actual traversed path.

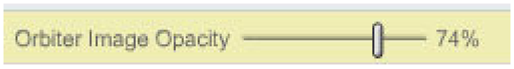
3.1.2.2 > Functionality & Interactions

At each downlink, the new orbiter image received from the orbiter replaces the current orbiter image in the Map Main Area. The orbiter image number feedback increases by one, so that if it previously read "7 / 7", at the next downlink it would read "8 / 8".

Clicking the left arrow button changes the orbiter image in the Map Main Area to the previously acquired orbiter image, and the orbiter image number feedback changes accordingly (so instead of reading " 8/ 8" it would read "7/ 8"). If the current orbiter image is the first one that was taken, the left arrow button is grayed out and not clickable.

Clicking the right arrow button changes the orbiter image in the Map Main Area to the next acquired orbiter image, and the orbiter image number feedback changes accordingly (so instead of reading " 7/ 8" it would read "8/ 8"). If the current orbiter image is the last one that was taken, the right arrow button is grayed out and not clickable.

3.1.3 > Orbiter Image Opacity



3.1.3.1 > Purpose

The Orbiter Image Opacity tool allows the user to specify the opacity of the currently displayed orbiter image in the Map Main Area. The orbiter image at 100% opacity may be too visually powerful and make it difficult to read and understand the other elements on the map, and so the user may want to lessen it's opacity. This would still give the user the awareness provided by the orbiter image, but without it being so dominant that it makes other elements difficult to discern.

3.1.3.2 > Functionality & Interactions

The user slides the slider widget (by clicking, holding, and dragging) to the left to increase the orbiter image's opacity and to the left the decrease the orbiter image's opacity. The orbiter image's opacity changes in real-time, as the user is sliding the bar. The scale goes from 0% to 100%. The number to the right of the slider widget gives the user real-time feedback as to the current orbiter image opacity.

3.1.4 > Map Options Button

3.1.4.1 > Purpose

The Map Options button allows the user to specify various options that affect the appearance of the Map Main Area.

3.1.4.2 > Functionality & Interactions

Clicking the Map Options Button displays the Map Options Menu. The right edge of the menu is aligned with the right-edge of the Map Options Button and the top edge of the menu is aligned with the bottom edge of the Map Options Button. Clicking the Map Options Button when the Map Options Menu is currently displayed hides the Map Options Menu. The Map Options Menu contains the following items (note the position of the menu dividers):

Hide Orbiter Image
Hide Grid
Hide Ruler
Hide Scale
Hide Downlink Points
Hide Text Notes
Hide Targets
Hide Worksites

Hide All Past
Hide Past Path
Hide Past Waypoints
Hide Past Observations
Hide Past Options

Hide All Future
Hide Future Path
Hide Future Waypoints
Hide Future Observations
Hide Future Options

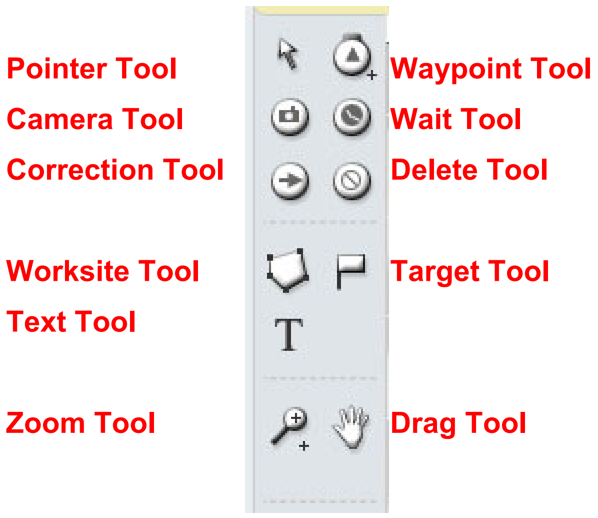
Grid Size...

All of the menu items with “Hide” as the first word act as visibility toggles. For example, clicking “Hide Past Waypoints” will hide the past waypoints on the Map Main Area, and the menu will disappear. Clicking on the Menu Options Button again would display the menu again, and the user would see “Show Past Waypoints”, and clicking this will display the past waypoints on the Map Main Area again. The same principle applies to the other “Hide/ Show” menu items.

There are two related accelerator menu items: “Hide All Past” and “Hide All Future.” These menu items act as shortcuts for toggling the visibility of all past or future symbols on the Map Main Area. For example, clicking “Hide All Past” will hide all past symbols on the Map Main Area, and the menu will disappear. Clicking on the Menu Options Button again would display the menu again, and the user would see “Show All Past”, “Show Past Waypoints”, “Show Past Path”, “Show Past Waypoints”, “Show Past Observations”, and “Show Past Options”.

Clicking the “Grid Size...” menu item would bring up a dialog box and make the Map Options Menu disappear. The Grid Size Dialog Box contains up and down arrow widgets for increasing or decreasing the grid size, and two feedback widgets: a textual and visual description of the grid size. The details of the grid size scale and options, and the exact layout of the Grid Size Dialog Box, are to be designed for the next version of this Specification.

3.2 > Map Toolbar



3.2.1 > Pointer Tool

3.2.1.1 > Purpose

The Pointer Tool allows the user to manipulate –move, rotate, select, delete, etc. - existing objects in the Main Map Area.

3.2.1.2 > Functionality & Interactions

Visual States

The Pointer Tool has three visual states in the Map Toolbar:

On (displayed when the Pointer Tool is the currently selected tool)

Off (displayed when the Pointer Tool is not the currently selected tool)

Over (displayed when the Pointer Tool is not the currently selected tool and the mouse is over the Pointer Tool icon in the Map Toolbar)

Tool tip

When the mouse is over the Pointer Tool icon in the Map Toolbar, a tool tip is displayed with the text “Selection Tool”.

Cursor

When the Pointer Tool is the currently selected tool and the cursor is hovered over the Map Main Area, the cursor changes to the following custom cursors:

Move Mode (the cursor is over the moveable area of a Waypoint or Observation Icon)

Rotate Mode (the cursor is over the rotatable area of a Waypoint or Observation Icon)

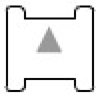
Default Mode (the cursor is not over a moveable or rotatable area of a Waypoint or Observation Icon)

Interactions with Waypoint and Observation Icons

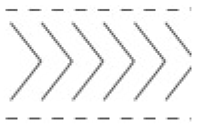
Waypoint and Observation Icons – Tool Tip
Moving the mouse over a Waypoint or Observation Icon displays a tool tip with:

1. The Name of the Waypoint or Observation
2. The Start Time of the Waypoint or Observation
3. The Duration of the Waypoint or Observation

Waypoint and Observation Icons – Move Mode
When the cursor is over the moveable area of a Waypoint or Observation Icon, the cursor changes to Move Mode. If the user clicks, holds, and drags, the Waypoint or Observation Icon will move with the cursor until the user releases the mouse button. While a Waypoint or Observation Icon is being moved it changes appearance to the Selected Waypoint or Observation Icon (which is representative of the Rover's actual size):



and the drive paths immediately before and after the moving icon change appearance to the Live Path (which is representative of the width of the Rover's actual drive path):

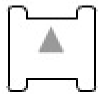


The Activity Planner automatically scrolls to the selected Waypoint or Observation, so that the selected Waypoint or Observation is vertically located around the center of the Activity Planner. The selected Waypoint or Observation is automatically selected in the Activity Planner.

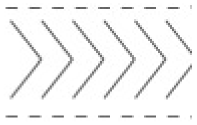
When the user releases the mouse button, the Waypoint or Observation Icon is still selected.

Waypoint and Observation Icons – Rotate Mode

When the cursor is over the rotatable area of a Waypoint or Observation Icon, the cursor changes to Rotate Mode. If the user clicks, holds, and drags, the Waypoint or Observation Icon (as well as the Rotate Mode cursor) will in real-time re-orient itself based on the cursor's relative position to the waypoint, until the user releases the mouse button. While a Waypoint or Observation Icon is being rotated it changes appearance to the Selected Waypoint or Observation Icon (which is representative of the Rover's actual size):



and the drive paths immediately before and after the moving icon change appearance to the Live Path (which is representative of the width of the Rover's actual drive path):

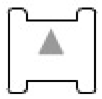


The Activity Planner automatically scrolls to the selected Waypoint or Observation, so that the selected Waypoint or Observation is vertically located around the center of the Activity Planner. The selected Waypoint or Observation is automatically selected in the Activity Planner.

When the user releases the mouse button, the Waypoint or Observation Icon is still selected.

Waypoint and Observation Icons –Single Selection Mode

After a user has either moved or rotated (or just clicked and quickly released) a Waypoint or Observation Icon, that icon is then selected and changes appearance to the Selected Waypoint or Observation Icon (which is representative of the Rover's actual size):



When a Waypoint or Observation Icon is in Single Selection Mode, but is not being dragged or rotated, the Live Path is not displayed.

The Activity Planner automatically scrolls to the selected Waypoint or Observation, so that the selected Waypoint or Observation is vertically located around the center of the Activity Planner. The selected Waypoint or Observation is automatically selected in the Activity Planner.

Waypoint and Observation Icons –Multiple Selection Mode

The user can perform the standard Shift-Click and Ctrl-Click in order to select multiple Waypoints or Observations. Nothing changes in the Activity Planner upon a multi-select.

Waypoint and Observation Icons –Single Selection Context Menu
Right-clicking a Waypoint or Observation Icon produces the following context menu:

Delete (Waypoint or Observation)
Delete Entire Plan After This Point
Add >
- Photo
- Wait
- Correction

Selecting either of the Delete menu items produces a confirmation dialog box. Upon confirmation, the appropriate delete action occurs.

Selecting a sub-menu item beneath 'Add' adds an Activity to a Waypoint (making it an Observation) or an existing Observation. The Activity Planner automatically scrolls to the selected Waypoint or Observation, so that the selected Waypoint or Observation is vertically located around the center of the Activity Planner. The selected Waypoint or Observation is automatically selected in the Activity Planner, and the selected Activity is added after the last existing Activity within that Observation. The newly added Activity is automatically in Edit Mode so that the user can quickly specify the Activity's parameters.

Waypoint and Observation Icons –Multiple Selection Context Menu
Right-clicking a Waypoint or Observation Icon produces the following context menu:

Delete

Selecting the Delete menu item produces a confirmation dialog box. Upon confirmation, the appropriate delete action occurs.

Waypoint and Observation Icons –Deselect
To deselect an Icon or Observation, the user clicks somewhere else in the Map Main Area, or else selects a different tool from the Map Toolbar.

Interactions with Downlink Icons

Downlink Icons – Tool Tip

Moving the mouse over a Downlink Icon displays a tool tip with:

1. The Downlink time

Interactions – Other

Effect on the Map Ruler

As the user moves the cursor around the Map Main Area, a faint but visible dotted line appears on the Map Ruler at the exact position of the cursor, on both the horizontal and vertical axis. This helps the user gain a solid understanding of exactly where they have their cursor positioned.

3.2.2 > Waypoint Tool

3.2.2.1 > Purpose

The Waypoint Tool allows the user to add Waypoints and Optional Waypoints to the Map Main Area.

3.2.2.2 > Functionality & Interactions

Visual States

The Waypoint Tool has three visual states in the Map Toolbar:

On (displayed when the Waypoint Tool is the currently selected tool)

Off (displayed when the Waypoint Tool is not the currently selected tool)

Over (displayed when the Waypoint Tool is not the currently selected tool and the mouse is over the Waypoint Tool icon in the Map Toolbar)

Expanded States
When the user clicks and holds the Waypoint Tool in the Map Toolbar, sub-tools are displayed for the user to choose from (by clicking). There are four total Waypoint Tools to choose from:

Waypoint Forward

Waypoint Reverse

Waypoint Option Forward

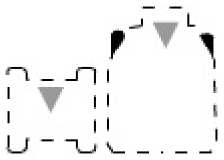
Waypoint Option Reverse

Tool tip
When the mouse is over the Waypoint Tool icon in the Map Toolbar, a tool tip is displayed with the appropriate text (based on which sub-tool is currently selected): “Waypoint Forward Tool”, “Waypoint Reverse Tool”, “Optional Waypoint Forward Tool”, “Optional Waypoint Reverse Tool”.

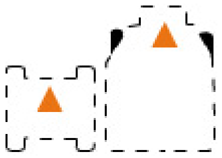
Cursor
When the Waypoint Tool is the currently selected tool and the cursor is hovered over the Map Main Area, the cursor changes to the following custom cursors (based on which sub-tool is currently selected):

Waypoint Forward – Mindstorm and Per

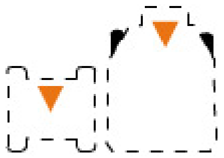
Waypoint Reverse – Mindstorm and Per



Waypoint Option Forward – Mindstorm and Per



Waypoint Option Reverse – Mindstorm and Per

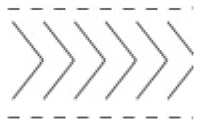


Interactions

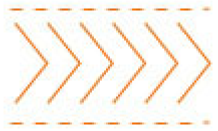
Placing a New Waypoint After the Last Planted Waypoint or Observation

As the user moves the cursor around the Map Main Area, a Live Path (which is representative of the width of the Rover's actual drive path) is drawn in real-time from the last Planted Waypoint or Observation Icon to the custom cursor. The Live Path appears as follows:

Live Path



Live Path Option Mode



Both the last Planted Waypoint or Observation Icon and the custom cursor rotate in real time to match the direction from the last Planted Waypoint or Observation Icon to the custom cursor.

The user clicks to confirm the placement of the Waypoint. A Planted Waypoint Icon appears below the cursor, and the appropriate Planted Path (regular or optional) is displayed from the last Planted Waypoint or Observation Icon to the newly Planted Waypoint Icon. The newly Planted Waypoint Icon then becomes the last Planted Waypoint Icon, and the Live Path is then drawn from that Planted Waypoint Icon to the custom cursor.

Downlinks

As the user drags the cursor around the screen, Downlink Icons are automatically placed along the Live Path, at the points where the system estimates Downlinks will occur (based on the length of the Live Path). Since the Live Paths are sized to represented the Rover's actual drive path, the Downlink Icons should automatically re-size to cover the entire width of the Live Path.

Restrictions

A Waypoint placed on the Map must be placed at a distance equal to or greater than the size of the custom cursor away from the last Planted Waypoint or Observation Icon. Essentially, the Map is used only for "drives". Small movements, or Corrections, are handled in the Activity Planner.

Reverse

When the user plants a Reverse Waypoint (or Optional Reverse Waypoint), the last Planted Waypoint or Observation Icon automatically reverses its direction. The assumption is that the user intended for the Rover to drive in reverse from the last Planted Waypoint or Observation to the newly Planted Waypoint.

Placing a New Waypoint Along a Planted Path

Users can also place a Waypoint along a Planted Path. When the user moves the cursor within a reasonable range of a Planted Path and stops moving the mouse for a given period of time (around a second or two), the customer cursor snaps to the Planted Path. It's orientation – as well as the orientation of the Planted Path's starting Planted Waypoint or Observation Icon – are automatically adjusted. If the user moves the mouse slowly, the custom cursor remains attached to the Planted Path. The Planted Path is essentially split into two, divided by the custom cursor. The two Planted Paths (one before the custom cursor and one after) are displayed as Live Paths.

If the user moves the mouse quickly, the custom cursor detaches itself from the Planted Path, and operates as described in ‘*Placing a New Waypoint After the Last Planted Waypoint or Observation*’ above.

Optional Waypoints can only be attached to Optional Planted Paths, and Regular Waypoints can only be attached to Regular Planted Paths.

Effect on the Map Ruler
As the user moves the cursor around the Map Main Area, a faint but visible dotted line appears on the Map Ruler at the exact position of the cursor, on both the horizontal and vertical axis. This helps the user gain a solid understanding of exactly where they have their cursor positioned.

3.2.3 > Camera Tool

3.2.3.1 > Purpose
The Camera Tool allows the user to add a Photo Activity to Planted Waypoints or Observations.

3.2.3.2 > Functionality & Interactions

Visual States
The Camera Tool has three visual states in the Map Toolbar:

On (displayed when the Camera Tool is the currently selected tool)

Off (displayed when the Camera Tool is not the currently selected tool)

Over (displayed when the Camera Tool is not the currently selected tool and the mouse is over the Camera Tool icon in the Map Toolbar)

Tool tip

When the mouse is over the Camera Tool icon in the Map Toolbar, a tool tip is displayed with the text “Camera Tool”.

Cursor
When the Camera Tool is the currently selected tool and the cursor is hovered over the Map Main Area, the cursor changes to the following custom cursor:

Interactions with Waypoint and Observation Icons
When the user moves the cursor within an acceptable range of a Planted Waypoint or Observation Icon, the Planted Waypoint or Observation Icon grows to 125% of its normal size. When the user moves the cursor away from the icon, the icon shrinks back to it’s original size. The user clicks in order to add a Photo Activity to the Waypoint or Observation. The following events occur:

- The Activity Planner automatically scrolls to the selected Waypoint or Observation, so that the selected Waypoint or Observation is vertically located around the center of the Activity Planner.
- The selected Waypoint or Observation is automatically selected in the Activity Planner.
- If the user clicked on a Waypoint, that Waypoint is automatically turned into an Observation, both in the Map Main Area and the Activity Planner.
- A Photo Activity is automatically added to the end of the Observation’s Activities.
- The newly added Photo Activity is automatically placed in Edit mode so that the user can specify the appropriate parameters.
- The selected Waypoint or Observation is placed in Selected Mode (see 3.2.1.2 above for details).

Clicking when the cursor is not within an acceptable range of a Planted Waypoint or Observation Icon has no effect.

Effect on the Map Ruler
As the user moves the cursor around the Map Main Area, a faint but visible dotted line appears on the Map Ruler at the exact position of the cursor, on both the horizontal and vertical axis. This helps the user gain a solid understanding of exactly where they have their cursor positioned.

3.2.4 > Wait Tool

3.2.4.1 > Purpose

The Wait Tool allows the user to add a Wait Activity to Planted Waypoints or Observations.

3.2.4.2 > Functionality & Interactions

Visual States

The Wait Tool has three visual states in the Map Toolbar:

On (displayed when the Wait Tool is the currently selected tool)

Off (displayed when the Wait Tool is not the currently selected tool)

Over (displayed when the Wait Tool is not the currently selected tool and the mouse is over the Wait Tool icon in the Map Toolbar)

Tool tip

When the mouse is over the Wait Tool icon in the Map Toolbar, a tool tip is displayed with the text "Wait Tool".

Cursor

When the Wait Tool is the currently selected tool and the cursor is hovered over the Map Main Area, the cursor changes to the following custom cursor:

Interactions with Waypoint and Observation Icons

When the user moves the cursor within an acceptable range of a Planted Waypoint or Observation Icon, the Planted Waypoint or Observation Icon grows to 125% of its normal size. When the user moves the cursor away from the icon, the icon shrinks back to it's original size.

The user clicks in order to add a Wait Activity to the Waypoint or Observation. The following events occur:

- The Activity Planner automatically scrolls to the selected Waypoint or Observation, so that the selected Waypoint or Observation is vertically located around the center of the Activity Planner.
- The selected Waypoint or Observation is automatically selected in the Activity Planner.
- If the user clicked on a Waypoint, that Waypoint is automatically turned into an Observation, both in the Map Main Area and the Activity Planner.
- A Wait Activity is automatically added to the end of the Observation's Activities.
- The newly added Wait Activity is automatically placed in Edit mode so that the user can specify the appropriate parameters.
- The selected Waypoint or Observation is placed in Selected Mode (see 3.2.1.2 above for details).

Clicking when the cursor is not within an acceptable range of a Planted Waypoint or Observation Icon has no effect.

Effect on the Map Ruler
As the user moves the cursor around the Map Main Area, a faint but visible dotted line appears on the Map Ruler at the exact position of the cursor, on both the horizontal and vertical axis. This helps the user gain a solid understanding of exactly where they have their cursor positioned.

3.2.5 > Correction Tool

3.2.5.1 > Purpose
The Correction Tool allows the user to add a Correction Activity to Planted Waypoints or Observations.

3.2.5.2 > Functionality & Interactions

Visual States
The Correction Tool has three visual states in the Map Toolbar:

On (displayed when the Correction Tool is the currently selected tool)

Off (displayed when the Correction Tool is not the currently selected tool)

Over (displayed when the Correction Tool is not the currently selected tool and the mouse is over the Correction Tool icon in the Map Toolbar)

Tool tip
When the mouse is over the Correction Tool icon in the Map Toolbar, a tool tip is displayed with the text "Correction Tool".

Cursor
When the Correction Tool is the currently selected tool and the cursor is hovered over the Map Main Area, the cursor changes to the following custom cursor:

Interactions with Waypoint and Observation Icons
When the user moves the cursor within an acceptable range of a Planted Waypoint or Observation Icon, the Planted Waypoint or Observation Icon grows to 125% of its normal size. When the user moves the cursor away from the icon, the icon shrinks back to it's original size. The user clicks in order to add a Wait Activity to the Waypoint or Observation. The following events occur:

- The Activity Planner automatically scrolls to the selected Waypoint or Observation, so that the selected Waypoint or Observation is vertically located around the center of the Activity Planner.
- The selected Waypoint or Observation is automatically selected in the Activity Planner.
- If the user clicked on a Waypoint, that Waypoint is automatically turned into an Observation, both in the Map Main Area and the Activity Planner.
- A Correction Activity is automatically added to the end of the Observation's Activities.
- The newly added Correction Activity is automatically placed in Edit mode so that the user can specify the appropriate parameters.

- The selected Waypoint or Observation is placed in Selected Mode (see 3.2.1.2 above for details).

Clicking when the cursor is not within an acceptable range of a Planted Waypoint or Observation Icon has no effect.

Effect on the Map Ruler
As the user moves the cursor around the Map Main Area, a faint but visible dotted line appears on the Map Ruler at the exact position of the cursor, on both the horizontal and vertical axis. This helps the user gain a solid understanding of exactly where they have their cursor positioned.

3.2.6 > Delete Tool

3.2.6.1 > Purpose
The Delete Tool allows the user to delete Planted Waypoints or Observations.

3.2.6.2 > Functionality & Interactions

Visual States
The Delete Tool has three visual states in the Map Toolbar:

On (displayed when the Delete Tool is the currently selected tool)

Off (displayed when the Delete Tool is not the currently selected tool)

Over (displayed when the Delete Tool is not the currently selected tool and the mouse is over the Delete Tool icon in the Map Toolbar)

Tool tip

When the mouse is over the Delete Tool icon in the Map Toolbar, a tool tip is displayed with the text “Delete Tool”.

Cursor
When the Delete Tool is the currently selected tool and the cursor is hovered over the Map Main Area, the cursor changes to the following custom cursor:

Interactions with Waypoint and Observation Icons
When the user moves the cursor within an acceptable range of a Planted Waypoint or Observation Icon, the Planted Waypoint or Observation Icon grows to 125% of its normal size. When the user moves the cursor away from the icon, the icon shrinks back to it’s original size. The user clicks in order to delete the Waypoint or Observation. They are first shown a confirmation dialog box before the delete action occurs. Deleting a Waypoint or Observation causes a new path to be drawn from the Waypoint or Observation directly before the deleted icon to the Waypoint or Observation directly after the deleted icon (with obvious differences when deleting the first or the last icon).

Clicking when the cursor is not within an acceptable range of a Planted Waypoint or Observation Icon has no effect.

Effect on the Map Ruler
As the user moves the cursor around the Map Main Area, a faint but visible dotted line appears on the Map Ruler at the exact position of the cursor, on both the horizontal and vertical axis. This helps the user gain a solid understanding of exactly where they have their cursor positioned.

3.2.7 > Worksite Tool

3.2.7.1 > Purpose
The Worksite Tool allows the user to visually define Worksites on the Map Main Area. In our limited domain, the Worksite serves mainly as an organizational and communication aid, and has no programmatic significance. However, for more advanced and autonomous robotic platforms, such as K9, the Worksite could serve as way to define areas that the user could directly command the Rover to explore.

3.2.7.2 > Functionality & Interactions

Visual States

The Worksite Tool has three visual states in the Map Toolbar:

On (displayed when the Worksite Tool is the currently selected tool)

Off (displayed when the Worksite Tool is not the currently selected tool)

Over (displayed when the Worksite Tool is not the currently selected tool and the mouse is over the Worksite Tool icon in the Map Toolbar)

Tool tip

When the mouse is over the Worksite Tool icon in the Map Toolbar, a tool tip is displayed with the text “Worksite Tool”.

Cursor

When the Worksite Tool is the currently selected tool and the cursor is hovered over the Map Main Area, the cursor changes to the following custom cursor:

Interactions

To be determined in a future version of this Specification.

Effect on the Map Ruler

As the user moves the cursor around the Map Main Area, a faint but visible dotted line appears on the Map Ruler at the exact position of the cursor, on both the horizontal and vertical axis. This helps the user gain a solid understanding of exactly where they have their cursor positioned.

3.2.8 > Target Tool

3.2.8.1 > Purpose

The Target Tool allows the user to visually define Targets on the Map Main Area. In our limited domain, the Target serves mainly as an organizational and communication aid, and has no programmatic significance. However, for more advanced and autonomous robotic platforms, such as K9, the Target could serve as way to define specific targets that the user could directly command the Rover to explore.

3.2.8.2 > Functionality & Interactions

Visual States

The Target Tool has three visual states in the Map Toolbar:

On (displayed when the Target Tool is the currently selected tool)

Off (displayed when the Target Tool is not the currently selected tool)

Over (displayed when the Target Tool is not the currently selected tool and the mouse is over the Target Tool icon in the Map Toolbar)

Tool tip

When the mouse is over the Target Tool icon in the Map Toolbar, a tool tip is displayed with the text “Target Tool”.

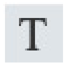
Cursor

When the Target Tool is the currently selected tool and the cursor is hovered over the Map Main Area, the cursor changes to the following custom cursor:

Interactions
To be determined in a future version of this Specification.

Effect on the Map Ruler
As the user moves the cursor around the Map Main Area, a faint but visible dotted line appears on the Map Ruler at the exact position of the cursor, on both the horizontal and vertical axis. This helps the user gain a solid understanding of exactly where they have their cursor positioned.

3.2.9 > Text Tool

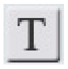


3.2.9.1 > Purpose
The Text Tool allows the user to add textual notes to the Map Main Area. For our specific task, a key use of these text notes is to annotate the Map with the discovered point values of the target sides.

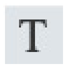
3.2.9.2 > Functionality & Interactions

Visual States
The Text Tool has three visual states in the Map Toolbar:

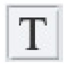
On (displayed when the Text Tool is the currently selected tool)



Off (displayed when the Text Tool is not the currently selected tool)



Over (displayed when the Text Tool is not the currently selected tool and the mouse is over the Text Tool icon in the Map Toolbar)



Tool tip

When the mouse is over the Text Tool icon in the Map Toolbar, a tool tip is displayed with the text "Text Tool".

Cursor
When the Text Tool is the currently selected tool and the cursor is hovered over the Map Main Area, the cursor changes to the following custom cursor:

Interactions
Adding Text
The user clicks anywhere on the Map Main Area and a standard blinking text cursor appears in the spot where they clicked. The user can then type freely. The text will extend horizontally unless the user presses the Enter key, which acts as a carriage return.

To exit from this mode, the user presses the Escape key, or else activates another Tool on the Map Toolbar.

Editing Text
To edit existing text, the user places the cursor over existing text and clicks. A standard blinking text cursor appears within the text string where the user clicked. The text string is now in Edit mode, and the system behaves in the same way as described above in *Adding Text*. To delete text, the user simply deletes all the characters in a text string.

Effect on the Map Ruler
As the user moves the cursor around the Map Main Area, a faint but visible dotted line appears on the Map Ruler at the exact position of the cursor, on both the horizontal and vertical axis. This helps the user gain a solid understanding of exactly where they have their cursor positioned.

3.2.10 > Zoom Tool

3.2.10.1 > Purpose
The Zoom Tool allows the user to zoom in and out on the Map Main Area in order to establish the most appropriate map size for the current situation.

3.2.10.2 > Functionality & Interactions

Visual States

The Zoom Tool has three visual states in the Map Toolbar:

On (displayed when the Zoom Tool is the currently selected tool)

Off (displayed when the Zoom Tool is not the currently selected tool)

Over (displayed when the Zoom Tool is not the currently selected tool and the mouse is over the Zoom Tool icon in the Map Toolbar)

Expanded States

When the user clicks and holds the Zoom Tool in the Map Toolbar, sub-tools are displayed for the user to choose from (by clicking). There are two total Zoom Tools to choose from:

Zoom In

Zoom Out

Tool tip

When the mouse is over the Zoom Tool icon in the Map Toolbar, a tool tip is displayed with the appropriate text (based on which sub-tool is currently selected): “Zoom In Tool”, “Zoom Out Tool”.

Cursor

When the Zoom Tool is the currently selected tool and the cursor is hovered over the Map Main Area, the cursor changes to the following custom cursors (based on which sub-tool is currently selected):

Zoom In

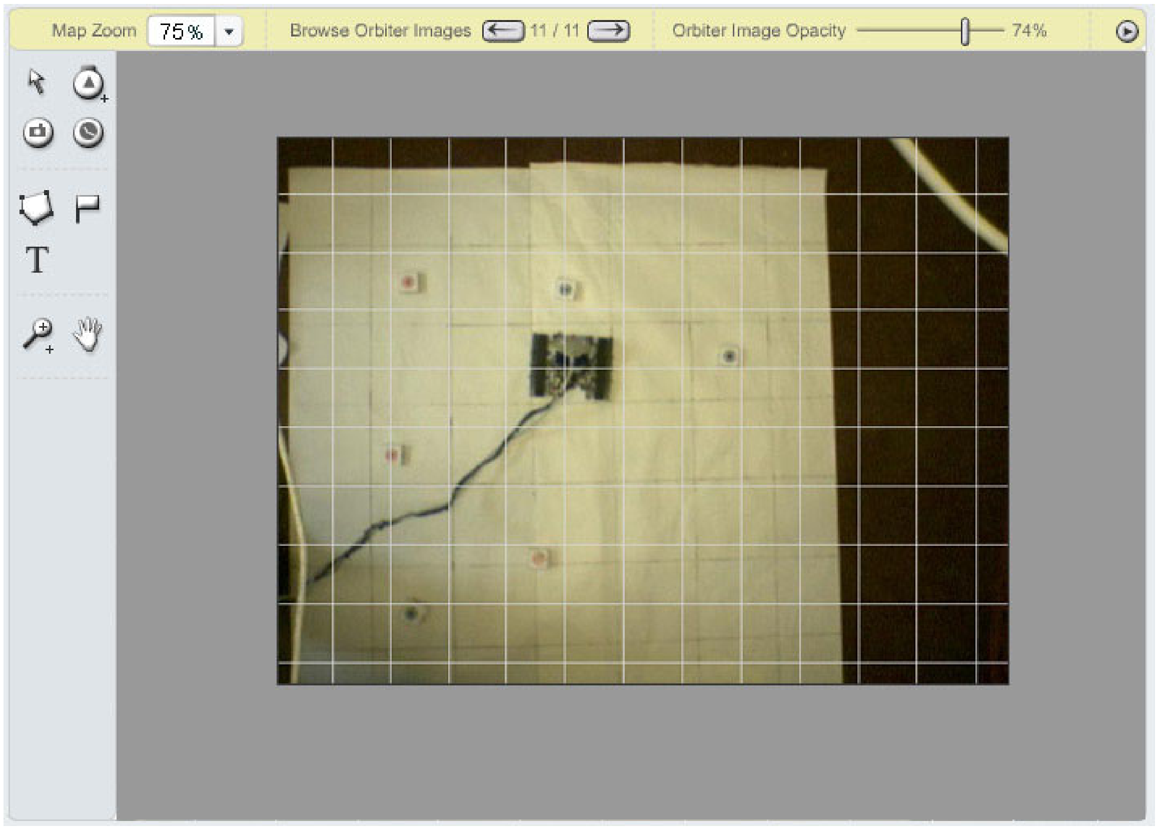
Zoom Out

Interactions

The user clicks anywhere on the Map Main Area, and the Map zooms in or out, depending on which sub-tool is selected.

If the Map Main Area Ruler is currently visible, the scale of the ruler adjusts accordingly, so that for example, when zooming in, the same number of pixels on the screen will represent less physical space, and the ruler scale will be ‘stretched out.’

If the Zoom level is set to less than 100%, the Map is centered in the Main Map Area, and is surrounded by a grey background color (#999999), such as represented in the following screen shot:



As the user zooms in or out with the Zoom Tool, the text in the Map Zoom tool in the Map Options Area changes to reflect the new zoom level.

Effect on the Map Ruler
As the user moves the cursor around the Map Main Area, a faint but visible dotted line appears on the Map Ruler at the exact position of the cursor, on both the horizontal and vertical axis. This helps the user gain a solid understanding of exactly where they have their cursor positioned.

3.2.11 > Drag Tool

3.2.11.1 > Purpose

The Drag Tool allows the user to drag around the visible Map Main Area when the Map is zoomed in more than 100%.

3.2.11.2 > Functionality & Interactions

Visual States
The Drag Tool has three visual states in the Map Toolbar:

On (displayed when the Drag Tool is the currently selected tool)

Off (displayed when the Drag Tool is not the currently selected tool)

Over (displayed when the Drag Tool is not the currently selected tool and the mouse is over the Drag Tool icon in the Map Toolbar)

Tool tip

When the mouse is over the Drag Tool icon in the Map Toolbar, a tool tip is displayed with the text “Drag Tool”.

Cursor

When the Drag Tool is the currently selected tool and the cursor is hovered over the Map Main Area, the cursor changes to the following custom cursor:

Interactions

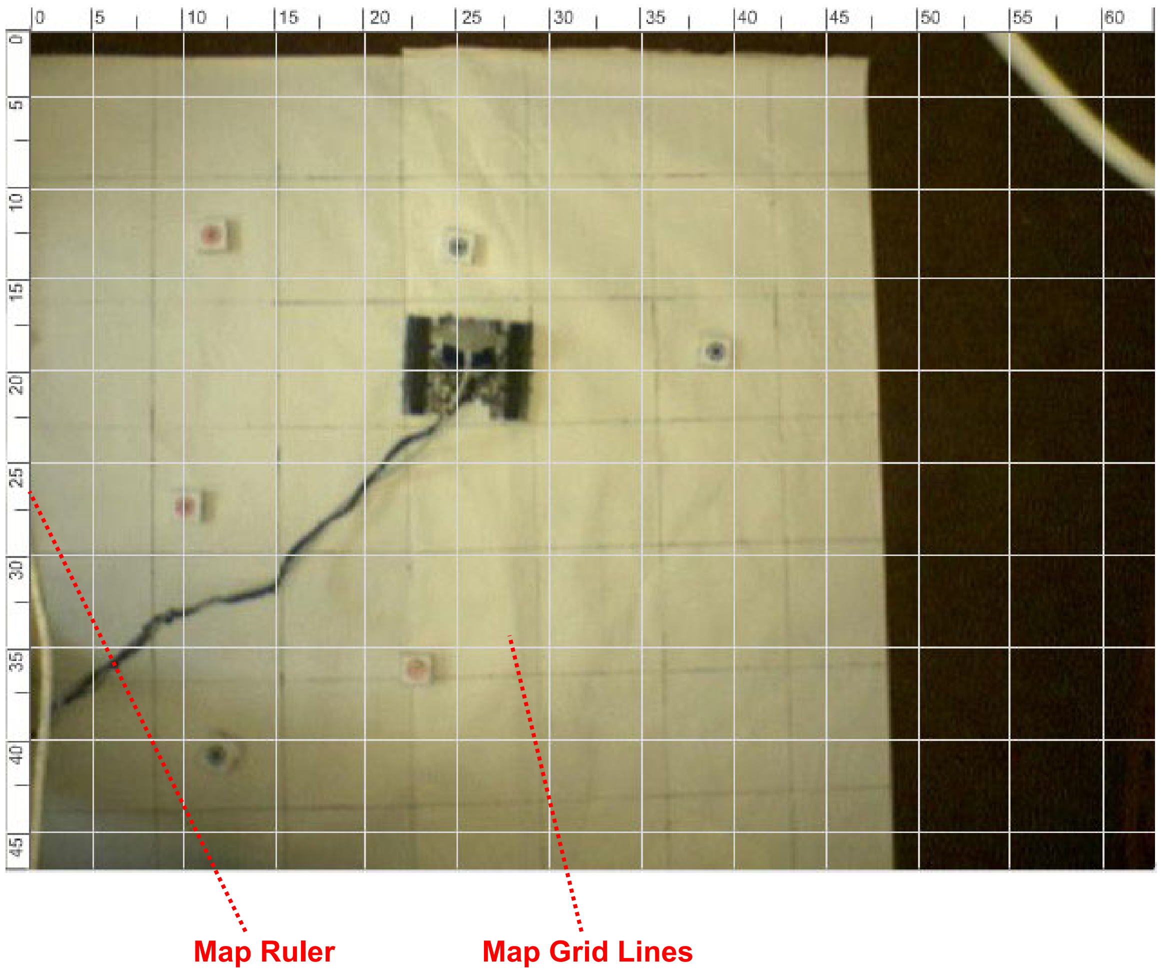
The user clicks, holds, and drags anywhere on the Map Main Area, and as they do so the entire Map moves with the cursor. The Ruler stays in place but the units adjust accordingly. This dragging action is ceased by releasing the mouse button.

The Drag Tool has no effect if the Map is not zoomed in more than 100%.

Effect on the Map Ruler

As the user moves the cursor around the Map Main Area, a faint but visible dotted line appears on the Map Ruler at the exact position of the cursor, on both the horizontal and vertical axis. This helps the user gain a solid understanding of exactly where they have their cursor positioned.

3.3 > Map Main Area



3.3.1 > Map Ruler

3.3.1.1 > Purpose

The Map Ruler gives the user an understanding of the scale being represented on the map. The tick marks on the Ruler should be aligned with the Map Grid Lines. The numbers on the Ruler should tell the user how far the corresponding area on the orbiter image is from an imaginary starting point in the upper left-hand corner. For example, if a tick mark is labeled as "40", the user would know that the corresponding area on the orbiter image is 40 Meters (or whatever unit is appropriate) away from the top of the playing field.

3.3.1.2 > Functionality & Interactions

The user mouses over the Ruler and a tool tip is displayed telling the user exactly where the cursor is located along the Ruler, including the units (for example: "41.56 M").

3.3.2 > Map Grid Lines

3.3.2.1 > Purpose

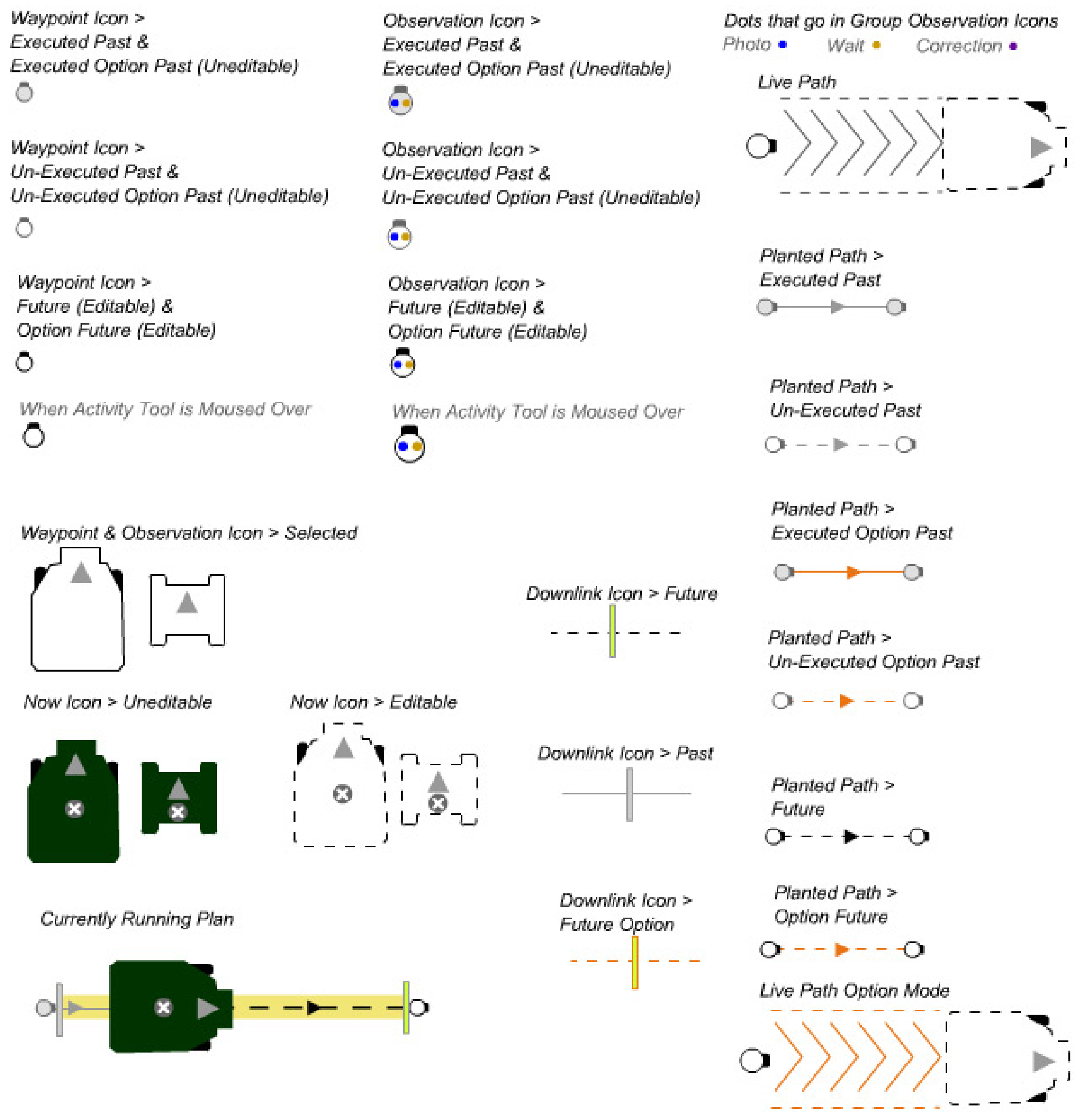
The Map Grid Lines provide a visual aid for the user to gain a solid understanding of the distances involved in the environment, and how those distances relate to each other.

3.3.2.2 > Functionality & Interactions

The user does not directly interact with the Map Grid Lines.

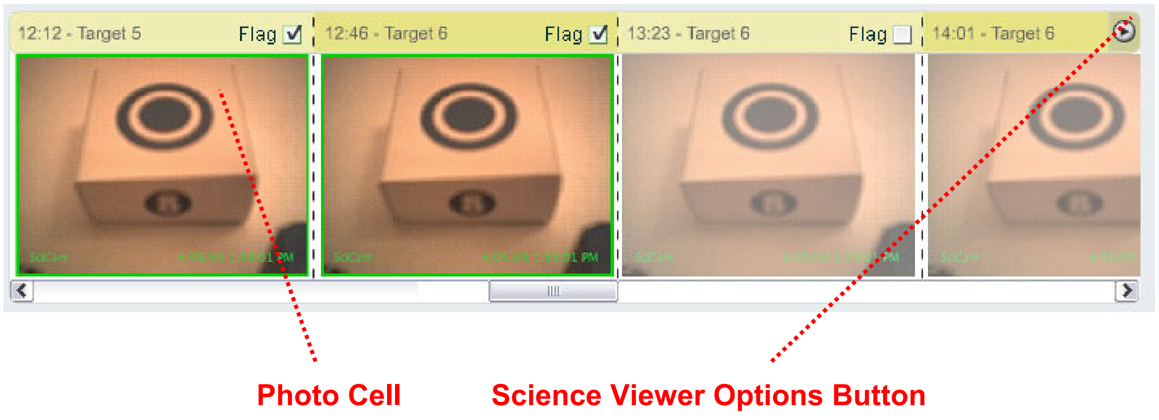
3.3.3 > Map Icons

The following is a visual definition of all icon types used on the Map:


Waypoint Icon > Executed Past & Executed Option Past (Uneditable)
Observation Icon > Executed Past & Executed Option Past (Uneditable)
Dots that go in Group Observation Icons
Photo
Wait
Correction
Live Path
Waypoint Icon > Un-Executed Past & Un-Executed Option Past (Uneditable)
Observation Icon > Un-Executed Past & Un-Executed Option Past (Uneditable)
Planted Path > Executed Past
Waypoint Icon > Future (Editable) & Option Future (Editable)
Observation Icon > Future (Editable) & Option Future (Editable)
Planted Path > Un-Executed Past
When Activity Tool is Moused Over
When Activity Tool is Moused Over
Planted Path > Executed Option Past
Waypoint & Observation Icon > Selected
Downlink Icon > Future
Planted Path > Un-Executed Option Past
Now Icon > Uneditable
Now Icon > Editable
Downlink Icon > Past
Planted Path > Future
Downlink Icon > Future Option
Planted Path > Option Future
Currently Running Plan
Live Path Option Mode

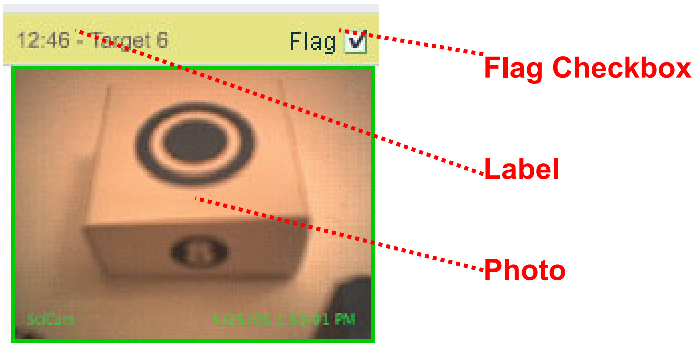
All the Icons above should be self-explanatory, but the Now Icon does warrant special mention. The Now Icon represents where the system thinks the Rover currently is along the planned path. As the Plan executes, the Now Icon animates in real-time along the planned path, and is not editable. At each Downlink point, the Uneditable Now Icon turns into the Editable Now Icon, so that the user can calibrate the system's model of ground truth (where the Rover currently is). This is explained further under the Activity Planner section in Calibration.

4 > Science Viewer



Note: the current design of the Science Viewer is geared towards our specific domain, in which Photos taken by the Rover's on-board camera are the only science data. It could, however, be easily modified to allow for the integration of other types of science data in future versions. Since our focus was not in viewing and organizing science data, we didn't fully explore the extensibility of this design.

4.1 > Photo Cell



4.1.1 > Purpose

The Photo Cell allows users to view the Photographs taken by the Rover's on-board camera, and to flag the photos they want to count towards their official score.

4.1.2 > Functionality & Interactions

Label

The Label at the top of the Photo Cell gives the user the time the photo was taken, and the name of the photo's parent Observation. If the parent Observation name is too long to fit in the Label area, the end of the parent Observation name is appended with ellipses. For example: "12:46 – Long Observation Na...". Mousing over the Label displays a tool tip with the entire parent Observation name.

The Label area alternates background color to make it easier to distinguish between each individual Photo Cell.

Flag Checkbox
The Flag Checkbox is used to flag the photos the user wants to count towards their official score. A checked Flag Checkbox signifies a Flagged Photo, and an unchecked Flag Checkbox signifies an Unflagged Photo.

Photo
A Flagged Photo is displayed at 100% opacity and with a thick green stroke around the photo. An Unflagged Photo is displayed at 65% opacity and with a thin light grey stroke.

Viewing Associated Observation Icon, Orbiter Image, and Activity
When the user mouses over anywhere on the Photo Cell, the Map and Activity Planner areas of the interface display information associated with that Photo:

- The Map displays the last Orbiter Image taken before the Photo was taken.
- The orbiter image number feedback in the Orbiter Image Browser displays the associated orbiter image number (such as "4 / 7").
- The Observation Icon in the Map Main Area associated with the Photo changes appearance to the Selected Waypoint or Observation Icon (which is representative of the Rover's actual size):

.

- The Activity Planner automatically scrolls to the associated Observation, so that the associated Observation is vertically located around the center of the Activity Planner. The associated Observation is automatically selected in the Activity Planner, and the associated Activity within that Observation is also selected.

When the user moves the mouse away from the Photo Cell, the Map and Activity Planner revert back to their previous states.

4.2 > Science Viewer Options Button

4.2.1 > Purpose

The Science Viewer Options button allows the user to specify various options that affect the appearance of the Science Viewer.

4.2.2 > Functionality & Interactions

Clicking the Science Viewer Button displays the Science Viewer Options Menu. The right edge of the menu is aligned with the right-edge of the Science Viewer Options Button and the bottom edge of the menu is aligned with the bottom edge of the Science Viewer Options Button. Clicking the Science Viewer Options Button when the Science Viewer Options Menu is currently displayed hides the Science Viewer Options Menu. The Science Viewer Options Menu contains the following items (note the position of the menu dividers):

Hide Flagged Photos
Hide Unflagged Photos

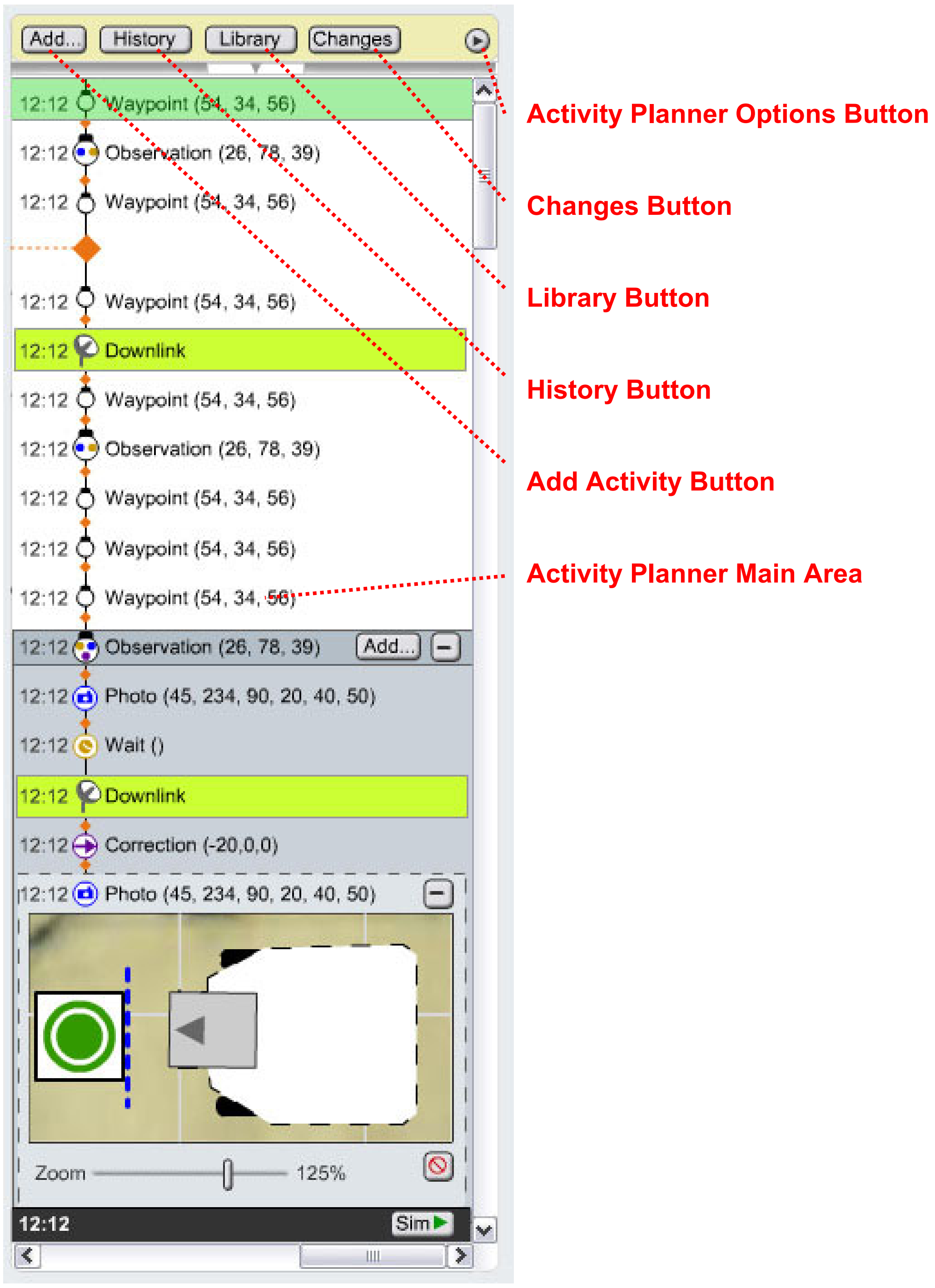
Photo Size >

- Large
- ✔ Medium
- Small

All of the menu items with “Hide” as the first word act as visibility toggles. For example, clicking “Hide Flagged Photos” will hide the Flagged Photos in the Science Viewer, and the menu will disappear. Clicking on the Menu Options Button again would display the menu again, and the user would see “Show Flagged Photos”, and clicking this will display the Flagged Photos in the Science Viewer again. The same principle applies to the other “Hide/ Show” menu items.

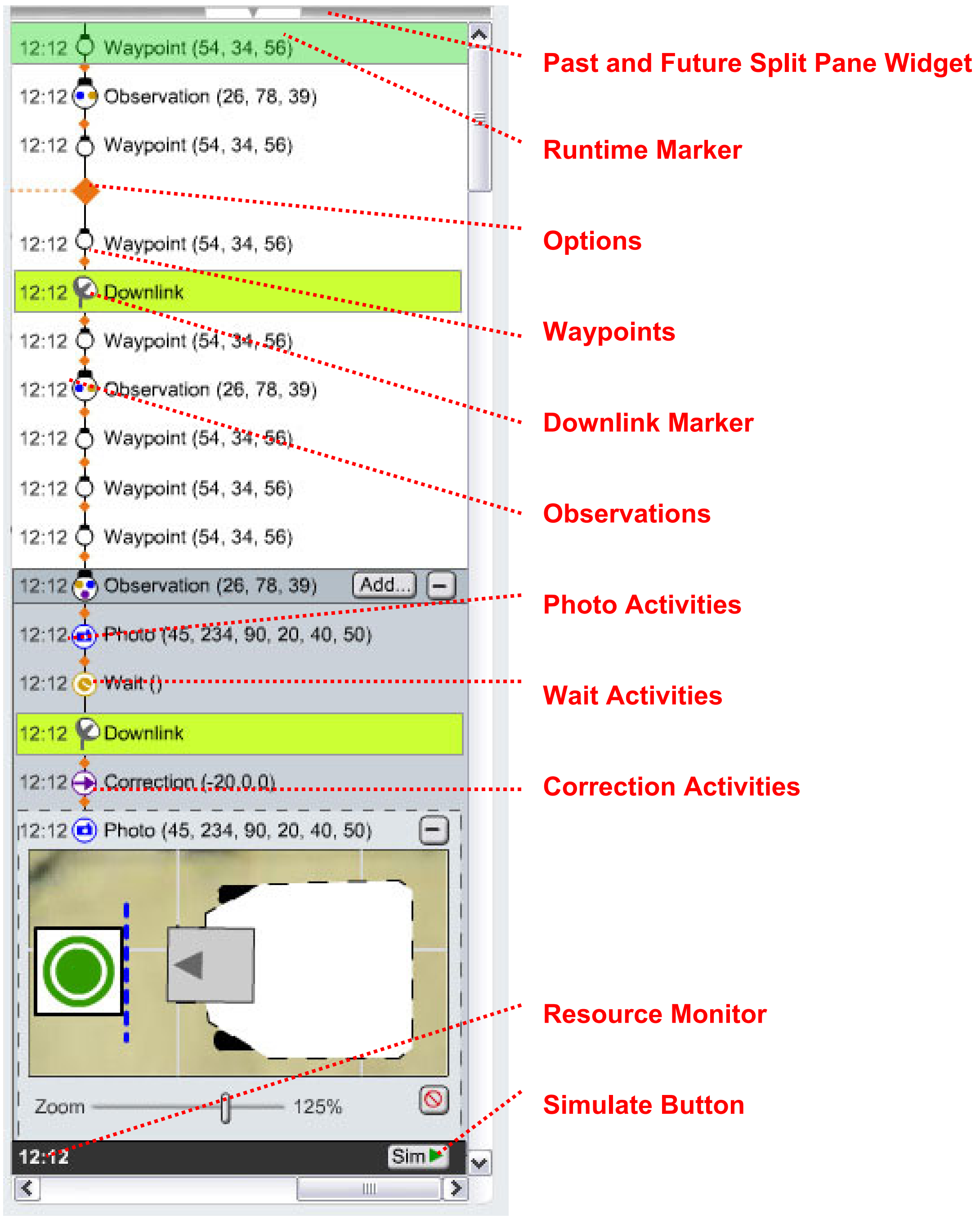
The Photo Size menu item and sub-menu items allow the user to specify the relative size of the photos as displayed in the Science Viewer.

5 > Activity Planner

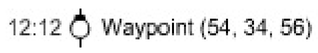


12:12 Waypoint (54, 34, 56)
12:12 Observation (26, 78, 39)
12:12 Waypoint (54, 34, 56)
12:12 Waypoint (54, 34, 56)
12:12 Downlink
12:12 Waypoint (54, 34, 56)
12:12 Observation (26, 78, 39)
12:12 Waypoint (54, 34, 56)
12:12 Waypoint (54, 34, 56)
12:12 Waypoint (54, 34, 56)
12:12 Observation (26, 78, 39)
12:12 Photo (45, 234, 90, 20, 40, 50)
12:12 Wait ()
12:12 Downlink
12:12 Correction (-20,0,0)
12:12 Photo (45, 234, 90, 20, 40, 50)
12:12

5.1 > Activity Planner Main Area


12:12 Waypoint (54, 34, 56)
12:12 Observation (26, 78, 39)
12:12 Waypoint (54, 34, 56)
12:12 Waypoint (54, 34, 56)
12:12 Downlink
12:12 Waypoint (54, 34, 56)
12:12 Observation (26, 78, 39)
12:12 Waypoint (54, 34, 56)
12:12 Waypoint (54, 34, 56)
12:12 Waypoint (54, 34, 56)
12:12 Observation (26, 78, 39)
Add...
12:12 Photo (45, 234, 90, 20, 40, 50)
12:12 Wait ()
12:12 Downlink
12:12 Correction (-20,0,0)
12:12 Photo (45, 234, 90, 20, 40, 50)
Zoom
125%
12:12
Sim
Past and Future Split Pane Widget
Runtime Marker
Options
Waypoints
Downlink Marker
Observations
Photo Activities
Wait Activities
Correction Activities
Resource Monitor
Simulate Button

5.1.1 > Waypoints


12:12 Waypoint (54, 34, 56)

5.1.1.1 > Purpose

Adding Waypoints to a Plan allows the user to specify specific points in the environment that they want the Rover to travel to. The system then figures out the low-level drive commands to send the Rover in order to reach those Waypoints.

5.1.1.2 > Functionality & Interactions

Adding a Waypoint to a Plan

Using the Add Activity Button
The user can click the Add Activity Button at the top of the Activity Planner and select “Waypoint”. A new Waypoint will be added to the bottom of the Plan, and will automatically be placed in Edit Mode.

Using Text Entry
The user can type “Waypoint (“ in the Activity Planner and the system will automatically recognize that they are entering a Waypoint Activity. A Waypoint Icon is then automatically displayed to the left of the text, and a time estimate (either Start Time or Duration) is displayed to the left of the Waypoint Icon. The Waypoint Edit Mode is automatically expanded below the Waypoint Activity. The user can then type the Waypoint Activity parameters, according to the following function signature:

```
Waypoint (X Position, Y Position, Drive Angle, Name)
```

Name is an optional parameter. If the user does enter a string for the Name parameter, that name is then displayed in the Activity Planner instead of the word “Waypoint”. When the user is not editing this line of code, they wouldn’t see the Name parameter. For example:

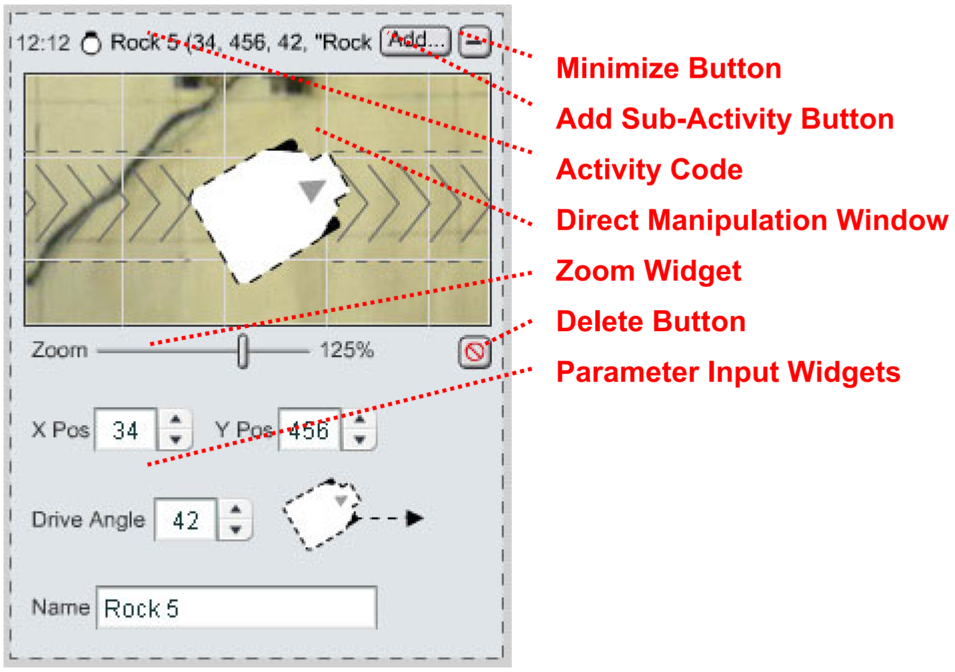
```
Turning Corner (45, 235, 34)
```

But if they were editing this line of code, the Name parameter is visible:

```
Turning Corner (45, 235, 34, “Turning Corner”)
```

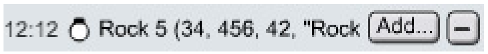
Editing a Waypoint

The following is a screen shot of a Waypoint in Edit Mode:



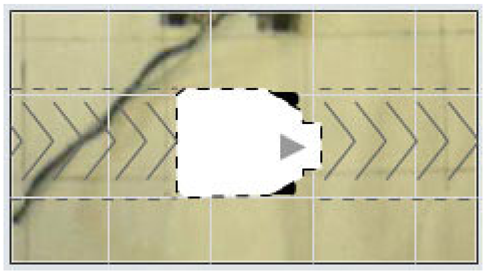
Users can click on a Waypoint in the Activity Planner or the Map Main Area to bring it into Edit Mode. When a Waypoint is in Edit Mode, the corresponding Waypoint Icon in the Map Main Area is in Selection Mode.

Activity Code



Users can edit the Waypoint Activity's parameters by directly editing the Activity Code text. As they do so, the Direct Manipulation Window, the Parameter Input Widgets, the graphical parameter representations (Drive Angle), and the corresponding Waypoint Icon in the Map Main Area change in real-time to reflect the new parameters. If the Activity Code is too long to fit in the designated area, it acts like a standard Text Box, in which only a specific amount of text is visible at any one time, and the user has to use the arrow keys to move through the text.

Direct Manipulation Window



Users can edit the Waypoint Activity's parameters by moving and rotating the Rover in the Direct Manipulation Window. As they do so, the Activity Code, the Parameter Input Widgets, the graphical parameter representations (Drive Angle), and the corresponding Waypoint Icon in the Map Main Area change in real-time to reflect the new parameters.

As the user moves the Rover Icon to the edges of the Direct Manipulation Window, the background moves in the direction of the edge (left, right, up, down, etc.), acting like a joystick, so that the user can effectively move the Rover Icon anywhere in the environment.

When the cursor is hovered over the Direct Manipulation Window, the cursor changes to the following custom cursors:

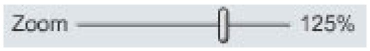
Move Mode (the cursor is over the moveable area of the Rover Icon)

Moving the Rover Icon around modifies the Waypoint's X Pos and Y Pos parameters.

Rotate Mode (the cursor is over the rotatable area of the Rover Icon)

Rotating the Rover Icon modifies the Waypoint's Drive Angle parameter.

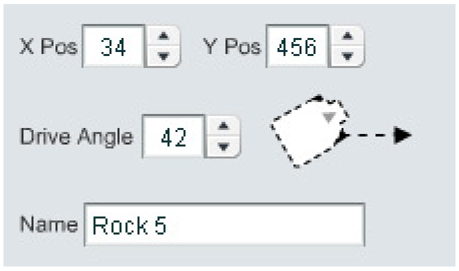
Zoom Widget



Zoom 125%

The Zoom Widget allows the user to zoom in and out of the image in the Direct Manipulation Window.

Parameter Input Widgets


X Pos 34 Y Pos 456

Drive Angle 42

Name Rock 5

Users can edit the Waypoint Activity's parameters by using the Parameter Input Widgets. As they do so, the Activity Code, the Direct Manipulation Window, the graphical parameter representations (Drive Angle), and the corresponding Waypoint Icon in the Map Main Area change in real-time to reflect the new parameters.

- X Pos: The X Position of the Waypoint in relation to the Map.
- Y Pos: The Y Position of the Waypoint in relation to the Map.
- Drive Angle: The angle a which the Rover is oriented as it drives along the specified path. For the Mind Storm, the only possible options are 0 or 180 (since the Mind Storm can only drive forward or reverse). For PER, any angle from 0 to 360 is possible.
- Name: The default or user-specified name given to the Waypoint.

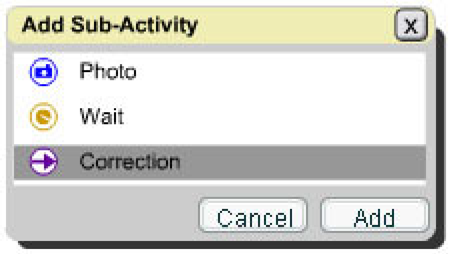
Delete Button

Pressing the Delete Button will remove the Activity from the plan, upon an 'OK' in a confirmation dialog box.

Add Sub-Activity Button

Add...

Users can add Sub-Activities to a Waypoint, at which point the Waypoint is changed into an Observation. Clicking the Add Sub-Activity Button results in the following dialog box:



Users click on a sub-activity and then the Add button to add a sub-activity to the Waypoint. The Add button is disabled until the user clicks on a sub-activity. Multiple selections are not allowed. After the user adds a sub-activity, the Add Sub-Activity dialog box is closed, the added sub-activity is in Edit Mode in the Activity Planner, and the Waypoint Icon is changed to an

Observation Icon. If the user hadn't renamed the default "Waypoint" Name, the Name of the Activity is changed to "Observation".

Minimize Button

To exit from Edit Mode, the user either presses the Minimize Button, or else clicks another Activity in the Plan.

5.1.2 > Observations

12:12 Observation (26, 78, 39)

5.1.2.1 > Purpose

Observations are essentially Waypoints that have Sub-Activities. They serve as a Plan organizational tool. Observations represent a waypoint where the Rover has stopped to do anything other than a long drive, such as taking a photo, or executing a short correction drive.

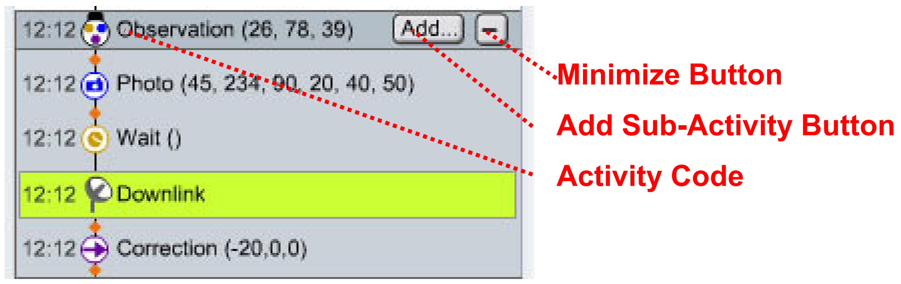
5.1.2.2 > Functionality & Interactions

Adding an Observation to a Plan

Users add Observations to the Plan through the Activity Planner by first adding a Waypoint (as described above) and then adding a Sub-Activity to that Waypoint (also described above).

Editing an Observation

The following is a screen shot of an Observation in Edit Mode:



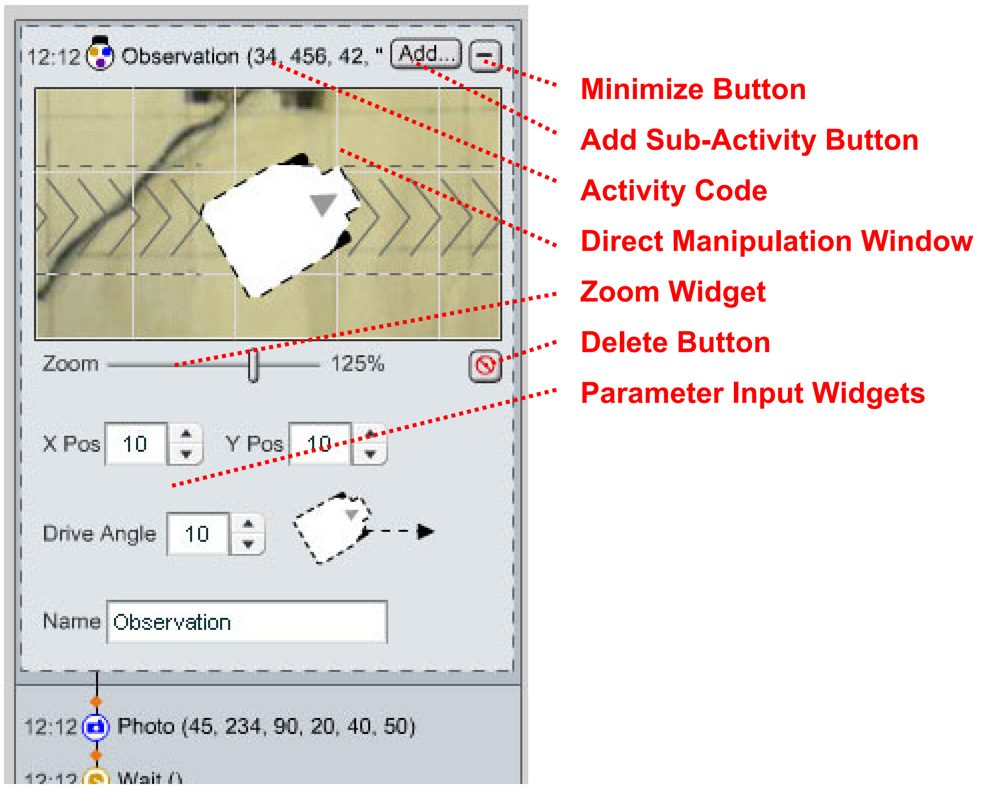
12:12 Observation (26, 78, 39)
12:12 Photo (45, 234, 90, 20, 40, 50)
12:12 Wait ()
12:12 Downlink
12:12 Correction (-20,0,0)

The user can click an Observation in the Activity Planner or the Map Main area to bring an Observation into Edit Mode. The Observation's Sub-Activities, which are normally hidden, and displayed. The user can click on any of the Sub-Activities to bring them into Edit Mode. When an

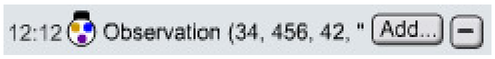
Observation is in Edit Mode, the corresponding Observation Icon in the Map Main Area is in Selection Mode.

Observation Waypoint Edit Mode

To edit the Waypoint that makes up the first Activity of an Observation, the user clicks the Observation's Activity Code when it is in Edit Mode. This will bring up the Observation Waypoint Edit Mode, as seen in the screen shot below:

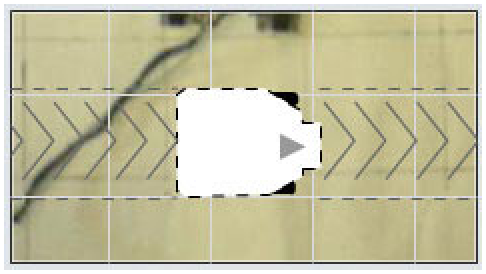


Activity Code



Users can edit the Observation Waypoint Activity's parameters by directly editing the Activity Code text. As they do so, the Direct Manipulation Window, the Parameter Input Widgets, the graphical parameter representations (Drive Angle), and the corresponding Waypoint Icon in the Map Main Area change in real-time to reflect the new parameters. If the Activity Code is too long to fit in the designated area, it acts like a standard Text Box, in which only a specific amount of text is visible at any one time, and the user has to use the arrow keys to move through the text.

Direct Manipulation Window



Users can edit the Observation Waypoint Activity's parameters by moving and rotating the Rover in the Direct Manipulation Window. As they do so, the Activity Code, the Parameter Input Widgets, the graphical parameter representations (Drive Angle), and the corresponding Waypoint Icon in the Map Main Area change in real-time to reflect the new parameters.

As the user moves the Rover Icon to the edges of the Direct Manipulation Window, the background moves in the direction of the edge (left, right, up, down, etc.), acting like a joystick, so that the user can effectively move the Rover Icon anywhere in the environment.

When the cursor is hovered over the Direct Manipulation Window, the cursor changes to the following custom cursors:

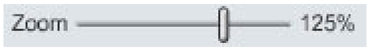
Move Mode (the cursor is over the moveable area of the Rover Icon)

Moving the Rover Icon around modifies the Observation Waypoint's X Pos and Y Pos parameters.

Rotate Mode (the cursor is over the rotatable area of the Rover Icon)

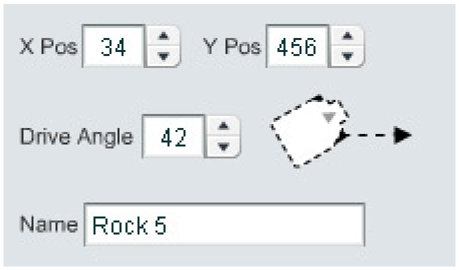
Rotating the Rover Icon modifies the Observation Waypoint's Drive Angle parameter.

Zoom Widget



The Zoom Widget allows the user to zoom in and out of the image in the Direct Manipulation Window.

Parameter Input Widgets



Users can edit the Observation Waypoint Activity's parameters by using the Parameter Input Widgets. As they do so, the Activity Code, the Direct Manipulation Window, the graphical parameter representations (Drive Angle), and the corresponding Waypoint Icon in the Map Main Area change in real-time to reflect the new parameters.

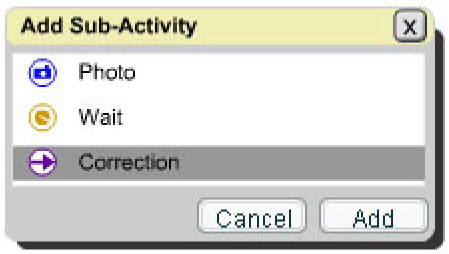
- X Pos: The X Position of the Observation Waypoint in relation to the Map.
- Y Pos: The Y Position of the Observation Waypoint in relation to the Map.
- Drive Angle: The angle a which the Rover is oriented as it drives along the specified path. For the Mind Storm, the only possible options are 0 or 180 (since the Mind Storm can only drive forward or reverse). For PER, any angle from 0 to 360 is possible.
- Name: The default or user-specified name given to the Observation.

Delete Button

Pressing the Delete Button will remove the Observation from the plan, upon an 'OK' in a confirmation dialog box.

Add Sub-Activity Button

Users can add Sub-Activities to an Observation clicking the Add Sub-Activity Button, which results in the following dialog box:

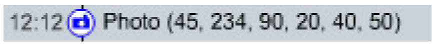


Users click on a sub-activity and then the Add button to add a sub-activity to the Observation. The Add button is disabled until the user clicks on a sub-activity. Multiple selections are not allowed. After the user adds a sub-activity, the Add Sub-Activity dialog box is closed, and the added sub-activity is in Edit Mode in the Activity Planner.

Minimize Button

When in Observation Waypoint Edit Mode, clicking the Minimize button collapses the Observation Waypoint Edit Screen. When in regular Observation Edit mode the user either presses the Minimize Button, or else clicks another Activity in the Plan to exit from Observation Edit Mode.

5.1.3 > Photo Activities


12:12 Photo (45, 234, 90, 20, 40, 50)

5.1.3.1 > Purpose

Adding a Photo Activity to a Plan allows the user to command the Rover to take a photograph.

5.1.3.2 > Functionality & Interactions

Adding a Photo to a Plan

Using the Add Activity Button

The user can click the Add Sub-Activity Button in either the Observation Edit Mode window or the Waypoint Edit Mode window and select “Photo”. A new Photo activity will be added to the bottom of the Sub-Plan, and will automatically be placed in Edit Mode.

Using Text Entry

The user can type “Photo (“ in the Activity Planner and the system will automatically recognize that they are entering a Photo Activity. A Photo Icon is then automatically displayed to the left of the text, and a time estimate (either Start Time or Duration) is displayed to the left of the Photo Icon. The user can then type the Photo Activity parameters, according to the following function signature:

Mind Storm:

Photo (X Offset, Y Offset, Angle)

PER:

Photo (X Offset, Y Offset, Rover Angle, Camera Angle, Image Distance, FOV)

The Photo Edit Mode is then automatically expanded below the Photo Activity.

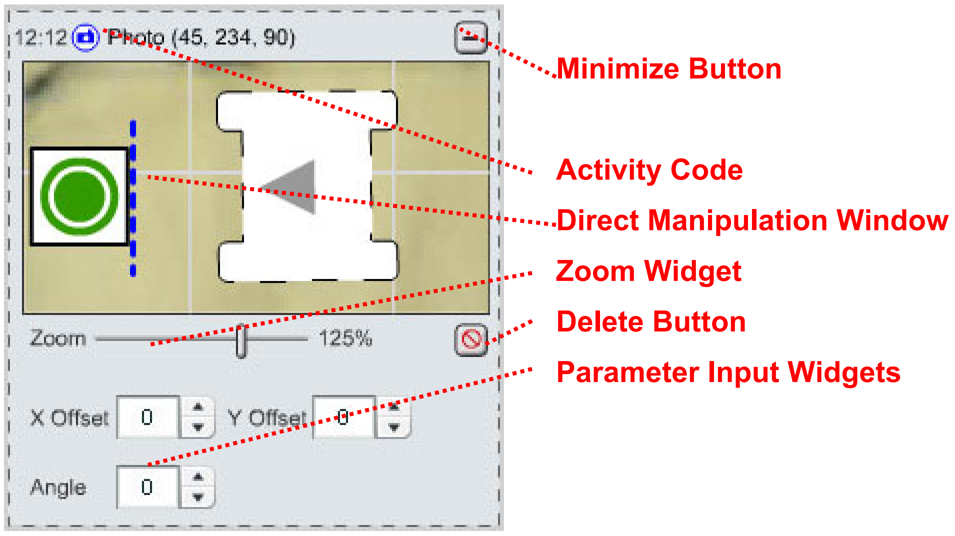
If the user types a Photo Activity in the Activity Planner outside the scope of an Observation, the system will do one of two things:

1. If there is an Observation directly above the Photo Activity, that Observation will expand, and the Photo Activity will be placed as the last Sub-Activity within that Observation.
2. Otherwise, the Waypoint above the Photo Activity will turn into an Observation, expand, and the Photo Activity will be placed as a Sub-Activity within that Observation.

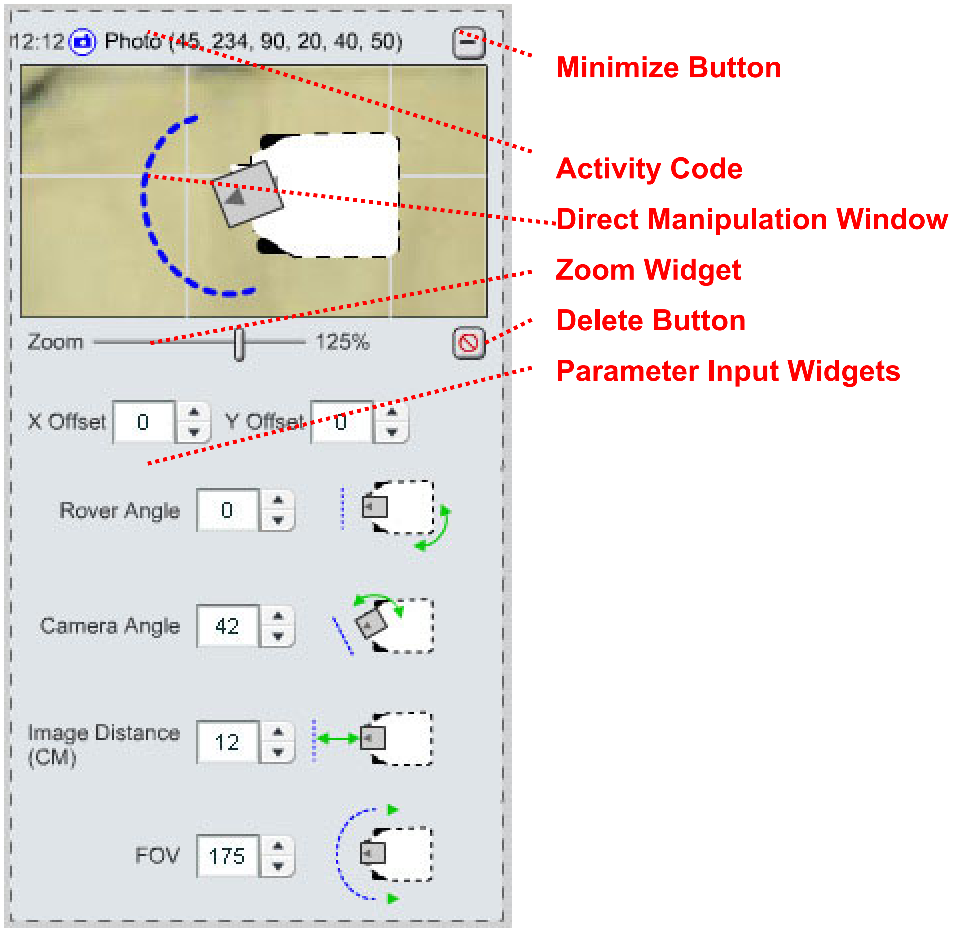
Editing a Photo

The following is a screen shot of a Photo Activity in Edit Mode:

Mind Storm:



PER:



Science Viewer

When a Past Photo Activity is in Selected (Edit) Mode, the following events occur:

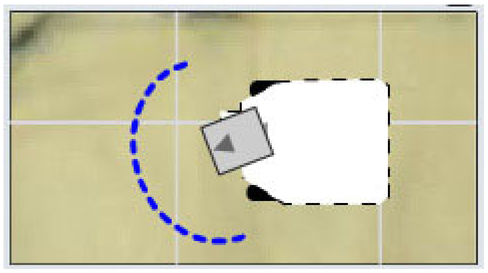
- The Science Viewer scrolls to the associated Photo, which is displayed in Selected Mode.
- The Map displays the last Orbiter Image taken before the Photo was taken.
- The orbiter image number feedback in the Orbiter Image Browser displays the associated orbiter image number (such as “4 / 7”).

When the Past Photo Activity is no longer in Selected Mode, the above interface elements revert back to their previous state.

Activity Code

Users can edit the Photo Activity’s parameters by directly editing the Activity Code text. As they do so, the Direct Manipulation Window, the Parameter Input Widgets, and the graphical parameter representations change in real-time to reflect the new parameters. If the Activity Code is too long to fit in the designated area, it acts like a standard Text Box, in which only a specific amount of text is visible at any one time, and the user has to use the arrow keys to move through the text.

Direct Manipulation Window



Users can edit the Photo Activity's parameters by moving and rotating the Rover in the Direct Manipulation Window. As they do so, the Activity Code, the Parameter Input Widgets, and the graphical parameter representations change in real-time to reflect the new parameters.

As the user moves the Rover Icon to the edges of the Direct Manipulation Window, the background moves in the direction of the edge (left, right, up, down, etc.), acting like a joystick, so that the user can effectively move the Rover Icon anywhere in the environment.

A dotted Photo Line is also displayed along with the Rover Icon in order to show an estimated surface plane of the photo that will be taken.

The user can directly manipulate:

- The Photo Line. For both Mindstorm and PER, the user can move and rotate the Photo Line, and the Rover Icon and the parameters will adjust automatically. For PER, the user can also directly manipulate the Field of View (FOV).
- The Rover. For both Mind Storm and PER, the user can move and rotate the Rover Icon, and the Photo Line and the parameters will adjust automatically.
- The Camera. For PER only, the user can independently move and rotate the camera, and the Photo Line and the parameters will adjust automatically.

When the cursor is hovered over the Direct Manipulation Window, the specific component (either the Photo Line, the Camera, or the Rover Icon) which would be modified – based on the cursor's position - is highlighted with a yellow halo around its representation. The cursor changes to the following custom cursors:

Move Mode (the cursor is over the moveable area of the Rover Icon or the Photo Line)

Moving the Rover Icon around modifies the Photo's X Offset and Y Offset parameters. Moving the Photo Line around modifies the Photo's X Offset, Y Offset, and for PER only, the Image Distance parameters. The user cannot move the Camera Icon independently of the Rover Icon (at least for our current robotic platforms).

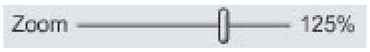
Rotate Mode (the cursor is over the rotatable area of the Rover Icon, the Camera Icon, or the Photo Line)

Rotating the Rover Icon modifies the Photo's Rover Angle parameter. Rotating the Camera Icon (PER only) modifies the Photo's Camera Angle parameter. Rotating the Photo Line modifies the Photo's X Offset, Y Offset, and Rover Angle parameters.

FOV Mode (the cursor is over either end of the Photo Line – PER only)

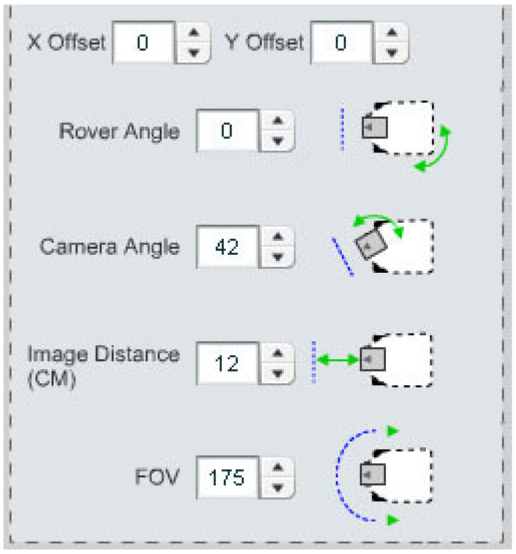
Directly manipulating the FOV modifies the Photo's FOV parameter. The exact interaction for directly manipulating the FOV is to be determined in the next version of this specification.

Zoom Widget



The Zoom Widget allows the user to zoom in and out of the image in the Direct Manipulation Window.

Parameter Input Widgets


X Offset 0 Y Offset 0

Rover Angle 0

Camera Angle 42

Image Distance (CM) 12

FOV 175

Users can edit the Photo Activity's parameters by using the Parameter Input Widgets. As they do so, the Activity Code, the Direct Manipulation Window, the graphical parameter representations, and the corresponding Waypoint Icon in the Map Main Area change in real-time to reflect the new parameters.

- X Offset. The horizontal distance the Rover should move from the original Observation Waypoint's position before taking a picture.
- Y Offset. The vertical distance the Rover should move from the original Observation Waypoint's position before taking a picture.
- Rover Angle. The angle the Rover should turn from the original Observation Waypoint's position before taking a picture.
- Camera Angle (PER only). The angle the Camera should rotate in relation to the Rover body before taking a picture.
- Image Distance (PER only). The distance the image should be taken from the camera. This effects the tilt of the camera.
- Field of View (PER only). The desired field of view for the photo. PER is capable of taking multiple images and then stitching them together, which effectively translates to a FOV.

Delete Button

🚫

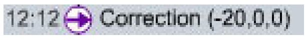
Pressing the Delete Button will remove the Activity from the plan, upon an 'OK' in a confirmation dialog box.

Minimize Button

⊟

To exit from Edit Mode, the user either presses the Minimize Button, or else clicks another Activity in the Plan.

5.1.4 > Correction Activities


12:12 Correction (-20,0,0)

5.1.4.1 > Purpose

Adding Correction Activities to the Plan allows the user to fine-tune the position of the Rover. While Waypoints are placed on the Map Main Area generally for long drives, Correction Activities are added to the Plan to specify very short drives. The drive distance for a Correction Activity is restricted to no longer than the size of the Rover.

5.1.4.2 > Functionality & Interactions

Adding a Correction to a Plan

Using the Add Activity Button

The user can click the Add Sub-Activity Button in either the Observation Edit Mode window or the Waypoint Edit Mode window and select “Correction”. A new Correction activity will be added to the bottom of the Sub-Plan, and will automatically be placed in Edit Mode.

Using Text Entry

The user can type “Correction (“ in the Activity Planner and the system will automatically recognize that they are entering a Correction Activity. A Correction Icon is then automatically displayed to the left of the text, and a time estimate (either Start Time or Duration) is displayed to the left of the Correction Icon. The user can then type the Correction Activity parameters, according to the following function signature:

Correction (X Offset, Y Offset, Angle)

The Correction Edit Mode is then automatically expanded below the Correction Activity.

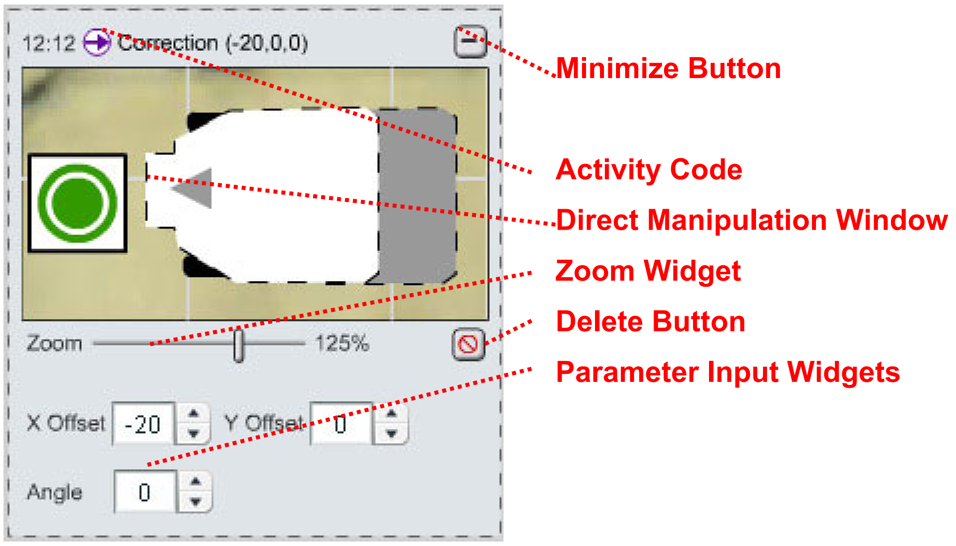
If the user types a Correction Activity in the Activity Planner outside the scope of an Observation, the system will do one of two things:

1. If there is an Observation directly above the Correction Activity, that Observation will expand, and the Correction Activity will be placed as the last Sub-Activity within that Observation.

2. Otherwise, the Waypoint above the Correction Activity will turn into an Observation, expand, and the Correction Activity will be placed as a Sub-Activity within that Observation.

Editing a Correction

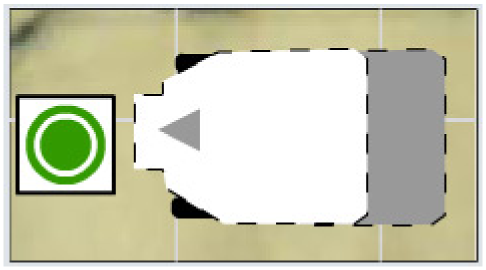
The following is a screen shot of a Correction Activity in Edit Mode:



Activity Code

Users can edit the Correction Activity's parameters by directly editing the Activity Code text. As they do so, the Direct Manipulation Window and the Parameter Input Widgets change in real-time to reflect the new parameters. If the Activity Code is too long to fit in the designated area, it acts like a standard Text Box, in which only a specific amount of text is visible at any one time, and the user has to use the arrow keys to move through the text.

Direct Manipulation Window



Users can edit the Correction Activity’s parameters by moving and rotating the Rover in the Direct Manipulation Window. As they do so, the Activity Code and the Parameter Input Widgets change in real-time to reflect the new parameters.

As the user moves the Rover Icon to the edges of the Direct Manipulation Window, the background moves in the direction of the edge (left, right, up, down, etc.), acting like a joystick.

The drive distance for a Correction Activity is restricted to no longer than the size of the Rover. The original Rover position icon is displayed behind the Correction Waypoint Rover icon in a dark grey.

When the cursor is hovered over the Direct Manipulation Window, the cursor changes to the following custom cursors:

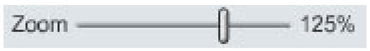
Move Mode (the cursor is over the moveable area of the Rover Icon)

Moving the Rover Icon around modifies the Correction’s X Offset and Y Offset parameters.

Rotate Mode (the cursor is over the rotatable area of the Rover Icon)

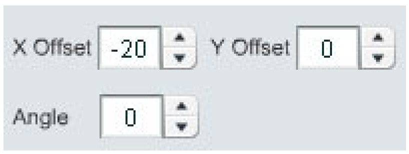
Rotating the Rover Icon modifies the Correction’s Angle parameter.

Zoom Widget



The Zoom Widget allows the user to zoom in and out of the image in the Direct Manipulation Window.

Parameter Input Widgets



X Offset -20 Y Offset 0

Angle 0

Users can edit the Correction Activity’s parameters by using the Parameter Input Widgets. As they do so, the Activity Code, the Direct Manipulation Window, and the corresponding Waypoint Icon in the Map Main Area change in real-time to reflect the new parameters.

- X Offset. The horizontal distance the Rover should move from the Rover's previous position.
- Y Offset. The vertical distance the Rover should move from the Rover's previous position.
- Angle. The angle the Rover should turn from the Rover's previous position.

Delete Button

Pressing the Delete Button will remove the Activity from the plan, upon an 'OK' in a confirmation dialog box.

Minimize Button

To exit from Edit Mode, the user either presses the Minimize Button, or else clicks another Activity in the Plan.

5.1.5 > Wait Activities

12:12 Wait ()

5.1.5.1 > Purpose

Adding Wait Activities to the Plan allows the user to command the Rover to stop and do nothing for a specified amount of time before moving on to the next Activity in the Plan.

5.1.5.2 > Functionality & Interactions

Adding a Wait Activity to a Plan

Using the Add Activity Button

The user can click the Add Sub-Activity Button in either the Observation Edit Mode window or the Waypoint Edit Mode window and select "Wait". A new Wait activity will be added to the bottom of the Sub-Plan, and will automatically be placed in Edit Mode.

Using Text Entry

The user can type "Wait (" in the Activity Planner and the system will automatically recognize that they are entering a Wait Activity. A Wait Icon is then automatically displayed to the left of the text, and a time estimate (either Start Time or Duration) is displayed to the left of the Wait Icon. The user can then type the Wait Activity parameters, according to the following function signature:

Wait ()

(Note: In this version of the Functionality Specification, there is one type of Wait command: Wait Until Next Download. It is conceivable, however, that other Rover platforms may have tools on board that would require the user to specify a particular amount of time to wait, and so we've left this option open.)

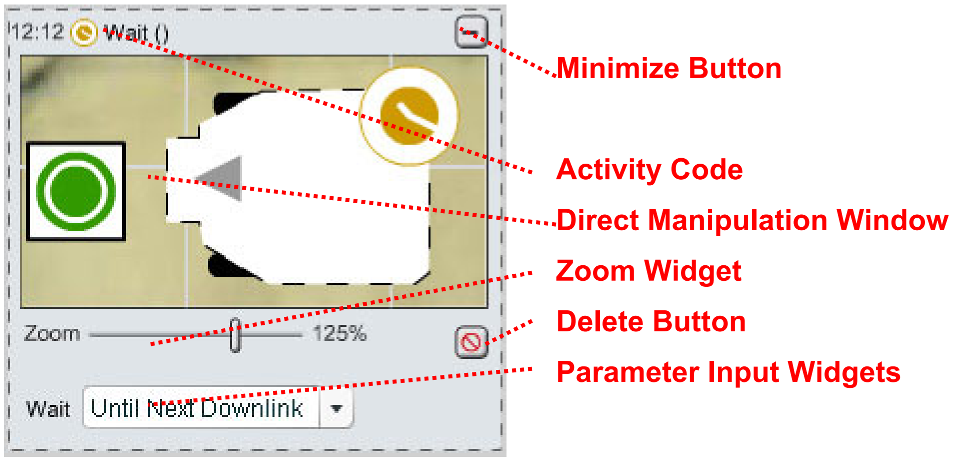
The Wait Edit Mode is then automatically expanded below the Wait Activity.

If the user types a Wait Activity in the Activity Planner outside the scope of an Observation, the system will do one of two things:

1. If there is an Observation directly above the Wait Activity, that Observation will expand, and the Wait Activity will be placed as the last Sub-Activity within that Observation.
2. Otherwise, the Waypoint above the Wait Activity will turn into an Observation, expand, and the Wait Activity will be placed as a Sub-Activity within that Observation.

Editing a Wait Activity

The following is a screen shot of a Wait Activity in Edit Mode:

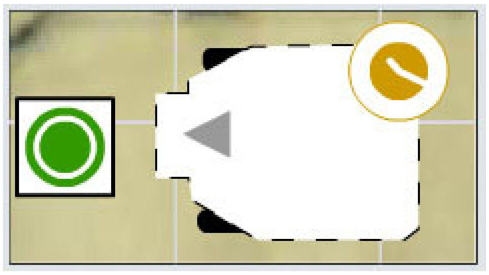


Wait Until Next Downlink

Activity Code

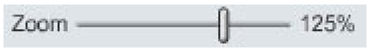
Users can edit the Wait Activity's parameters by directly editing the Activity Code text. As they do so, the Direct Manipulation Window and the Parameter Input Widgets change in real-time to reflect the new parameters. If the Activity Code is too long to fit in the designated area, it acts like a standard Text Box, in which only a specific amount of text is visible at any one time, and the user has to use the arrow keys to move through the text.

Direct Manipulation Window



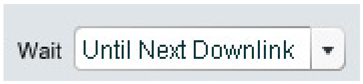
The user cannot interact with the Direct Manipulation Window for a Wait Activity. It is only displayed as feedback so the user knows where the Rover is located when the Wait command will execute. The user is made aware of this lack-of-affordance by the fact that the cursor does not change when being moved around the Direct Manipulation Window.

Zoom Widget



The Zoom Widget allows the user to zoom in and out of the image in the Direct Manipulation Window.

Parameter Input Widgets



Users can edit the Wait Activity's parameters by using the Parameter Input Widgets. In this version of the Functionality Specification, there is one type of Wait command (and so only one item in the combo box): Wait Until Next Download. It is conceivable, however, that other Rover platforms may have tools on board that would require the user to specify a particular amount of time to wait, and so we've left this option open.

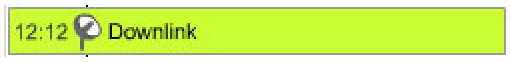
Delete Button

Pressing the Delete Button will remove the Activity from the plan, upon an 'OK' in a confirmation dialog box.

Minimize Button

To exit from Edit Mode, the user either presses the Minimize Button, or else clicks another Activity in the Plan.

5.1.6 > Downlink Marker



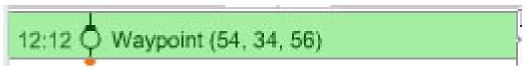
5.1.6.1 > Purpose

The Downlink Marker shows the user where in their Plan they will receive telemetry data.

5.1.6.2 > Functionality & Interactions

Downlink Markers are automatically added to the Plan as the user adds Activities. The user does not manipulate the Downlink Markers; they are only there to give the user feedback of what points in the Plan the user will receive telemetry data back from the Rover.

5.1.7 > Runtime Marker



5.1.7.1 > Purpose

The Runtime Marker shows the user which Activity the system estimates is currently being executed on the Rover.

5.1.7.2 > Functionality & Interactions

The Runtime Marker always remains in the same place. As the Plan executes it scrolls upward, and the estimated currently running activity moves behind the Runtime Marker. The user does not manipulate the Runtime Marker; it is only there to give the user feedback of the system's estimate of which Activity is currently being executed by the Rover.

When an Observation arrives at the Runtime Marker, the Observation automatically expands to display it's Sub-Activities, so that the user can see when each of the Sub-Activities is being executed (estimated).

5.1.8 > Resource Monitor

5.1.8.1 > Purpose

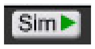
The Resource Monitor gives the user feedback of the relevant total Resource consumption of a given plan (for both the Nominal or Optional Plans).

5.1.8.2 > Functionality & Interactions

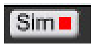
The user does not manipulate the Resource Monitor; it is only there to give the user feedback of the relevant total Resource consumption of a given plan. When there is more than one Plan Option visible on the screen at once, a separate Resource Monitor is displayed for each of the Plans. This allows the user to compare the total Resource consumption of each Plan Option. In our limited Rover domain, the only crucial Resource is the total time taken to executed a Plan. For other, more complex Rover domains, the Resource Monitor would also display other relevant Resources, such as battery consumption or memory usage.

5.1.9 > Simulate Button

Play



Stop



5.1.9.1 > Purpose

The Simulate Button allows the user to visually simulate a Plan in the Map Main Area.

5.1.9.2 > Functionality & Interactions

The user presses the Simulate Play Button to simulate a Plan in the Map Main Area. The Now Icon moves along the Planned path at twice the normal speed. As it arrives at Observations, it rotates and waits according to the Sub-Activities within the Observation. As the Simulation plays, the Activities in the Activity Planner scroll up as if they were being executed in real-time (but at twice the speed).

When the Simulation is currently playing, the Simulate Play Button changes to the Simulate Stop Button. The user pressed the Simulate Stop Button in order to stop the Simulation, at which point all interface elements return to their previous states, and the Simulate Stop Button changes to the Simulate Start Button.

Each Plan Option has it’s own Simulate Button.

5.1.10 > Options

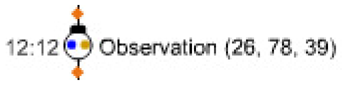
5.1.10.1 > Purpose

Adding options to a Plan allow the user to specify optional Plan paths ahead of time. For example, they may want to linger and examine a target if they discover it is of high value, but otherwise they would just move on to the next target. Explicitly planning out these two options ahead of time allows the user to quickly choose between one of the two options when they reach their decision point, rather than having to spend time re-planning in real-time when they reach their decision point.

5.1.10.2 > Functionality & Interactions

Adding an Option

The Plan as represented in the Activity Planner includes Option Triggers, which are small orange diamonds:

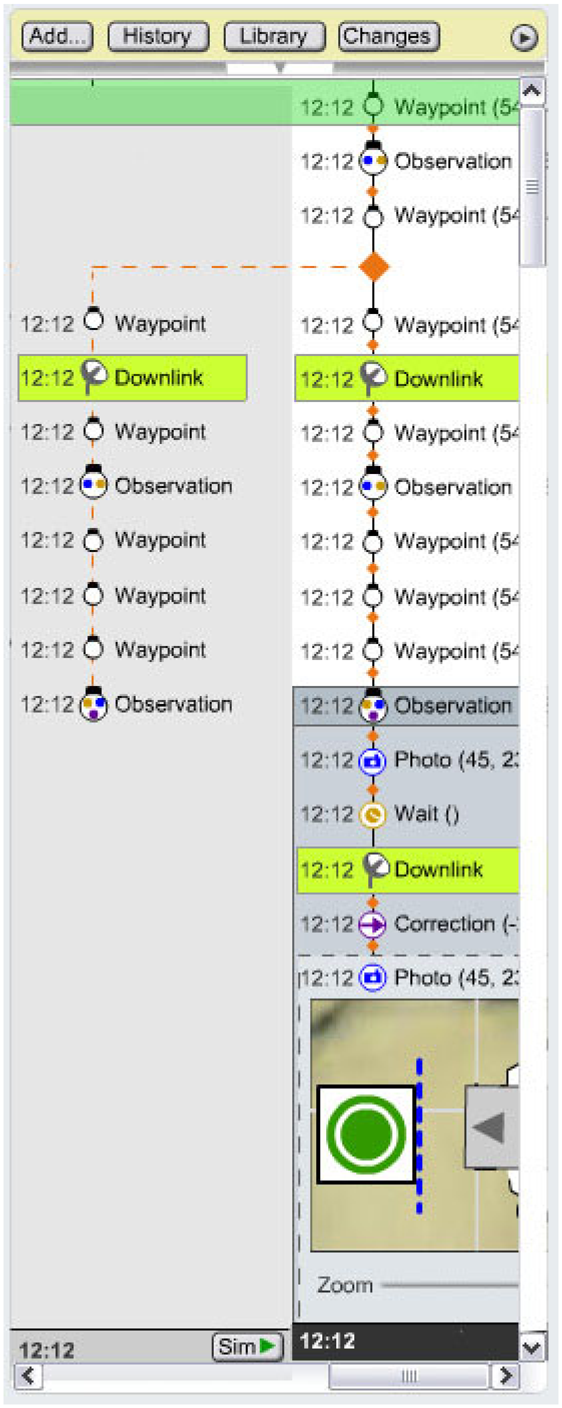


These represent points in the plan where the user can add Optional Plan paths. The user interacts with these Option Triggers to add Optional Plan paths. The details of this interaction are to be designed for the next version of this Specification.

Users can add up to two Optional Plan paths at any given Option Trigger, giving them a total of three possible Options at any given decision point (one nominal and two optional). This number is derived from user research, but should be verified with further user testing.

Option Representation in the Activity Planner

An Option is represented in the Activity Planner with a large orange diamond and a dotted orange line going off to the left or the right. The Optional Plan then proceeds down the screen in the same way as an Nominal Plan, except the line connecting the Activities is orange and dotted, the background color is a light gray, the detailed signature of each Activity is not displayed (unless selected), and the background color of the Resource Monitor is a light gray. See screen shot below:



The Resource Monitor and Simulation Button are applicable to each specific Optional Plan path.

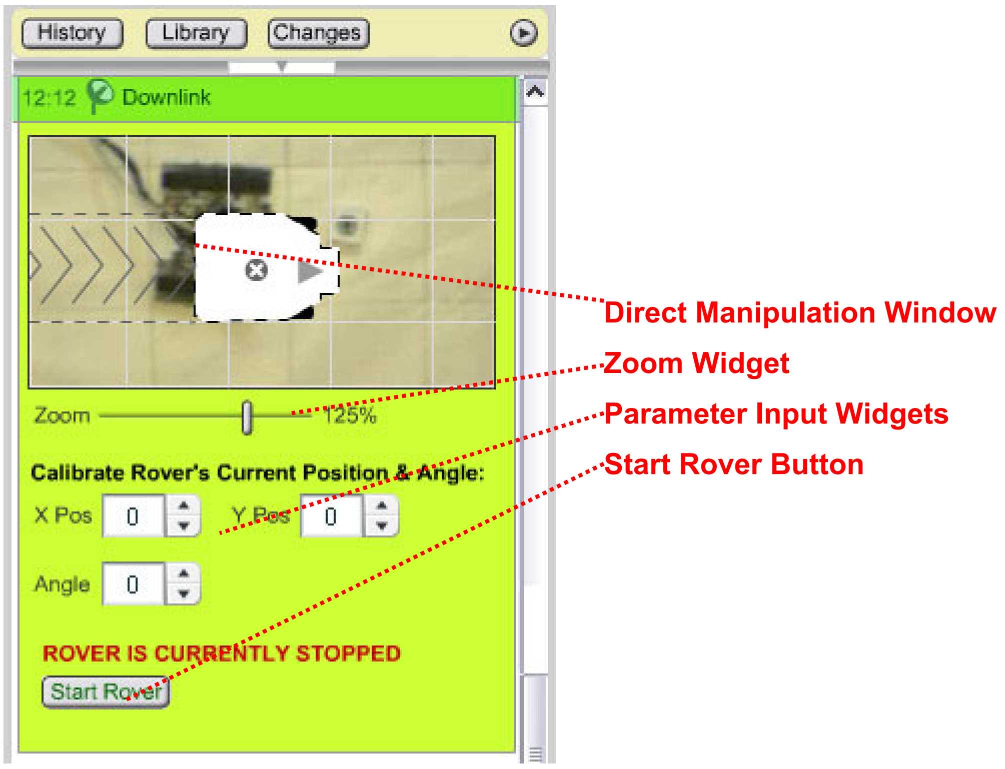
Our original design ideas had separate representations for different types of Options (Fork, Delay, Rush, Delete, Insert, Skip, Detour). However, when we showed these representations to users at JPL, they had difficulty understanding the graphical representations of the different option types. They commented that they would rather see the Optional Plan path represented as a completely separate line, even if it meant repeating Activities that were present in the Nominal Plan path. This streamlined representation also makes it very easy for users to compare the Resources used in each Optional Plan path. So for now, we are only representing the Optional Plan paths as single unique lines. This representation, however, needs to be fully flushed out and user tested, and should be further refined in the next version of this Specification (for example: how would you represent an Optional Delete?).

Interactions

The exact interactions involved with specifying, viewing, and then choosing an Optional Plan are to be designed for the next version of this Specification. Questions to be addressed:

- What is the exact interaction for adding an Optional Plan?
- Currently, the user has to horizontally scroll to see three Optional Plan side-by-side. Is this acceptable? Should more screen-real-estate be delegated to the Activity Planner?
- How and at what points can users turn an Optional Plan into a Nominal Plan?
- What if two Optional Plan paths visually overlap? Is there a way to hide and show individual Optional Plan paths?
- Are all types of Options (Fork, Delay, Rush, Delete, Insert, Skip, Detour) adequately representable using the simplified single line Optional Plan path representation?

5.1.11 > Calibration



5.1.11.1 > Purpose

Calibration allows the user to update the system's model of ground truth (where the Rover currently and actually is) after each Downlink.

5.1.11.2 > Functionality & Interactions

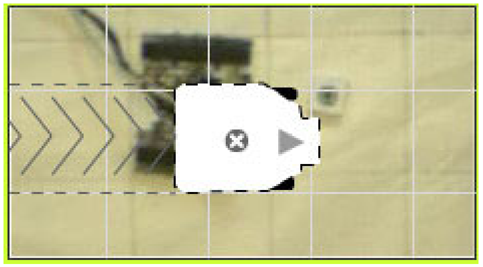
Temporal Aspects

As the Plan executes in real-time, the Activities in the Activity Planner scroll up past the Runtime Marker. When the Plan gets to a Downlink point, the Rover automatically stop executing Activities. The Downlink Marker moves behind the Runtime Marker, and the Downlink Marker animates to expand and display the Calibration interface.

The user will see the Rover's actual position from the newly acquired Orbiter Image. They will then be given a chance to update the system's model of ground truth, or where the Rover currently is. This assures that the rest of the plan will be executed according to where the Rover actually is, as opposed to where the system thought the Rover would be.

The Now Icon is displayed in the Calibration interface in it's editable state so that the user can manipulate the Now Icon. After the user has corrected the position of the Now Icon, they press the Start Rover button to continue execution of the Plan.

Direct Manipulation Window

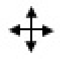


Users can edit the Calibration's parameters by moving and rotating the Rover in the Direct Manipulation Window. As they do so, the Parameter Input Widgets change in real-time to reflect the new parameters.

As the user moves the Rover Icon to the edges of the Direct Manipulation Window, the background moves in the direction of the edge (left, right, up, down, etc.), acting like a joystick, so that the user can effectively move the Rover Icon anywhere in the environment.

When the cursor is hovered over the Direct Manipulation Window, the cursor changes to the following custom cursors:

Move Mode (the cursor is over the moveable area of the Rover Icon)

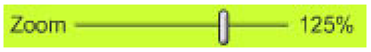


Moving the Rover Icon around modifies the Calibration's X Pos and Y Pos parameters.

Rotate Mode (the cursor is over the rotatable area of the Rover Icon)

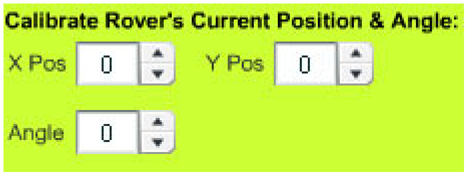
Rotating the Rover Icon modifies the Calibration's Angle parameter.

Zoom Widget



The Zoom Widget allows the user to zoom in and out of the image in the Direct Manipulation Window.

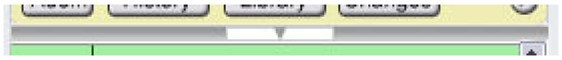
Parameter Input Widgets



Users can edit the Calibration's parameters by using the Parameter Input Widgets. As they do so, the Direct Manipulation Window and the corresponding Now Icon in the Map Main Area change in real-time to reflect the new parameters.

- X Pos. The X position of the Now Icon.
- Y Pos. The Y position of the Now Icon.
- Angle. The angle of the Now Icon.

5.1.12 > Past and Future Split Pane Widget



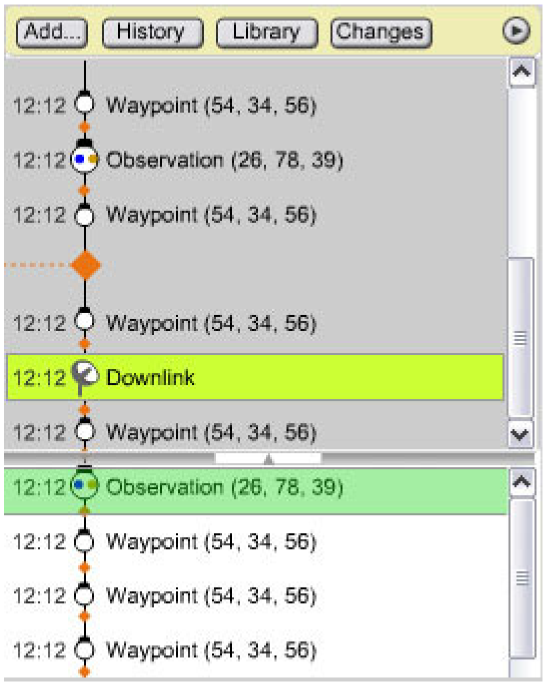
5.1.12.1 > Purpose

The Past and Future Split Pane Widget allows the user to quickly hide and show Activities that have recently been executed. It is essentially a dividing line between Activities that have yet to execute or are currently executing and those Activities which have already executed.

5.1.12.2 > Functionality & Interactions

The Past and Future Split Pane Widget hides past Activities by default. To Display past Activities, the user clicks the Split Pane Widget, which then animates downward to reveal the most recent past Activities. The arrow in the center of the Split Pane Widget points upward when past Activities are visible. To hide visible past Activities, the user clicks the Split Pane widget, which then animates upward to hide the past Activities.

Past Activities are displayed in the Activity Planner with a grey background, and are not editable. All of the clickable behaviors of Observations, Waypoints, and Activities (such as clicking a Waypoint and having it's corresponding Waypoint in the Map Main Area turn highlighted) are still applicable to past Activities, but user cannot edit any of the parameters of past Activities. Visible past Activities appear as follows:


Add...
History
Library
Changes
12:12 Waypoint (54, 34, 56)
12:12 Observation (26, 78, 39)
12:12 Waypoint (54, 34, 56)
12:12 Waypoint (54, 34, 56)
12:12 Downlink
12:12 Waypoint (54, 34, 56)
12:12 Observation (26, 78, 39)
12:12 Waypoint (54, 34, 56)
12:12 Waypoint (54, 34, 56)
12:12 Waypoint (54, 34, 56)

5.1.13 > Text Entry Behavior

Text entry in the Activity Planner Main Area should behavior similarly to any standard text editor, such Word or Note Pad. However, there are small nuances involved with the Activity Planner that need to be completely flushed out. This will be handled in the next version of this Specification.

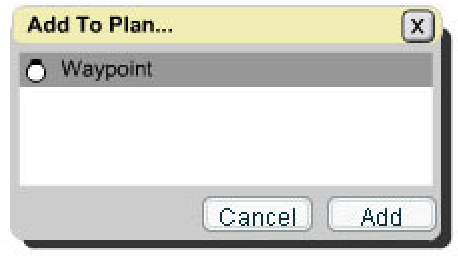
5.2 > Add Activity Button

5.2.1 > Purpose

The Add Activity Button allows the user to add Activities to the Plan without typing the code directly.

5.2.2 > Functionality & Interactions

Users can add Activities to the Plan by clicking the Add Activity Button, which results in the following dialog box:



Users click on an Activity and then the Add button to add an Activity to the Plan. The Add button is disabled until the user clicks on an Activity. Multiple selections are not allowed. After the user adds an Activity, the Add Sub-Activity dialog box is closed, and the added Activity is in Edit Mode in the Activity Planner.

Note that currently the only type of Activity the user can add from the Add Activity Button is a Waypoint. (Of course they can add Sub-Activities, such as Photo, Wait, and Correction, to a Waypoint or Observation using the Add Sub-Activities Button within a Waypoint or Observation Edit Mode screen.) With our current domain and design, Waypoint is the only type of activity that can logically be added to the Plan at this level. However, we wanted the design to be flexible enough that that other types of Activities could be added in future versions of this Specification if need be.

5.3 > History Button

5.3.1 > Purpose

The History Button displays a pop-up window where the user can view, search, sort, and filter all executed Activities from the current and past sols (days).

5.3.2 > Functionality & Interactions

Although the user can use the Past and Future Split Pane Widget in the Activity Planner Main Area to quickly view recently executed Activities, we also think there is a need for a more detailed and isolated view of past Activities, or History. This isolated view, in it's own window, would allow the user to view Activities from the current as well as past sols. It would also offer more advanced interactive features that would be relevant to History but not present in the Activity Planner Main Area, such as searching, sorting, and filtering, and potentially other advanced features. The details of the History window need to be worked out in the next version of this Specification.

5.4 > Library Button

5.4.1 > Purpose

The Library Button allows the user to access the Library in a separate window. The Library allows the user to create and store macros, so that they can quickly add commonly-used combinations of Activities to their Plan without having to create them from scratch.

5.4.2 > Functionality & Interactions

We think there is a strong need to have a Library area of the interface where users can create and access macros. These macros would be commonly-used combinations of Activities which the user could then easily add to their Plan, specifying each time the parameters required by the macro. The details of the Library – it's functionality and interactions – are to be designed for the next version of this Specification.

5.5 > Changes Button

5.5.1 > Purpose

The Changes Button allows the user to access the Changes window, where they can review and revert back to specific edit points.

5.5.2 > Functionality & Interactions

The Changes window would work in a similar fashion to the PhotoShop History Palette. The exact functionality and interactions are to be designed for the next version of this Specification.

5.6 > Activity Planner Options Button

5.6.1 > Purpose

The Activity Planner Options button allows the user to specify various options that affect the appearance and behavior of the Activity Planner.

5.6.2 > Functionality & Interactions

Clicking the Activity Planner Options Button displays the Activity Planner Options Menu. The right edge of the menu is aligned with the right-edge of the Activity Planner Options Button and the top edge of the menu is aligned with the bottom edge of the Activity Planner Options Button. Clicking the Activity Planner Options Button when the Activity Planner Options Menu is currently displayed hides the Activity Options Menu. The Activity Planner Options Menu contains the following items (note the position of the menu dividers):

View Visual Mode
View Code Mode

View Activity Start Times
View Activity Durations

View Visual Mode and View Code Mode act like radio buttons: only one can be selected at a time, and one of the options must always be selected. Whichever option is selected will have a checkmark graphic to the left of the text. When in Visual Mode, the Activity Planner displays all graphical representations, including Waypoint, Observation, and Activity icons, Direct Manipulation Windows, and Parameter Input Widgets. When in Code Mode, the Activity Planner displays text only. This is an expert mode in which the user is not slowed down by graphical representations and can essentially type code freely.

View Activity Start Times and View Activity Durations act like radio buttons: only one can be selected at a time, and one of the options must always be selected. Whichever option is selected will have a checkmark graphic to the left of the text. When View Activity Start Times is selected, the Activity start times are displayed next to each line in the Activity Planner. When View Activity Durations is selected, the Activity durations are displayed next to each line in the Activity Planner. This was designed this way because of limited screen space. This should be verified with user tests however, as it may turn out that users need to view both the Activity start times and the Activity durations at the same time.

There are many other possibly items that could be added to the Activity Planner Options Menu, and these possibilities will be explored in the next version of this Specification.