Instructions for a Typical Interaction

This instruction will walk a user through the task of "beaming down an away team". It assumes that you have already selected the Transporter Room as the destination of the turbolift on the main screen.

NOTE: At any time, you can click the 'Next' button, and the system will advise you as to the next step in accomplishing the high-level task. Clicking the 'Why' button will request the system to explain why the step it asked you to perform is necessary. Also, you can speak to the system by briefly pressing 'CTRL' and speaking into a microphone. (Saying the phrase "Computer what is next?" is equivalent to clicking the 'Next' Button, and the phrase "Why?" is equivalent to the 'Why' Button.)

The interaction of beaming down an away team involves three main sub-steps: A, B and C. A and B can be performed in any order, and C must be performed after A and B are both complete.

A

First, you must clear the transporter pads of debris. This can be accomplished by clicking the button labeled 'Debris Cleared', or by uttering the phrase "Computer I've cleared the transporter pads". After the system acknowledges success, it is necessary to board the transporter pads. This is accomplished first by clicking on any subset of the buttons labeled "01"-"06" to select which pads are occupied by the away team. Second, you must inform the system that the pads are indeed boarded by clicking the 'Pads Boarded' button or uttering the phrase "Computer I've boarded the transporter pads".

B

First, you must run a planetary scan of the destination planet. This is accomplished through clicking the 'Run Planetary Scan' button or by uttering "Computer I've scanned the destination planet". Second, the transport coordinates must be selected. This is accomplished by clicking on the map of the planet (green rectangle), thus denoting the location of the transport, followed by clicking 'Submit Coordinates'. Alternatively, if you want to transport to a specific longitude and latitude, say 127 degrees by 83 degrees, you can utter "Computer I've chosen the transport coordinates at one two seven degrees mark eight three degrees".

C

After the computer acknowledges success of steps A and B, it will ask you if you would like it to energize the transporters for you. You can answer yes by clicking the 'Yes' Button or uttering "affirmative" or "yes". Alternatively, you can ignore this prompt and energize the beams

yourself by clicking the 'Energize' Button or uttering "Computer please energize the transporters". After success has been acknowledged, you are done.

What Worked

Technically, the biggest success of this project was getting the speech subsystem working. I wanted to replicate the feel of the *Starship Enterprise* as much as possible, and on the television show, the characters routinely talk to the computer as they are using it, and operating the ship. Thus, my goal was to incorporate the speech system into my project so that you could either work through the task model using the buttons and widgets of the GUI, or by talking to the system. Additionally, whenever the system responds to the user, the text-to-speech generator speaks the results of the user's actions to the user. This is also in accordance with the way the computer works onboard the *Enterprise*. The text-to-speech engine was simple to use, just pass it a text string, and it speaks the words. However, for my project, I developed a more complex speech-recognition grammar that includes present and past tense for most of the tasks in the Task Model. The inputs to the tasks, such as Transport Coordinates, or Dilithium Crystals are specified in the grammar as well, and the phrasing of the commands has also been altered to sound more military-like and formal, as opposed to the original grammar, which was worded in a fairly informal manner.

What Didn't Work

My original vision for the task model and domain simulator for this project was more complex than the actual final version. In this original model, the simulator of the ship had various states, such as the Dilithium Chamber door's open-or-closed status, and the status of the fuel pod for the Impulse Engine. Each of the tasks in the Task Model, in addition to accomplishing certain parts of the overall task, would have a side effect within the simulated ship. These side effects could cause issues for other tasks. For instance, if I wanted to exchange the Warp Core's Dilithium Crystal, I needed to begin by turning off the Warp Core. If I abandoned that task midway through and beam down an away team, it would fail because the deactivated Warp Core wouldn't be providing power to the transporter. However, implementing this within the Task Model meant using a large number of postcondition and precondition tags. Normally, this shouldn't be a problem, but because many of the subtasks set and un-set simulation states, these postconditions resulted in the tasks being marked as "successful" before they were even attempted. Even using the sufficient modifier on the postconditions didn't solve this problem, so I decided to simplify the Task Model and so that the tasks didn't have many side effects and they simply passed input and output slots to each other for data flow and postcondition evaluation.

Future Directions

One possible future direction for this project would be to make a much more "realistic" simulation of the *Starship Enterprise*. By this, I mean that the other rooms of the ship and hardware/machinery would need to be incorporated into the application, and a much larger task model that encompasses the entire ship would need to be created, using the one from this project as a starting point. The purpose of this extended application would be primarily for entertainment, as *Star Trek* fans could "walk through" the application, and imagine themselves operating the starship from their favorite television program. Making this into a true "game" would require several technical enhancements from the current project. Firstly, the domain simulation of the *Enterprise* should be moved out of JavaScript and into Java. The reason for this is that the domain simulation would grow to a much larger size, and maintaining all of that code within JavaScript would be a troublesome task. Another important technical enhancement would be to make the user interface seem more like the ones used on the television show, to enhance realism and believability. This enhancement would be implemented with two main changes: first, the ship's consoles and computer screens would be mimics of those seen on the show. That alone would require the use of a GUI toolkit other than Java Swing to get the 'Trek' look-and-feel. Second, a 3D walkthrough application of the *Enterprise*'s interior would add a sense of immersion into the system and task model. Not only would this enhancement make the system more believable because of the graphical quality, but certain actions that are handled in the current application with mouse clicks could be improved to use video game or virtual reality technology to make the tasks of moving items around inside the ship more closely resemble their on-screen counterparts.

Another future direction of this application is more firmly grounded in reality. The task model and domain simulation could be adapted to fit a modern-day naval vessel, like a Destroyer or Aircraft Carrier. The purpose of this new work would be to create a learning tool for naval crewmen for operating the expensive and dangerous machinery on modern naval vessels without the risk of failure or mistakes in the real environment.