Adding Scripting to a Public Bulletin Board

Adam M. Fass
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
Tel: 1-412-268-4074
E-mail: afass@cs.cmu.edu

Randy Pausch
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
Tel: 1-412-268-3579
E-mail: pausch@cmu.edu

ABSTRACT
MessyBoard is a projected networked 2D bulletin board. Our experience with MessyBoard suggests that it is useful for some kinds of communication, but users have requested new behaviors such as automatic deletion of old content, automatic posting of new content from the web, and simple collaborative games.

In order to rapidly build and experiment with automatic behaviors, we have integrated the Python scripting language into the MessyBoard client and created a simple development environment. A group of scripters have built several interesting behaviors, including puzzles, magnetic poetry, automatic posting of images, and a news ticker.

KEYWORDS: Collaborative applications, MessyBoard, Python

INTRODUCTION
MessyBoard is a networked 2D bulletin board [2], similar to the Notification Collage [4], that we project on the wall of our lab as shown in figure 1. Anyone can see the content in a window on their own PC, and users can add text notes, pictures, and web links by dragging and dropping or cutting and pasting content from any application into the MessyBoard window. A central server communicates with all of the clients over the internet to keep everyone's display synchronized.

Researchers at Carnegie Mellon University and the University of Virginia have been using MessyBoard for the past eight months. Based on our informal observations, we believe that MessyBoard helps in making shared decisions, making announcements, and scheduling meetings, and that it is a fun way for members of a group to communicate with each other. [2] However, users have frequently requested that we add automated behaviors. For example, users want MessyBoard to automatically delete old content when the board is full and add new content when nobody is posting anything. Users have also suggested more sophisticated behaviors, like games or making old content fade or shrink over time.

To facilitate rapid prototyping, we have integrated the Python scripting language [6] into the MessyBoard client. We created an object-oriented Python API (the "Messy API") for manipulating the contents of MessyBoard and a simple development environment called "MessyDev" for testing and debugging Python scripts.

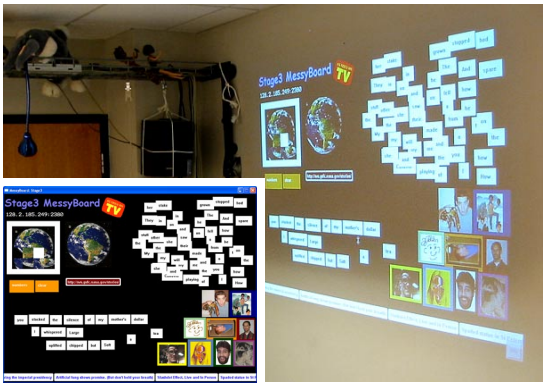

Figure 1: MessyBoard is projected on the wall of our lab. Users can run the client in a window on their own computers to add content.

EXAMPLE GAMES AND BEHAVIORS
We released MessyDev to the Stage3 research group at Carnegie Mellon University in order to see how they would use the Messy API and whether or not it was sufficient for the things they wanted to do. These scripters have used MessyDev over a two-week period and they have written several interesting behaviors:

- Sending the text of a note to Google's image search [3] and putting the first resulting picture on the board
- A Sliding Piece Puzzle that makes a puzzle out of any picture on MessyBoard
- Automatically moving objects so that they don't overlap
- Automatically posting images from a USB camera on MessyBoard at regular intervals
- A news ticker that automatically retrieves headlines and links from RSS feeds [7]

IMPLEMENTATION
In order to experiment with automatic behaviors, we needed a way to quickly prototype, test, and modify behaviors. Since MessyBoard itself began as a rapid prototype, our code base was not designed to support this kind of flexibility. Even if we could add new behaviors quickly, each change would require us to either upgrade the server, make all of our users upgrade to a new client program, or both.

Instead, we chose to integrate the Python scripting language [6] with the MessyBoard client program. The Python interpreter is open source, and the code base was designed to support this kind of integration. Python has been used suc-

cessfully as a rapid prototyping language in several other projects. [1][5][8] By integrating Python with the MessyBoard client, we allow a single client to change the behavior of MessyBoard using the existing network protocol without upgrading the server or any of the other clients.

We modified the MessyBoard client so that Python scripts can manipulate the board through a simple procedural API. This API supports the same operations that a user can perform with the MessyBoard client (creating, modifying, and destroying objects on the board) and it allows the script to register a callback function for all the events to which the user interface responds. Thus, the API allows Python scripts to behave as automated users, with the same capabilities for observation and manipulation as a human user. This approach allowed us to add automation with a minimal amount of modification to MessyBoard and preserve backward compatibility with existing clients and servers.

The simple procedural API gives a Python script all the power we thought was necessary, but we wanted an object-oriented API with more convenience functions. We built the "Messy API" entirely in Python on top of the simple procedural API. The new API provides a class hierarchy for representing different kinds of objects (notes, pictures, links, etc.) All objects have member functions to allow modification. The Messy API allows multiple callbacks for different event types, as opposed to the single callback for all events provided by the simple procedural API.

EXAMPLE: AUTOMATIC ARROW

To illustrate how the Messy API is used, we shall consider a simple example: a script that automatically places an arrow next to the newest item on the board. Our script begins by creating an arrow off screen where nobody can see it:

```
arrow = desktop.createArrow(-200, -200)
```

Next, we define a listener function to be called every time an object is created:

```
def onCreate(desktop, object):
  left, top, right, bottom = object.getRect()
  arrow.setPosition(left - 180, top)
  arrow.bringToFront()
```

The listener function gets the position of the new object and then moves the arrow to a location 180 pixels to the left of the new object. Finally, it brings the arrow to the top of the Z order so that it will slightly overlap the new object. The last step is to register our listener function so that it will be called every time a new object is created:

```
desktop.addListener(CREATE_EVENT, onCreate)
```

DEVELOPMENT ENVIRONMENT

We have created the MessyDev development environment in order to provide a convenient way to develop and test scripts. To avoid disturbing existing servers, MessyDev automatically runs a local MessyBoard server. The main window looks just like MessyBoard and has all of the same functionality. A control panel allows the user to reset the environment and execute a script file with a single click. The user can experiment by executing code from a scratch buffer in the control panel or by typing code into notes in the main window.

FUTURE WORK

Some scripters have asked for the ability to create custom user interfaces on MessyBoard that respond to mouse clicks, cursor movements, etc. Current MessyBoard scripts cannot do this because clients do not get information about clicks and cursor movements on other clients. We plan to modify the MessyBoard client to broadcast these events to other clients so that scripts can respond to them.

We plan to distribute MessyDev to more researchers in order to see what they want to build and gather more suggestions for improving the API and development environment. With its built-in network synchronization and simple scripting language, we believe that MessyDev may become a powerful tool for prototyping collaborative applications.

ACKNOWLEDGEMENTS

We would like to thank all the members of the Stage3 research group at CMU for using MessyDev and giving us feedback. This work is funded by DARPA, the Office of Naval Research (ONR), NSF and Intel.

REFERENCES

1. Conway, M. et al. Alice: lessons learned from building a 3D system for novices. Proceedings, CHI 2000, 486-493.
2. Fass, A.M., Forlizzi, J., and Pausch, R. MessyDesk and MessyBoard: Two Designs Inspired By the Goal of Improving Human Memory. Proceedings, DIS 2002.
3. Google. Google Image Search. http://www.google.com/
4. Greenberg, S., and Rounding, M. The Notification Collage: Posting Information to Public and Personal Displays. Proceedings, CHI 2001, 514-521.
5. Hinsen, K. The Molecular Modeling Toolkit: a case study of a large scientific application in Python. Proceedings, 6th International Python Conference. http://www.python.org/workshops/1997-10/proceedings/hinsen.html
6. Python Software Foundation. Python. http://www.python.org
7. RDF Site Summary (RSS). http://www.purl.org/rss/1.0/
8. Scherer, D., Dubois, P., & Sherwood, B. (2000). VPython: 3D Interactive Scientific Graphics for Students, Computing in Science and Engineering, Sept./Oct. 2000, 82-88.