A fast, interactive 3D paper-flier metaphor for digital bulletin boards

Laurent Denoue, Les Nelson, Elizabeth Churchill
FX Palo Alto Laboratory
3400 Hillview Ave, Bldg 4
Palo Alto, CA 94304 USA
{denoue, nelson, churchill}@fxpal.com

ABSTRACT
We describe a novel interface for presenting interactive content on public digital bulletin boards. Inspired by paper fliers on physical bulletin boards, posted content is displayed using 3D virtual fliers attached to a virtual corkboard by virtual pushpins. Fliers appear in different orientations, creating an attractive, informal look, and have autonomous behaviors like fluttering in the wind. Passers-by can rotate, move and fold fliers; they can also interact with fliers' live content. Flier content is streamed from a server and represented by the system on large screen displays using a real-time cloth simulation algorithm. We describe our prototype, and offer the results of an initial evaluative user study.

KEYWORDS: real-time animated user interfaces, 3D simulation of paper fliers, digital bulletin boards.

INTRODUCTION
The Plasma Poster Network (PPN) is a network of large screen, interactive, digital, bulletin boards. The PPN offers community members a convenient means to publish live, digital content (e.g. Web pages, text, images, animations, movies) to public spaces [see also 3, 4, 8]. In this paper, we describe an experimental interface to the Plasma Posters. This novel interface moves away from the desktop metaphor of the PC screen: windows resemble paper fliers posted on a virtual corkboard using virtual pushpins. Figure 1 shows an arrangement of simulated fliers.

We believe this is an inviting metaphor for public bulletin boards; people's familiarity with paper fliers and the movement in the interface invites interaction in a way that flat, static windows do not.

The virtual fliers are simulated with a real-time physics engine. Each flier is represented by a grid of particles moved in real-time under the forces of gravity and wind. This graphics simulation generates a very realistic simulation of papers attached to a corkboard with pushpins. Passers-by can manipulate each flier using simple gestures (e.g. dragging at the edges to "lift" corners and rotate them, see also [1]), and move windows by removing and replacing virtual pushpins. They can also interact with the live content, e.g. following hyperlinks, viewing animations and creating digital "scribbles" on top of the content.

Figure 1. The 3D paper flier simulation runs on a public plasma screen with a touch-screen overlay. Users can fold, rotate, add/remove pushpins; they can also interact with the live content streamed from a server.

SYSTEM ARCHITECTURE
Our implementation uses a client/server approach (Figure 2). The client hosts our physics engine responsible for simulating the paper-fliers. The server is responsible for collecting content sent by remote users and streaming its representation to the client.

Posting and streaming content
Users can post new web content, either by filling in a web form or by sending an email to the server's designated email address. For each received URL, the server opens a new web browser window and asks it to navigate to the posted URL. It then notifies the client that a new post has arrived and communicates a unique window id. Periodically, the client asks the server for the content of this window id. When contacted, the server brings the corresponding window on the foreground, captures its content and streams it back to the client on a socket as a bitmap image. The client uses this image as a texture object to represent the content of each paper flier.

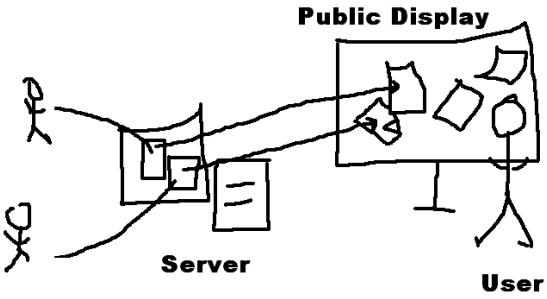

Figure 2. Remote users post web pages. The server opens a new window for each page and streams its content to the public display. The public display maps it into the 3D simulation. Users at the board can manipulate the fliers and their content.

Simulating paper fliers on the client

In a first prototype, we used the normal 2D windows API to simulate rotated and folded windows, achieving results similar to [1]. However, this implementation did not allow real-time manipulations and dynamic behaviors such as wind. Fortunately, we discovered that researchers had already designed algorithms to simulate cloth animations. These techniques were originally conceived for the garments industry [5], and have been recently improved to animate cloths on virtual characters in 3D games [6]. We adapted these techniques for the specific purpose of simulating paper fliers in real-time.

We represent the structure of a paper flier with a rectangular grid of particles. The number of particles is proportional to the size of the paper flier we want to simulate. In our experiments, we represent X pixels with (X*7) / 1024 + 4 particles. For example, a 512*256 paper flier will be represented with a grid of 7*5 particles. Using a proportional ratio between pixels and number of particles is important to provide a uniform look and feel of the paper flier, regardless of its size. For example, folds look the same on big and small fliers. The actual coefficients represent a good compromise between visual realism and speed. Fewer particles would not permit natural folds; more would prevent us from simulating 20 fliers at interactive rates.

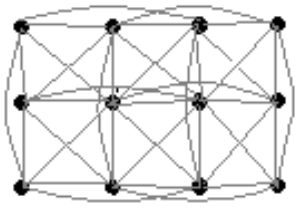

Figure 3. A paper flier is represented by a set of particles arranged in a rectangular grid. Horizontal and diagonals represent distance constraints that are used to assure that particles remain together.

At each time step (i.e. 0.02sec), the physics engine updates the position of each particle individually. We take into account a number of forces such as gravity, wind and user actions. Without further computation, particles would move independently of each other. In order to maintain the original shape of the grid, constraints need to be applied. We use 3 types of constraints: 1) structural constraints link each particle to its neighbors, 2) sharing constraints link particles diagonally, and 3) bending constraints link every other particles horizontally and vertically (Figure 3).

In many cloth simulation algorithms [5], spring models are used to simulate these constraints. At each time step, a huge system of differential equations is then solved to satisfy the nonlinear forces generated by the springs. In [7], Provot showed that solving this system is not sufficient: cloth typically exhibits a "stretchy" effect, which is unacceptable to simulate stiff paper objects. The solution is to add stiffness to the springs, but this results in stiffer equations that take a long time to solve. As a solution, Provot proposed to keep a moderate stiffness in the springs and complete the simulation by iteratively solving distance constraints between particles. Basically, particles too far apart are moved closer, and particles that are too close are moved away from each other.

For our requirement of real time simulation, we took Provot's approach to the next level: we replaced the springs with distance constraints, eliminating the computationally expensive step of modeling spring forces altogether. We found that the loop has to be iterated at least 15 times to get very good simulation of a paper flier represented by 7*7 particles. With less iteration, the paper still resembles a stretchy piece of fabric. To further boost performance, we replaced the square roots used to evaluate distance constraints with an approximation function involving only a division [6]. With this simple optimization, our OpenGL implementation in C++ simulates 20 fliers of 7*7 particles on a Pentium II with a low-end NVIDIA Vanta graphics card at interactive rates.

Attachment constraints

Paper fliers are often attached to a physical bulletin board with pushpins, staples, etc. To simulate these attachment points, we simply set the particle's mass to zero. As a result, forces become null and the particle no longer moves.

Initially, each flier is attached by 2 pushpins, one on the top left and the other on the top right of the grid. Users can remove them by clicking on them. Removing one pushpin causes the flier to animate under the force of gravity, providing very realistic swings. When both pushpins are removed, the flier falls down, although we prevent it to fall outside of the limits of the screen.

Texture mapping of streamed content

The grid of particles defines a mesh of triangles. Arranged as described in Figure 4, these triangles are sent as a series

of triangle strips to the OpenGL pipeline (n-1 strips for n rows of particles).

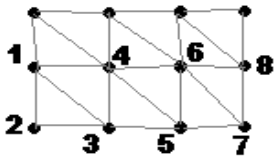

Figure 4. The grid of particles defines a mesh of triangles. Particles (1,2,3) define one triangle; Adding particle 4 defines another one, etc. Triangles are the unit used to "render" the content of each paper flier with OpenGL's texture mapping functions.

We then use OpenGL's texture mapping functions to decorate the grid with the content of the paper flier received from the server (Figure 5). As a result, users see a 3D interactive simulation of paper fliers.

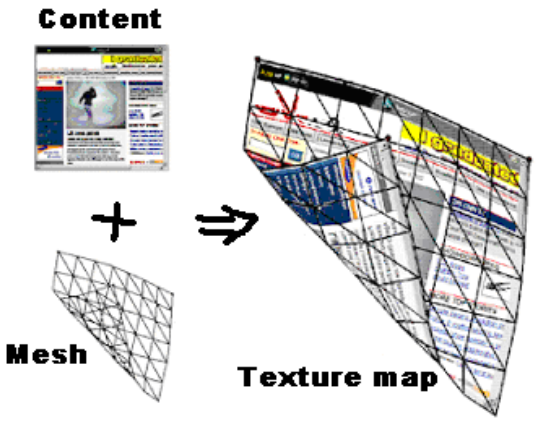

Figure 5: Streamed content image is mapped onto a 3D paper flier using OpenGL texture mapping.

USER INTERACTION

With the overlay touch-screen attached to the large screen plasma display, users can interact with the simulated fliers and their content.

Direct manipulation of a paper flier

When a user grabs a border of a flier, the closest particle's location is tied to the finger's location on the touch-screen. The particle's mass and velocity are set to zero. Finally, its Z position is incremented a little so that users feel they have lifted the flier from the screen. This technique provides very realistic folds. We also found that the bending constraints linking every other particle horizontally and vertically were very useful to help the flier flip back to a flat resting position. We also added a wind force to push back each particle against the virtual corkboard. These constraints and forces prevent the cloth from interpenetrating itself. Together, they allow us to ignore the case of the paper colliding with itself (also known as "self-collision"), further speeding up the overall simulation.

Interacting with content

Users can also manipulate the content of each flier. For example, when a user "clicks" on a hyperlink displayed on a paper flier, the flier behaves as it was displaying the actual web page, causing proper navigation. To make this happen, we use an approach similar to Virtual Network Computing (VNC) [9].

First, clicks are detected on the screen. To identify which flier was clicked, we render each one with a special texture that encodes the flier identifier and the (X,Y) position of each pixel on the texture. This rendering happens in the double buffer, thus not showing on the screen. This special texture is set up as follows. In a 24 bits per pixel screen buffer, each pixel is represented by 3 bytes: red, green and blue. We use the red to encode the flier identifier, green to encode the X and blue to encode the Y. To determine the texture coordinate of a specific point, we simply need to read the pixel color at the click location.

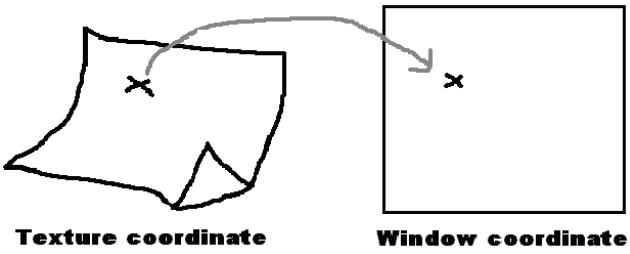

Figure 6. Texture coordinates on the 3D object are converted back into window coordinates and used by the server to regenerate the same event on the original GUI window.

We send this coordinate back to the server (Figure 6). Upon receipt, the server brings the corresponding window to the foreground of all windows and regenerates the click event using the Windows API mouse_event. The window receives the event as if it was created by a user interacting on the server machine: for example, a click on a hyperlink causes the web browser to navigate properly to the new page.

200ms after sending the events to the server, the client requests for an update of this window. Unless the user interacts with another flier, the selected flier becomes the active flier and is updated more frequently than the others. In our current implementation, we poll every 500ms for the last active flier and every second for the others. Note that when users interact with the borders of a flier (e.g. they fold the flier), we stop updating the content of all fliers. This insures that the 3D animation does not slow down. Polling of content resumes as soon as the user lifts the finger from the touch-screen.

EVALUATION AND DISCUSSION

During the development and testing of this new interface, many researchers and staff from our lab approached the bulletin board and commented on its appearance, offering anecdotal evidence that the paper flier metaphor and the animation effects were successful as attractors.

A user study was carried out using a touch screen plasma display and six pre-created fliers showing one static web page containing mainly text, one dynamic web page showing a cartoon flash animation, and 4 static pictures. After a brief demonstration of the prototype, six people were asked to freely interact with the prototype. A semi-structured interview on the design ideas and usability followed.

Users easily folded and rotated the fliers. Two users suggested that a multi-finger touch-screen would help in lifting a corner and rotating a flier underneath. This is a limitation of our current single touch-screen overlay. As they lifted a flier situated under another, two users pointed out that the flier would pass through the one in the front. This is a problem with our current simulation where we don't fully implement collision detection between fliers. An efficient solution has been proposed in [2]. These issues contribute to breaking the paper flier metaphor.

After clicking a hyperlink on a web page, 3 users noted a delay they judged too long before the page was refreshed; 2 users repeatedly clicked at the same location because they thought their action had not been registered by the system. In a desktop environment, users are visually notified when they click on a hyperlink (e.g. the hyperlink color is changed and a navigation icon animates in the Web browser). Here, this feedback does not work because of slow content updates. Although not always possible for performance reasons, one obvious solution is to refresh the content faster. Another solution is that the server sends the bounding boxes of the active areas in the original GUI window. With this information, the OpenGL client could immediately notify users with a color change or a sound before the content even gets updated.

Although its refresh rate was similar to the other web page, no one complained about the flash animation. This suggests that user expectations were different depending on whether they interacted with the content or simply looked at a streamed content. It should be noted that users had never seen the flash animation playing at normal speed on a desktop computer; furthermore, the animation was playing a cartoon, and other types of content might raise different user expectations.

Finally, all 6 users noted that it was hard to read the text on the static web page. In our case, this was not due to the flier being rotated as in [1] because the problem persisted after adjusting the flier to a normal horizontal position. It most certainly had to do with the fact that the web page was designed to be rendered in a desktop environment and not a big public display.

CONCLUSION AND FUTURE WORK

Our paper flier metaphor appears to be visually appealing and intriguing to our user population, encouraging us to further study this new interface on public displays. However a number of design improvements have been identified. First, we need a better collision detection algorithm to prevent fliers to go through each other. To make the system more responsive to user interaction, we are implementing a more intelligent streaming server: only the updated regions of a window are sent back to the client, not its entire area (see [9]). Finally, a multi-finger / multi-hand touch-screen would be very beneficial to support our metaphor. Our current software implementation can easily accommodate multiple touch events and we are looking into using existing multiple-point touch-screen prototypes.

ACKNOWLEDGMENTS

The authors thank Jim Baker for supporting this research.

REFERENCES

1. Beaudouin-Lafon, M. Novel interaction techniques for overlapping windows. In Proc. of UIST 2001, 153-154.
2. Bridson, R., Fedkiw, R., Anderson, J. Robust Treatment of Collisions, Contact and Friction for Cloth Animation. In Proceedings of SIGGRAPH 2002, 594–603.
3. Fass, A.M., Forlizzi, J., Pausch, R. MessyDesk and MessyBoard: Two Designs Inspired By the Goal of Improving Human Memory. In Proc. of DIS 2002, 303-311.
4. Greenberg, S. and Rounding, M. The Notification Collage: Posting Information to Public and Personal Displays. CHI Letters 3(1), 515-521, 2001.
5. House, D.H., Breen, D. Cloth Modeling and Animation, ISBN 1-56881-090-3. Published by A K Peters,2000.
6. Jakosen, T. Advanced Character Physics. In Proceedings of GDCONF 2001, Game Developers Conference, 2001.
7. Provot, X. Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior. In Proceedings of Graphics Interface '95, 147-154.
8. Snowdon, D.N., Grassot, A. Diffusing Information in Organizational Settings: Learning from Experience. In Proc. of CHI 2002, pp. 331-338
9. Virtual Network Computing (VNC), http://www.uk.research.att.com/vnc/winvnc.html