MAPS May Summary 2003

This month I spent a number of hours cleaning up the code. I created some new functions for repeated sections of code. I also added lots of comments and organized the numerous variable declarations. Finally, I created a number of new modules and separated more of the code into them, thus making it more logically arranged. This was necessary because the former utility module contained way too many lines of code (and functions). I also went through the beginning comment blocks on almost every function and made sure that the references were up to date.

The most exciting thing this month was the URAP meeting we had to discuss our summer plans. Stefan, Andy, Dan, Dave and I discussed how the MAPS and LifeLine projects should be better integrated into one another. We came up with a better picture of how and which parts of the projects should communicate with the others. In particular we identified that currently MAPS can talk to LifeLine but that we need to establish a conduit that allows LifeLine to communicate back to MAPS. We also identified some adjustments that would need to be made to the MAPS database schema to allow for user modeling and the error correction stuff. This is the next step for the MAPS project. We are also going to need to integrate the user modeling and error correction stuff into the gui.

Some things I did with the code:

After finally discovering a bug, I realized that the program was crashing because I was adding pictures that were bigger than 64KB, which is an absolute limit. So I added a check function and I throw up a box to the user if the picture (or sound) is too big that they are trying to add.

A question I had was whether or not the Delete Script box should stay open after someone deletes a script so that they can delete a few at a time. I guess that in general people probably won't be deleting a bunch of scripts, whereas I create some for debugging and end up deleting them. So for now I did not implement it to stay open.

Implementing a blank node to always be at the end of the script. This allows the user to click on the blank node to continue adding nodes to the end of the script. Previously, the user would have to click on the last node on the script, and this isn't as clear of a way to add to the end of the script as clicking on a blank node. The way it ended up working almost always has a blank node at the end. When the last node is partially full (i.e. has only a sound or only a picture), there may or may not be a blank node after it. In the case where there is no blank node following the partially full node, then clicking on the partially full node will put the user in append mode. If the partially blank node is highlighted and the missing media is added, it will be added to that node. Otherwise, if there is a fully blank node at the end, then selecting the blank node puts you in append mode. There could also be two blank nodes at the end. If you are in append mode, then it will just add to the first blank node (which should be highlighted). If you change to edit mode, however, then the first blank node will act as an inserted fully blank node and clicking on it will leave you in edit mode. In this case you must click on the last blank node in order to return to append mode.

Another thing I did was to make the play, play from here, and delete buttons only visible under real nodes (this is only noticeable when there are less than 5 nodes in the script). Similarly for the insert lines, only those that are between two real nodes (or at the far right for five nodes) are visible.