Technical Report TR05-005

Department of Computer Science
Univ. of North Carolina at Chapel Hill

Facetop: Integrating Artifact and User in Synchronous Paired Collaborations via Semi-Transparent Video

David Stotts, Karl Gyllstrom, Dorian Miller, and Jason McC. Smith,

Department of Computer Science
University of North Carolina
Chapel Hill, NC 27599-3175

stotts@cs.unc.edu

March 3, 2005

Facetop: Integrating Artifact and User in Synchronous Paired Collaborations via Semi-Transparent Video

David Stotts, Karl Gyllstrom, Dorian Milller, and Jason McC. Smith
Dept. of Computer Science
Univ. of North Carolina at Chapel Hill
{stotts, gyllstro, dorianm, smithja}@cs.unc.edu

Abstract. The Transparent Video Facetop is a novel user interface concept that supports not only single-user interactions with a PC, but also close pair collaborations, such as that found in collaborative Web browsing, remote medicine, and in distributed pair programming. A live, full-screen semi-transparent video image of the user is mirrored horizontally and composited into the framebuffer with the desktop; for collaborative applications, we composite the images of two users. On shared applications (single windows, or entire shared desktops) when one user points, the other user sees his/her video image pointing to the same screen artifact. The Facetop allows a distributed pair to experience some of the facial expressions and face-to-face communications aspects that are present in *in-situ* collaborations. It also allows members of a distributed pair to point conveniently, quickly, and naturally to their shared work, in the same manner (manually) that they do when seated side-by-side. We discuss the architecture of Facetop, our user studies, and domains of application, including support for hearing-impaired users via signing.

Overview

Communication is the central component to collaborative development environments. Many of these environments include or rely on some application level facility to assist users in this interaction. The predominant facilities in this area are email, interactive-chat clients, and video-teleconferencing, each with particular advantages and disadvantages that dictate which roles and requirements they are most suited for.

In this report we present a novel technique for tightly integrating video teleconferencing with desktop artifacts that users wish to collaboratively work with. Called the Transparent Video Facetop, our methods have proven effective in enhancing the experience of collaborating pairs working synchronously on tasks that, in person, require finger pointing and manual gesturing. As a side benefit of the integral use of video, Facetop also supports hearing-impaired users in distributed collaborations by allowing effective signing.

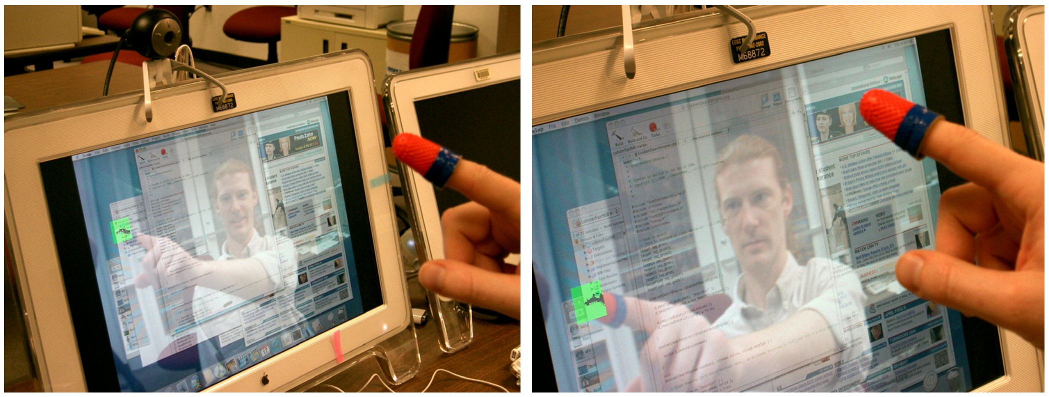


Figure 1: Facetop physical setup, with iBot video camera

Basic Facetop concepts

The transparent video Facetop is a novel enhancement of the traditional WIMP user interface, so nearly ubiquitous on today's computers. In the Facetop, the user sees him/her self as a "ghostly" image apparently behind the desktop, looking back at the icons and windows from the back. Instead of a traditional desktop, we see a "*face" top*. This self-image is used for visual feedback and communications both to the user as well as to collaborators; it is also used for desktop/application control and manipulation via a fingertip-driven "virtual mouse".

Figure 1 shows the physical setup for a computer with a Facetop being displayed on a monitor. Note the video camera sitting on top the LCD panel pointing back at the user; in our current work we use a $100 Sony iBot, giving us an image that is 640 x 480 pixels of 24-bit color, captured 30 frames per second. The Facetop video window shows the PC user sitting at his/her workspace; we reverse the image horizontally so that when the user moves a hand, say, to the left, the image of the hand mirrors this movement on the screen. In software, and using a high-performance 3D-graphics video card, we make the video window semi-transparent and composite it with the desktop image itself.

Once we have the full screen video with transparent image compositing we get the illusion of the user watching the desktop from behind. The mirrored means that if the user physically points towards an icon on the desktop, the Facetop image points to the icon as well (with proper spatial calibration of the camera and user locations). Using image analysis techniques we then track the user's fingertip in the backing window, and optionally drive the mouse from this tracker. Figure 1 shows this finger tracking using a colored rubber thimble. The user can then manipulate the desktop of a *projected* computer, for example, from his seat while successfully communicating the areas of interest on the screen to others watching the projection.

Related Work

Facetop uses supporting technology from several areas: collaboration theory and systems, video analysis, user interfaces, graphics. In many ways, the novelty of Facetop is not new graphics, or new image analysis, or new collaboration theory, but that combining a few simple methods in one integrated application is remarkably enabling, as the Web was enabling with known networking mechanisms. We summarize some of these related projects here. Englebart first demonstrated use of analog video of the user for collaboration in 1968 in his NLS/Augment system, but his work did not include integration of the video image(s) with the work as we do in Facetop, and it was not digitally manipulable. It was, in essence, the first video conferencing system, limited to 2 users.

Pointing in Collaborative Applications

Several systems have dealt with the issue of two users needing to provide focus (to point) at different, or independent, locations on a shared screen. The common solution is to provide two mouse pointers and let each user control his/her own independently. Since one of our Facetop studies has been in support of distributed air programming (explained below), we mention specifically a two-mouse system by Hanks [21], built to support pair programming over the Internet. This approach is fundamentally different from using human device fingers to naturally manually point as in Facetop. The two-mouse approach lacks the integrated facial expressions and the enhanced sense of presence that video provides.

Transparency, UI, Video, and Gestures

Many prior research projects have experimented with different aspects of techniques we have unified in the Facetop. Several researchers have made systems with transparent tools, windows, pop-ups, sliders, and widgets that allow see-thru access to information beneath; these are primarily used for program interface components [8,11]. Other researchers have studied the ability of computer users to work effectively in the presence of such transparent overlays. Harrison [26] found that 50% transparent palettes can greatly improve workspace visibility without degrading icon selection performance. Gutwin et al. [25] have shown that users can perform targeting tasks well with 25%, or even 10% opacity, depending on the complexity of the workspace data.

Many systems have some user embodiment and representation in them (avatars), especially in distributed virtual environments such as that of Benford et al. [10], but these tend to be generated graphics and not live video. Giving your PC "eyes" is a growing concept, as is illustrated by a 2001 seminar at MIT [12]. A system being developed in Japan [9] uses hand activities as signals to programs; the system uses silhouettes to make recognition easier and faster. Our ideas for fingertip gesture control in the Facetop are related to the many efforts under way to recognize pen gestures and other ink-based applications; the Tablet PC based on Windows with ink is now commercially available from several manufacturers. They are also related to efforts in the past to recognize human facial features and motions; a few examples include face tracking and facial recognition [7,16,27,]. Our work in Facetop makes use of such tracking techniques, but is not specifically (for now) creating new ones.

Collaborative systems, distributed workgroups

There have been far too many systems built for graphical support of collaboration to list in this short paper. Most have concentrated on synthetic, generated graphics, video conferencing enhancements. *ClearBoard* [4,23] is one system that is especially related to our research in that integration of video image and work was a goal. ClearBoard was a non-co-located collaboration support system that allowed two users to appear to sit face to face, and see the shared work between them. The ClearBoard experiments showed that face-to-face visibility was enhancing to collaboration effectiveness. However, the workstations required were expensive and used custom-built hardware. One of the advantageous points of the Facetop is its use of cheap and ubiquitous equipment. Clearboard showed each user the other; Facetop allows each user to see both, and this arrangement has several useful applications we discuss in following sections. Clearboard was also limited in being usable only in front on the special monitor; Facetop, with its user self-image, can be used in projected environments and other scenarios where the user has no physical monitor to reach out and touch.

The novelty of Facetop is using transparency to tightly integrate the user video image with the desktop work. Toolkits such as the *videoSpace* kit [22] are available for producing application like Facetop, though the heavy processing and transparency in Facetop make Apple development most efficient with current platforms.

One last project we use results from is BellCore's *VideoWindow* project [5]. In this experiment, two rooms in different buildings at BellCore (coffee lounges) were outfitted with video cameras and wall-sized projections. In essence, an image of one lounge was sent to the other and projected on the back wall, giving the illusion in each room of a double-size coffee lounge. The researchers discovered that many users found the setup to be very natural for human communication, due to its size. Two people, one in each room, would approach the wall to converse, standing a distance from the wall that approximated the distance they would stand from each other in face-to-face conversations. The conclusion:

Video, when made large, was an effective and convincing communication tool.

We leveraged this finding in creating the dual-head Facetop that we use for synchronous, collaborative activities.

Two-head Collaborative Facetop

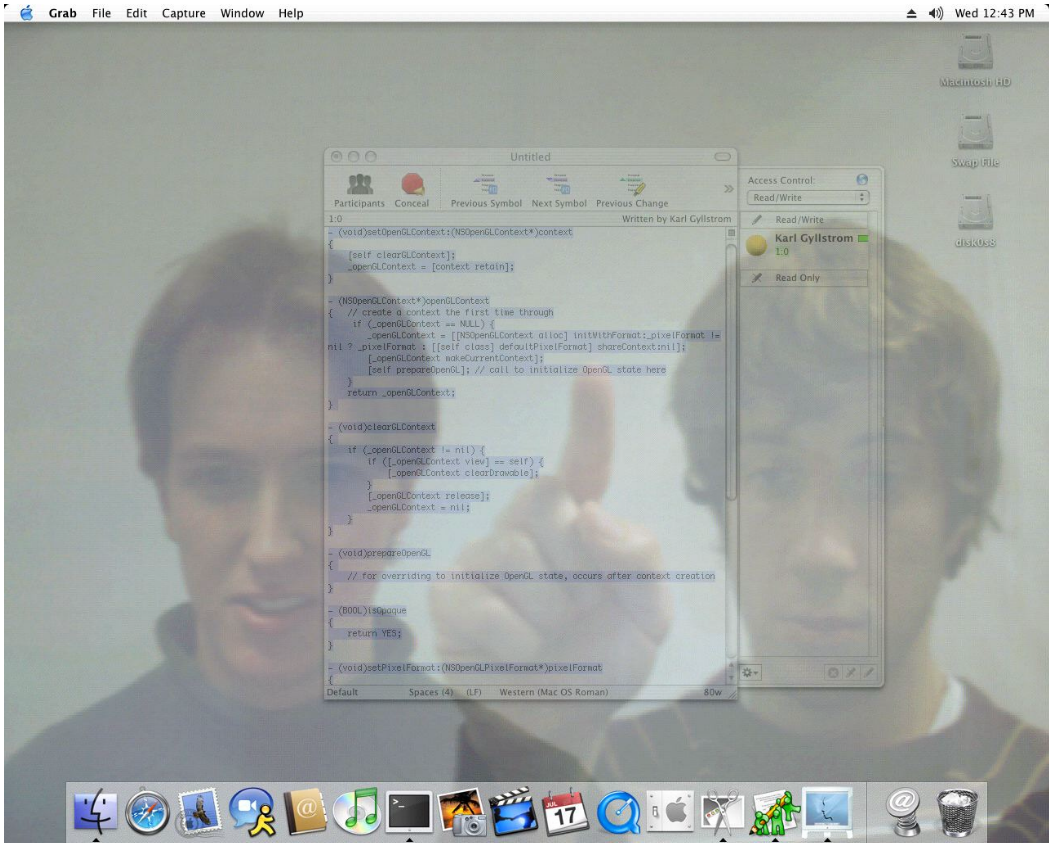
Though the presentation so far has been in the context of a single-user PC interface, an more interesting domain of application for the Facetop is in collaborative systems – specifically in systems for supporting synchronous paired tasks. We have been investigating a two-head Facetop for the past year for use in distributed form of pair programming (dPP), a collaborative software development practice used in Extreme Programming [1,2,3]. This investigation is an extension to earlier studies designed to see if distributed pairs could pair program effectively communicating over the Internet rather than sitting side-by-side [6,15].

In previous dPP experiments, programmers worked as a pair using COTS software, including the shared desktop program *pcAnywhere* (Intuit) and *Yahoo*

Messenger for audio. The *pcAnywhere* application provides a shared workspace, so that the two programmers are effectively working on a single host computer, and each sees exactly the same desktop as they would sitting side-by-side at the host PC. *Yahoo messenger* provides voice communications, and occasional text exchange.

Our experiments found that programming pairs working synchronously in this environment were as effective in the distributed setting as they were co-located. In post-experimental interviews, teams consistently told us:

- They missed facial expressions and the sense of presence that comes with face-to-face work;
- They wanted a way to point at the shared work they were discussing (as they do sitting side-by-side).



1:0 Written by Karl Gyllstrom

```
- (void)setOpenGLContext:(NSOpenGLContext*)context
{
    [self clearGLContext];
    _openGLContext = [context retain];
}

- (NSOpenGLContext*)openGLContext
{   // create a context the first time through
     if (_openGLContext == NULL) {
        _openGLContext = [[NSOpenGLContext alloc] initWithFormat:_pixelFormat !=
nil ? _pixelFormat : [[self class] defaultPixelFormat] shareContext:nil];
        [_openGLContext makeCurrentContext];
        [self prepareOpenGL]; // call to initialize OpenGL state here
    }
    return _openGLContext;
}

- (void)clearGLContext
{
    if (_openGLContext != nil) {
        if ([_openGLContext view] == self) {
            [_openGLContext clearDrawable];
        }
        [_openGLContext release];
        _openGLContext = nil;
    }
}

- (void)prepareOpenGL
{
    // for overriding to initialize OpenGL state, occurs after context creation
}

- (BOOL)isOpaque
{
    return YES;
}

- (void)setPixelFormat:(NSOpenGLPixelFormat*)pixelFormat
{
```

Figure 2: Dual-head Facetop for synchronous collaboration

The Facetop provides potential solutions to both of these problems via its video capabilities, which are very different from basic webcam-style techniques. The video image is large, and frame rates run from 15 to 30 fps, showing facial details and fine motor movements of the fingers and lips. The video image is also tightly and seamlessly integrated with the shared workspace via transparency, thereby eliminating the "dual" nature of video teleconferencing solutions. Users do not have to switch their attention from desktop, to video, back to desktop.

For the dual-user Facetop, we have built a setup that has both video streams (each collaborator) superimposed on a shared desktop, illustrated in Figure 2. Each user sits slightly to the right so that the two heads are on different sides of the frame when the two streams are composited. Collaborating users continue, as before, to communicate audibly while using the Facetop via an Internet chat tool like Yahoo messenger. Each user now can see where the other points in the shared workspace.

User self-view + transparency

The Facetop combines and extends work from several different domains of computing research. Gesture-based computer controls have existed for a while, for example. The Facetop, however, is unique among these for two reasons. The first is transparency: the Facetop blends the traditional desktop with a video stream of the user, mirrored and made semi-transparent. The second is the video cues the user image gives: the user is *in* the desktop, as live background wallpaper, rather than making detached gestures apart from the image of the desktop. These video cues have proven very effective at giving fine and intuitive control of the cursor to the user in various tasks and applications we have experimented with.

We allow the user to dynamically control the transparency level of the Facetop window, altering it from fully opaque to fully transparent during execution for varying useful effects. We can completely mask the desktop by making the Facetop window fully opaque, as in Figure 3 (right). We can similarly set the Facetop window to full transparency; in this form, the desktop under it shows through fully and little to no user video is visible, giving a traditional desktop appearance. Figure 3 (left) shows a nearly transparent Facetop.

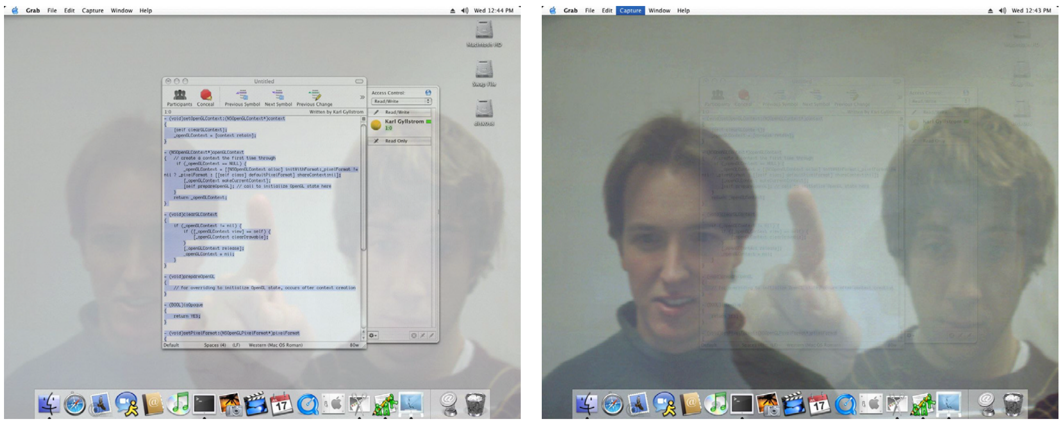


Figure 3: Varying levels of transparency in 2-head Facetop

Most uses for the Facetop will involve a semi-transparent Facetop setting, as shown back in figure 2, giving a mix of user video image and desktop application window content on the screen. In this mix, the user's finger can clearly be seen pointing at various text lines in the editor window. Each user has transparency level controls that are independent of the settings chosen by the partner. A user can set the level (from opaque to transparent) of each video image separately (self and partner image), as well as the level of the desktop. In this way, each user can get different communications effects. If both user images are set to high visibility, and the desktop low, Facetop is a form of video conferencing system. Bring the desktop up to visible, and the unique integration of user image with shared work happens, allowing pointing and discussion. Some users may wish not to see themselves (and can still effectively point by finger tracking and watching the mouse pointer) and have only the partner image visible on the desktop. Fade all video out completely and you have the traditional desktop.

Facetop features and capabilities

The Facetop as a concept works fine on a PC with any display technology -- a monitor, a projector, an immersive device -- but its unique aspects are most pronounced and most effective in a projected environment. In fact, the concept of background user video as visual cues for control and communication came about when our research group was discussing other work using a projected PC and trying to visually interpolating along the line formed by a person's pointing arm. We here summarize the various capabilities and optional behavior a Facetop user can employ in support of paired collaborations.

No camera registration issues

Facetop does not need any registration of the cameras. This is unique among applications using video for selection of items and identification of locations within the frame. A better to way to express this feature might be to say that in Facetop we have dynamic registration, done by the user 30 times a second. This is because the user employs the self-image as visual feedback for controlling where the hand is placed in the camera field of view in order to make the screen image line up with desktop items. The camera can even move during use and there will be no interruption in service; the user simply adapts almost instantly to the shifted video image and moves his/her hand to the proper location in 3-space.

Finger tracking and Finger gestures for control

One interesting feature in the Facetop is finger tracking, a function that can be turned on or off. The Facetop has great value as a pure communication tool, especially in collaborative applications like dPP, via finger pointing and facial expressions alone, even if no finger tracking and mouse control is done. However, tracking and mouse control does add some interesting and useful capabilities for users that wish to use them. Figure 1 shows finger tracking on.

In the Facetop, the user's fingertip can function as a mouse driver, so applications like browsers can be controlled with finger motions rather than the mouse. The tracker provides the <x,y> location information for moving the mouse; the more difficult problem is designing and implementing gestures that can serve as mouse clicks, drags, etc. When tracking in on, moving the mouse pointer during tracking can be toggled on and off separately. The search neighborhood can be viewed as a box on the screen at the fingertip (see figure 1 for example); this mode shows in the box the filtered bits that the tracker actually works with, rather than showing the source image.

We do finger gestures from video analysis alone, not via some active device like a glove or phantom. The work most closely related to our Facetop video analysis is from the image-processing lab of Tony Lindberg in Sweden. Researchers there have developed tracking algorithms for capturing hand motions rapidly via camera input, and have developed demonstrations of using tracked hand motions to interact with a PC [13,14]. One application shows a user turning on lights, changing TV channels, and opening a PC application using various hand gestures while seated in front of a PC. Another experiment shows careful tracking of a hand as it display one, two, and three fingers, and scales larger and smaller. A third experiment uses hand gestures in front of a camera to drive the mouse cursor in a paint program.

The missing concept in Lindberg's work (and in other hand-gesture work), one that we are exploiting for Facetop, is the immersion of the user *into* the PC environment. In Lindberg's work the user is still an object separate and apart from the PC being interacted with. In the Facetop, the user is given the illusion of being part of the environment being manipulated. We think this immersion gives very useful and important visual cues that are absent in earlier gesture experiments. These visual cues give the feedback needed by a user to fine-grained control of the desktop, and also give a more naturally learned and manipulated interface.

The Facetop tracker gives us mouse-pointer location and causes mouse motion, but the harder issue is how to click the mouse. The method we currently use is occlusion of the fingertip. When the mouse pointer has been positioned, the user makes a pinching fist of sorts, hiding the fingertip in the hand or between the other fingertips. The tracker notes the loss of the tip, and begins a timer. If the tip reappears (user raises the finger) in a ½ second, a single-click mouse event is generated at the mouse pointer location. If the tip remains hidden for between ½ and 1 second, a double-click event is generated. User studies (discussed in a later section) have so far shown that this motion is not hard to learn and even master. It is sufficient to open/close windows, drag them, resize them, select links in Web browsers, and even position the mouse between characters in documents.

Automatic video fade in/out

One particularly useful gesture the tracker performs is automatic detection of when the fingertip enters and leaves the camera field of view; this *autofade* mode can be turned on or off. When autofade is on, if the fingertip leaves the camera view (like would happen is a pointing user drops his/her arm), then the Facetop video image fades away (to some preset level, perhaps completely off). When the user raises his/her arm again, and the fingertip is again detected in the camera view, the Facetop video is faded back in to the previous user-set transparency level for use. This mode helps limit video clutter on the screen during a collaboration; it is useful in single-user mode for applications like *Powerpoint* presentations projected on a screen. With appropriate finger gestures, a user can point at slide content, and also flip to the next slide. By dropping the hand from view, the *Powerpoint* slide alone shows on the screen, with no user video.

3-D tracking, and audio cues

Our fingertip tracker also provided course-grained information in the Z direction, allowing Facetop to generate events based on how close or how far the finger is moved towards or away from the camera plane. We currently do this by estimating the pixel area covered by fingertip in each frame. As it increases, we consider the finger to be moving towards the camera; as the area shrinks, the finger is correspondingly moving away.

3-D information like this can be used to do mouse clicks, for example. A pushing motion, where the finger is moved towards the camera a sufficient amount in a short enough span of time, can be interpreted as the click event. A hold can be interpreted as not moving the finger back right away. We have also experimented with using Z-direction information to place directional audio cues into a user's headphones. In this way we are beginning experiments with giving sight-impaired users audio cues as to where a collaborating partner is focusing attention on a workspace.

Voice controls

Using the voice recognition engine built into Mac OS X, we have made several of the parameters in Facetop settable via voice command. For example, saying "go face" will cause the video to become mostly opaque, hiding the desktop and making Facetop mostly a teleconferencing tool. Saying "go top" causes the opposite, making the video mostly transparent and allowing mostly desktop to show. Saying "go up" or "go down" changes the alpha level (transparency) up or down 5% respectively. Saying "go full" makes Facetop full screen (essentially starting the video overlay; saying "go small" parks the video in a small window at screen bottom. These commands allow the most common manipulations to be done without stopping the use of hands.

Chalk passing, floor control

Passing locus of control among collaborators in a shared application is an important issue, called *floor control*, or *chalk passing*. The user who has "the chalk" is the one who drives the mouse and initiates activities. Our tracker algorithm has a loss recovery mode that produces an interesting chalk passing behavior in the dual-user Facetop. When tracking, if the user hides the fingertip so the tracker cant track it, the tracker searches the image for it to reappear. If the other use happens to lift a fingertip into view, the tracker will find that one, and the mouse cursor passes to the other user. The two users can pass control back and forth simply by "tossing" the mouse pointer fingertip to fingertip.

User Evaluations

Controlled user evaluations are still ongoing, but we have some usability results to report from our first experiments. To date we have had 15 users try the basic Facetop to determine if live background video is a viable, usable concept as an interface for manipulating the PC environment. We set up the Facetop up in a room with white walls so that there would not be a busy background to add visual clutter to the screen image.

As might be expected, arm fatigue is a problem for continuous use of the fingertip-based mouse feature. For browsing a hypertext, this is not a major issue, as much time is spent reading vs. actually manipulating the screen. Users drop their arm during these quiescent periods, and then raise it to point when ready to navigate more. The video image on-screen gives the visual cues needed for nearly instant positioning of the mouse pointer directly where needed.

Another problem reported by several users is visual clutter. Most users adapted quickly and comfortably to the moving image as background "wallpaper"; transparency was set at different levels by different users, and there did not seem to be a preferred level of mixing of desktop with user-image other than to say that both were visible. The human eye/brain is able to pay attention (or ignore) the face or the desktop respectively, depending on the cognitive task – depending on whether the user wants to read the screen contents or to communicate (in the two-head version).

Users were queried specifically as to visual clutter or confusion. A few objected, but most found the adjustability of transparency fine-grained enough to get to a level where they were not distracted or hindered in using the desktop.

We also created a networked tic-tac-toe game for usability trials of the dual head version and had 11 pairs of users try it. The users were a class of 8-grade students who came to the department for research demonstrations. Five of the users took less that 5 minutes to become facile with the interface, learning to move and click the mouse well enough to Web browse. All users were able to successfully play the game (which involves clicking on GUI buttons) in the 30 minute time-frame of the trials.

Distributed Pair Programming

Previous research [17,19] has indicated that pair programming is better than individual programming in a co-located environment. Do these results also apply to distributed pairs? It has been established that distance matters [18]; face-to-face pair programmers will most likely outperform distributed pair programmers in terms of sheer productivity. However, the inevitability of distributed work in industry and education calls for research in determining how to make this type of work most effective. Additionally, Extreme Programming (XP) [1,2] usually has co-located pairs working in front of the same workstation, a limitation that ostensibly hinders use of XP for distributed development of software.

We have been investigating a video-enhanced programming environment for the past year for use in distributed Pair Programming and distributed Extreme Programming (dPP/dXP) [1,2]. Pair programming is a software engineering technique where two programmers sit at one PC to develop code. One types ("drives") while the other reviews and assists ("navigates"); roles swap frequently. The benefits of pair programming are well known in co-located situations [3]; we have been exploring if they remain in distributed contexts [6,15].

We had five of the pairs involved in past dPP experiments (with audio and shared desktop only) try the Facetop environment for small pair programming "shakedown" tasks. Since all had tried the earlier environments, the trials were designed to see if the "video made large" features in Facetop overcame the lack of pointing ability and lack of facial expressions reported by these teams before (the lack of whiteboard they reported is still being investigated, and is discussed in the next section).

All teams were quite comfortable using the Facetop, and did not consider visual complexity or clutter an issue. We suspect this is due to concentration on programming focusing the attention on the various text windows of the desktop. All dPP teams were able to complete small programs with no problems.

They also reported setting varying levels of user image transparency to suit personal taste. Given that the video images can be completely faded out, leaving nothing but desktop, the current Facetop is "no worse" than our previous audio-only environments. However, no teams chose to completely fade out the video and use audio only. All teams left the user images visible to some extent and did use the video to point to code being discussed.

In post-trial interviews, the overall impression was that Facetop was an interesting improvement over the audio-only dPP environment used before. Each team was asked "if you were to do a longer dPP development, would you prefer to use Facetop or the original audio-only environment?" All teams expressed a preference for Facetop.

These simple usability trials do not reveal if the preference for Facetop was emotional or qualitative only, or if the added video and sense of presence increases programmer effectiveness. We find these early usability trials compelling enough, though, to start larger, controlled experiments to see if Facetop can have an impact on quantitative aspects of software, such as design quality or error counts.

Universal access for hearing-impaired collaborators

For computer users with hearing impairments, we are experimenting with the Facetop video being used for support of signing and lip reading during synchronous collaborations. Programmers with audio impairments, for example, can do side-by-side pair programming with current technology, but they cannot participate in dPP using the audio-only environments we first experimented with.

As a basic trial of the video capabilities in Facetop to support signing, we set up two hearing-impaired users in the two-head Facetop. We separated them physically, and then gave them a shared desktop containing a Word document. We asked them to have a conversation about the document and about their experiences using the system. The trial ran for about 15 minutes, and both users expressed complete satisfaction with their ability to communicate via signing in the Facetop video. We find this especially interesting, since the Facetop makes possible something that was heretofore not possible (hearing-impaired users participating effectively in distributed collaborations).

Ongoing studies

We are currently collecting data in an experiment comparing Facetop use to dual-mouse-pointer applications with traditional webcam-style video. Video was one issue discussed at a workshop on distributed pair programming at XP/AU 2002. This workshop was attended by over 30 people, many of whom had tried some form of distributed pair programming and were working on tools to improve the effectiveness of such activities. The consensus on video was that "webcam" style, postage stamp video – small image and low frame rate – was of little value in enhancing communications or sense of presence in a distributed pairing. However, it was felt that video, if large enough and real enough, was of potential value and worth further research. This has been a driver in our studies.

Our experimental setup has two non-colocated humans doing two tasks: playing checkers, where the humans must collaborate on a strategy for playing against the computer; and assembling a jigsaw puzzle, where one user has control of the pieces, and the other assists with suggestions. Each pair will do these tasks on a shared desktop. In one trial they have the Facetop for collaboration support; in the other trial, they have two mouse pointers and webcam-style video in a moveable window. Both setups have audio.

We are also running these trials with two different groupings of users: pairs of people with fully functional senses, and pairs of people that are hearing impaired and must communicate via signing. Though we have several trials completed, the full data not be in for another month as of this writing.

Technical Issues in System Implementation

Facetop is implemented on a Macintosh platform. The two-head version runs on a peer-to-peer gigabit network between two machines to get the very high bandwidth we need for 30 fps video stream exchange. Current experimental versions built for best-effort use of the switched Internet give about 20 frames a second, which is useable for signing but not for accurate lip reading. The implementation is elegantly simple, and potentially ubiquitous due to its modest equipment needs. Facetop uses a $100 Sony iBot camera, and runs with excellent efficiency on an Apple Powerbook, even when processing 30 video frames a second. No supplemental electronics are needed for wearing on the hand or head for tracking or gesture detection. Facetop is minimally invasive on the user's normal mode computer use.

The current prototype was generated with a Macintosh G5 with a high-end graphics card to perform the image transparency. It is implemented on MacOS X by taking advantage of the standard Quartz Extreme rendering and composition engine. QE renders every window as a traditional 2D bitmap, but then converts these to OpenGL textures. By handing these textures to a standard 3D graphics card, it allows the highly optimized hardware in the 3D pipeline to handle the compositing of the images with varying transparency, resulting in extremely high frame rates for any type of image, including video blended with the user interface.

The video application, with tracking capabilities, is run in a standard MacOS window, set to full screen size. Using OpenGL, setting the alpha channel level of the window to something under 0.5 (near-transparency) gives the faint user image we need. Though we have been speaking of the Facetop as giving the user an illusion of being "behind" everything, the Facetop is actually the topmost application window on the Mac desktop, and it passed through any mouse events it does not wish to handle.

Finger tracking, jitter, and mouse control

Using finger tracking as a pointing device has some limitations. In order to get fast response and still have plenty of processor left to run desktop applications, we do tracking by looking for a colored thimble (rubber tip from office supply, painted with acrylic) the user wears on one or more fingers. We have a color calibration dialog that allows fine definition of the color under specific lighting conditions. The video image is then chromakeyed on that color, leaving a black-and-white image to search (where the tip is black). The highlighted box in figure 1 around the finger is the region the tracker operates in, and in this view we show the actual data bits being examined (a debugging mode that can be toggled on and off). As the user moved the hand around in view of the camera, the tracker constantly finds the center of mass of the fingertip and reports an *<x,y>* coordinate location for each frame.

Compared to a mouse cursor, the finger tip takes up many pixels on screen, even when far away. As such, it is difficult to emulate the pixel resolution of the mouse without additional functionality. Simply using the center of the tip can create difficulties when the orientation of the tip changes - a common occurrence in normal human motion – causing an erratic cursor that jumps to different points as the center of the tip changes slightly.

The method we employ to assist such resolution and jitter problems is a smoothing function on the mapping between the tracked finger and the actual cursor. The mouse usage is separated into two modes: projectile and sensitive. When the user is moving his hand quickly to another portion on the screen (projectile), the cursor is simply moved to the middle of the pixels corresponding to the tip. When the user is not moving quickly (for sensitive tasks such as clicking), the position the cursor is moved to is an average of the last several samples of the tip location, allowing for a smoother appearance of the cursor that is more easily controlled by the user. The transition between these modes is triggered by comparing the distance between the tracked object in contiguous samples to a threshold.

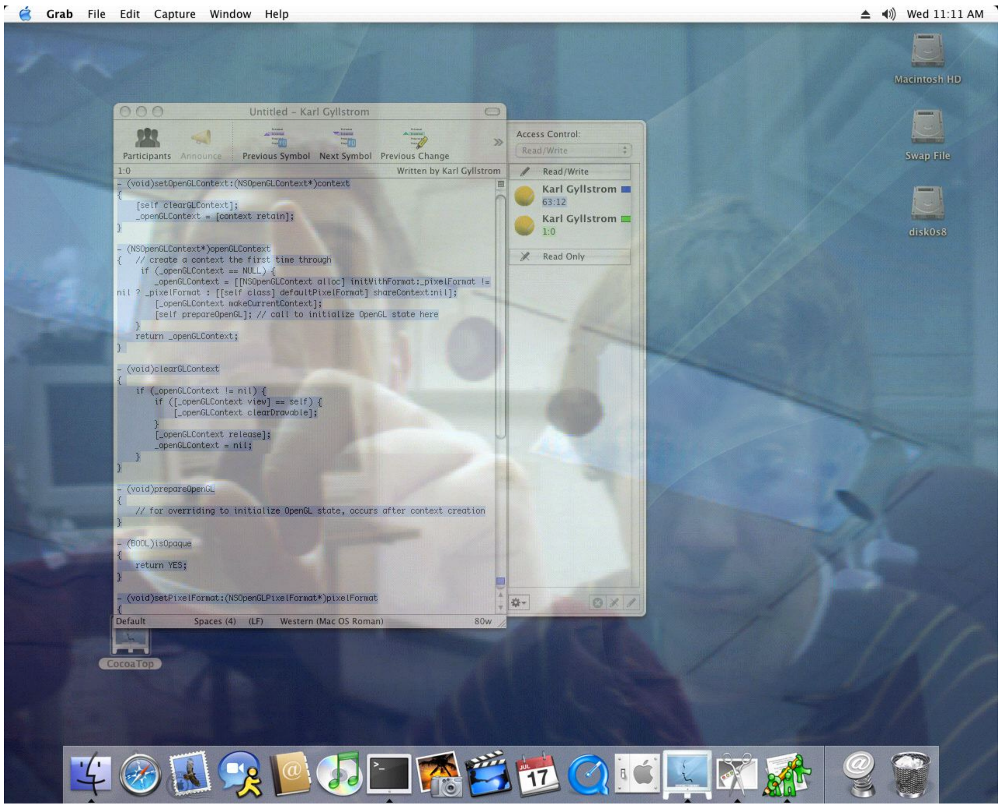
Reducing visual clutter

Visual clutter seems an obvious problem (though our early usability trials have not shown users to object to it), especially in the two-head collaborative version of the Facetop interface. The human brain is good at concentrating on the signal of interest (the desktop, the face) and ignoring the other. While semi-transparent video allows for a more powerful integration of video and one's computer desktop, it can sometimes introduce noise and ambiguity [24] into the display. Items in the background, such as walls, pictures, and books, are rarely useful in a video-teleconferencing session, but often add noise and clutter to the desktop when semi-transparently overlaid.

Most Facetop-based applications will be enhanced, and the potential visual confusion reduced, by the user sitting against a neutral colored, plain background with neutral clothes, like shown in figures 2 and 3. However, many uses for Facetop will not be in ideal circumstances, giving images more like the visually cluttered one shown in figure 4. Here the users are overlapped, the laboratory background is complex and apparent, and the desktop itself has figured wallpaper.

We would like Facetop to adapt to such noisy environments when they are unavoidable. We have taken several measures to mitigate the influence of background clutter. The simplest one involves exposing a control over the transparency value of each window. Users can move a slider interactively to set the transparency as needed. Another technique we use for managing visual clutter was mentioned earlier: the *autofade* mode, where the Facetop tracker recognizes when the fingertip enters or leaves the video frame and correspondingly raises and lowers the video levels.

We have also implemented a more dynamic transparency control that uses certain input events as signals of the user's current task within Facetop, and adjusts transparency to fit that task. The first control uses the time since the user's last keyboard or mouse event to distinguish whether a user is editing or conversing. With a longer time span since typing or clicking, it is likely the user is talking to his collaborator, and the video is made more opaque. Conversely, a short time span indicates that the user is editing and should see the document more clearly, so the transparency of the video is increased. Voice can also be used as a signal of the user's current task. When either user is speaking, we reduce the transparency of each user's overlaid video so they can see each other more clearly.

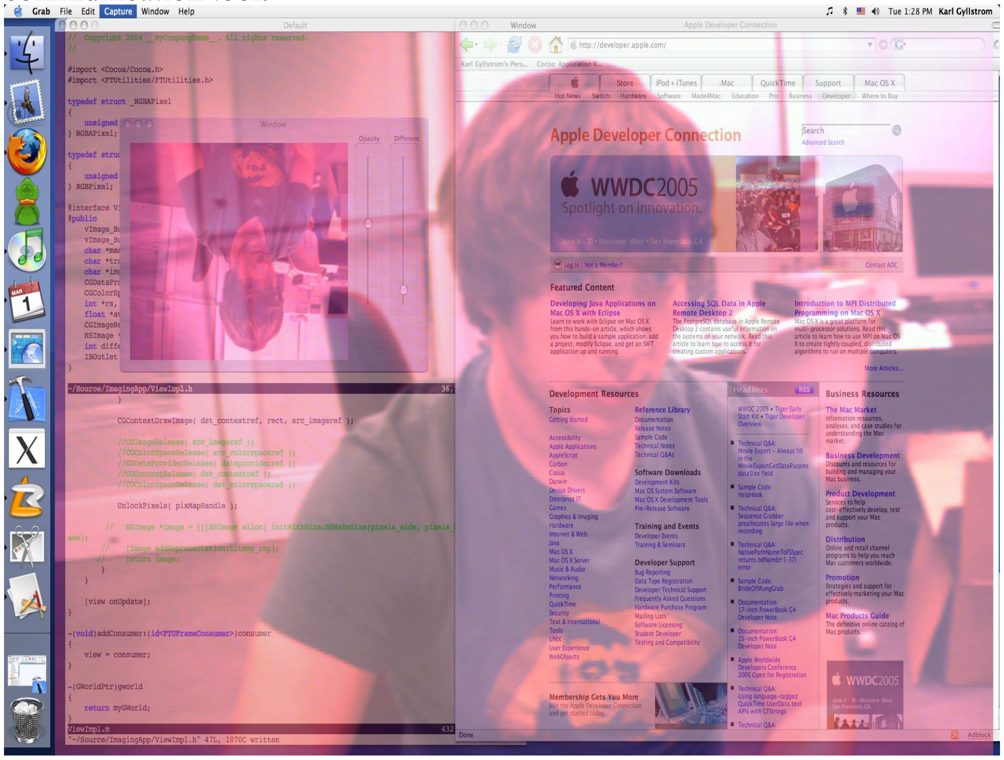


```
- (void)setOpenGLContext:(NSOpenGLContext*)context
{
    [self clearGLContext];
    _openGLContext = [context retain];
}

- (NSOpenGLContext*)openGLContext
{   // create a context the first time through
     if (_openGLContext == NULL) {
        _openGLContext = [[NSOpenGLContext alloc] initWithFormat:_pixelFormat !=
nil ? _pixelFormat : [[self class] defaultPixelFormat] shareContext:nil];
        [_openGLContext makeCurrentContext];
        [self prepareOpenGL]; // call to initialize OpenGL state here
    }
    return _openGLContext;
}

- (void)clearGLContext
{
    if (_openGLContext != nil) {
        if ([_openGLContext view] == self) {
            [_openGLContext clearDrawable];
        }
        [_openGLContext release];
        _openGLContext = nil;
    }
}

- (void)prepareOpenGL
{
    // for overriding to initialize OpenGL state, occurs after context creation
}

- (BOOL)isOpaque
{
    return YES;
}

- (void)setPixelFormat:(NSOpenGLPixelFormat*)pixelFormat
{
```

Figure 4: Visual clutter in 2-head Facetop… wallpaper, background, overlap

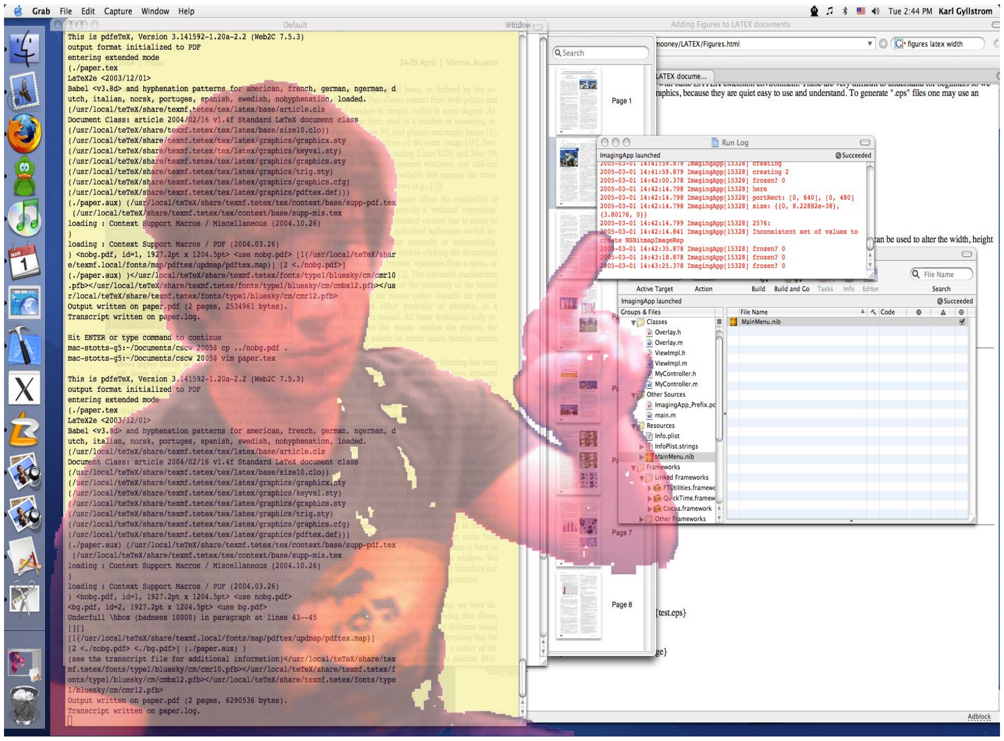
Other methods we have implemented involve different image manipulation techniques, reducing visual confusion directly in the video frames. One is removing pixels from the video that correspond to objects in the background. The process involves a few steps. First, a user must create a training sample of the background by activating the training sample acquisition feature within Facetop while not in the line of sight of the camera. This creates a bitmap of the background that future frames with the user in them are compared to. On each frame, each pixel of the new frame is compared to the pixel of the background image, forming a binary image mask with 1's that correspond to pixels that are significantly different from the background image. This binary image is then eroded and dilated to reduce noise. We apply this mask to the source video frame such that pixels corresponding to 0's in the mask are set to full transparency, or invisible, and pixels corresponding to 1's in the mask are set to the current transparency level (determined by either of the previous methods). This method converts a screen like Figure 5 into a screen as shown in Figure 6.

One other image form we are working with is embossing. Instead of showing the user in realistic video, for example, the same visual cues can be given by showing a gray-scale, embossed image. We currently do this via a delta computation, detecting which pixels have changed more that a certain percentage from one frame to the next. The resulting image is basically an outline of the user head and arm/hand, without backround items (which tend not to move). We switch back to realistic video when the Facetop is made opaque for use as a communication tool.



```
//  Copyright 2004 __MyCompanyName__. All rights reserved.
//

#import <Cocoa/Cocoa.h>
#import <FTUtilities/FTUtilities.h>

typedef struct _RGBAPixel
{
    unsigned
} RGBAPixel;

typedef struc
{
    unsigned
} RGBPixel;

@interface Vi
@public
    vImage_Bu
    vImage_Bu
    char *mas
    char *tra
    char *img
    CGDataPro
    CGColorSp
    int *rs,
    float *av
    CGImageRe
    NSImage *
    int diffe
    IBOutlet
}
~/Source/ImagingApp/ViewImpl.h                                              36,
            }

            CGContextDrawImage( dst_contextref, rect, src_imageref );

            //CGImageRelease( src_imageref );
            //CGColorSpaceRelease( src_colorspaceref );
            //CGDataProviderRelease( dataproviderref );
            //CGContextRelease( dst_contextref );
            //CGColorSpaceRelease( dst_colorspaceref );

            UnlockPixels( pixMapHandle );

        //   NSImage *image = [[[NSImage alloc] initWithSize:NSMakeSize(pixels_wide, pixels_
ase];
        //    [image addRepresentation:bitmap_rep];
       //     return image;
        }
    }

    [view onUpdate];
}

-(void)addConsumer:(id<FTUFrameConsumer>)consumer
{
    view = consumer;
}

-(GWorldPtr)gworld
{
    return myGWorld;
}
ViewImpl.m                                                                  432
"~/Source/ImagingApp/ViewImpl.h" 47L, 1070C written
```

Figure 5. Visual clutter in single-user Facetop



```
This is pdfeTeX, Version 3.141592-1.20a-2.2 (Web2C 7.5.3)
output format initialized to PDF
entering extended mode
(./paper.tex
LaTeX2e <2003/12/01>
Babel <v3.8d> and hyphenation patterns for american, french, german, ngerman, d
utch, italian, norsk, portuges, spanish, swedish, nohyphenation, loaded.
(/usr/local/teTeX/share/texmf.tetex/tex/latex/base/article.cls
Document Class: article 2004/02/16 v1.4f Standard LaTeX document class
(/usr/local/teTeX/share/texmf.tetex/tex/latex/base/size10.clo))
(/usr/local/teTeX/share/texmf.tetex/tex/latex/graphics/graphicx.sty
(/usr/local/teTeX/share/texmf.tetex/tex/latex/graphics/keyval.sty)
(/usr/local/teTeX/share/texmf.tetex/tex/latex/graphics/graphics.sty
(/usr/local/teTeX/share/texmf.tetex/tex/latex/graphics/trig.sty)
(/usr/local/teTeX/share/texmf.tetex/tex/latex/graphics/graphics.cfg)
(/usr/local/teTeX/share/texmf.tetex/tex/latex/graphics/pdftex.def)))
(./paper.aux) (/usr/local/teTeX/share/texmf.tetex/tex/context/base/supp-pdf.tex
 (/usr/local/teTeX/share/texmf.tetex/tex/context/base/supp-mis.tex
loading : Context Support Macros / Miscellaneous (2004.10.26)
)
loading : Context Support Macros / PDF (2004.03.26)
) <nobg.pdf, id=1, 1927.2pt x 1204.5pt> <use nobg.pdf> [1{/usr/local/teTeX/shar
e/texmf.local/fonts/map/pdftex/updmap/pdftex.map}] [2 <./nobg.pdf>]
(./paper.aux) )</usr/local/teTeX/share/texmf.tetex/fonts/type1/bluesky/cm/cmr10
.pfb></usr/local/teTeX/share/texmf.tetex/fonts/type1/bluesky/cm/cmbx12.pfb></us
r/local/teTeX/share/texmf.tetex/fonts/type1/bluesky/cm/cmr12.pfb>
Output written on paper.pdf (2 pages, 2534961 bytes).
Transcript written on paper.log.

Hit ENTER or type command to continue
mac-stotts-g5:~/Documents/cscw 2005$ cp ../nobg.pdf .
mac-stotts-g5:~/Documents/cscw 2005$ vim paper.tex

This is pdfeTeX, Version 3.141592-1.20a-2.2 (Web2C 7.5.3)
output format initialized to PDF
entering extended mode
(./paper.tex
LaTeX2e <2003/12/01>
Babel <v3.8d> and hyphenation patterns for american, french, german, ngerman, d
utch, italian, norsk, portuges, spanish, swedish, nohyphenation, loaded.
(/usr/local/teTeX/share/texmf.tetex/tex/latex/base/article.cls
Document Class: article 2004/02/16 v1.4f Standard LaTeX document class
(/usr/local/teTeX/share/texmf.tetex/tex/latex/base/size10.clo))
(/usr/local/teTeX/share/texmf.tetex/tex/latex/graphics/graphicx.sty
(/usr/local/teTeX/share/texmf.tetex/tex/latex/graphics/keyval.sty)
(/usr/local/teTeX/share/texmf.tetex/tex/latex/graphics/graphics.sty
(/usr/local/teTeX/share/texmf.tetex/tex/latex/graphics/trig.sty)
(/usr/local/teTeX/share/texmf.tetex/tex/latex/graphics/graphics.cfg)
(/usr/local/teTeX/share/texmf.tetex/tex/latex/graphics/pdftex.def)))
(./paper.aux) (/usr/local/teTeX/share/texmf.tetex/tex/context/base/supp-pdf.tex
 (/usr/local/teTeX/share/texmf.tetex/tex/context/base/supp-mis.tex
loading : Context Support Macros / Miscellaneous (2004.10.26)
)
loading : Context Support Macros / PDF (2004.03.26)
) <nobg.pdf, id=1, 1927.2pt x 1204.5pt> <use nobg.pdf>
<bg.pdf, id=2, 1927.2pt x 1204.5pt> <use bg.pdf>
Underfull \hbox (badness 10000) in paragraph at lines 43--45
[][]
[1{/usr/local/teTeX/share/texmf.local/fonts/map/pdftex/updmap/pdftex.map}]
[2 <./nobg.pdf> <./bg.pdf>] (./paper.aux) )
(see the transcript file for additional information)</usr/local/teTeX/share/tex
mf.tetex/fonts/type1/bluesky/cm/cmr10.pfb></usr/local/teTeX/share/texmf.tetex/f
onts/type1/bluesky/cm/cmbx12.pfb></usr/local/teTeX/share/texmf.tetex/fonts/type
1/bluesky/cm/cmr12.pfb>
Output written on paper.pdf (2 pages, 6290536 bytes).
Transcript written on paper.log.
```

```
ImagingApp launched                                                   Succeeded
2005-03-01 14:41:59.879 ImagingApp[15328] creating
2005-03-01 14:41:59.879 ImagingApp[15328] creating 2
2005-03-01 14:42:00.378 ImagingApp[15328] frozen? 0
2005-03-01 14:42:14.798 ImagingApp[15328] here
2005-03-01 14:42:14.798 ImagingApp[15328] portRect: [0, 640], [0, 480]
2005-03-01 14:42:14.798 ImagingApp[15328] size: {{0, 8.22882e-38},
{3.80176, 0}}
2005-03-01 14:42:14.799 ImagingApp[15328] 2576:
2005-03-01 14:42:14.841 ImagingApp[15328] Inconsistent set of values to
create NSBitmapImageRep
2005-03-01 14:42:35.878 ImagingApp[15328] frozen? 0
2005-03-01 14:43:18.878 ImagingApp[15328] frozen? 0
2005-03-01 14:43:25.378 ImagingApp[15328] frozen? 0
```

Figure 6. Controlling visual clutter with automatic background elimination

Directions for continuing research

With NSF support for equipment, we are building a high-speed network of PCs that will function as a distributed "graphics card" for a virtual projected collaboration space. Users in this collaboration space will have a graphics PC attached to his/her normal working PC. Desktop information from the user's PC will be captured by the slave graphics PC, and then will be communicated in access-controlled ways to the other graphics PCs in the collaboration space.

These spaces will be projected on office walls by 2 or 3 projectors side-by-side, giving a double or triple-wide bit field for desktop artifacts. This extra large projected desktop will be populated with artifacts from the user's own workspace, as well as artifacts from collaborating users (with properly applied read-only, or read-write privileges). We are developing the concepts for an N-head Facetop to be used in this virtual collaborations space. The problems to be solved are more difficult than for the 2-head Facetop.

Finally, we are starting a project with emergency room physicians at Washington Hospital Center (WHC) to incorporate a Facetop capability into their emergency medical records system. We will need to develop video methods to help users distinguish between user video images, and data signals that resemble video (such as x-rays and patient photographs). This project will also incorporate into Facetop better hand gestures that have been developed at WHC for allowing surgeons in operating rooms to control the medical records without touching screens or keyboards.

In addition to these projects that are started, we have Facetop being evaluated by major industrial and financial companies for collaborative engineering design, examination of financial records, distance learning classroom uses, discussion of architectural blueprints, and web page layout. In addition to these commercial users, we have responded to evaluation requests from English professors, physics researchers, artists and art teachers, and business professors. The desire of these companies, educators, and artists to try our work supports our conclusion that the Facetop is a novel and useful new paradigm for support of close synchronous paired collaborations.

Acknowledgements. This work was partially supported by a grant from the U.S. Environmental Protection Agency, # R82-795901-3.

References

[1] Beck, K., *Extreme Programming Explained*, Addison-Wesley, 2000.

[2] Wells, J. D., "Extreme Programming: A Gentle Introduction," 2001, available on-line at http://www.extremeprogramming.org/

[3] A. Cockburn and L. Williams, "The Costs and Benefits of Pair Programming," *eXtreme Programming and Flexible Processes in Software Engineering -- XP2000*, Cagliari, Sardinia, Italy, 2000.

[4] H. Ishii, M. Kobayashi, and J. Grudin, "Integration of inter-personal space and shared workspace: ClearBoard design and experiments," *Proc. of ACM Conf. on Computer Supported Cooperative Work*, Toronto, 1992, pp. 33-42.

[5] R. S. Fish, R. E. Kraut, and B. L. Chalfonte, "The VideoWindow System in Informal Communications," *Proc. of ACM Conf. on Computer Supported Cooperative Work*, Los Angeles, 1990, pp. 1-11.

[6] P.Baheti, L.Williams, E.Gehringer, and D.Stotts, "Exploring the Efficacy of Distributed Pair Programming," XP Universe 2002, Chicago, August 4-7, 2002; Lecture Notes in Computer Science 2418 (Springer), pp. 208-220.
[7] Birchfield, S., Elliptical Head Tracking Using Intensity Gradients and Color Histograms, IEEE Conf on Computer Vision and Pattern Recognition, Santa Barbara, CA, June 1998.
[8] Eric A. Bier, Ken Fishkin, Ken Pier, Maureen C. Stone, "A Taxonomy of See-Through Tools: The Video, http://www.acm.org/sigchi/chi95/Electronic/documnts/videos/eab1bdy.htm, Proc. of CHI '95, Xerox PARC.
[9] T. Nishi, Y. Sato, H. Koike, "SnapLink: Interactive Object Registration and Recognition for Augmented Desk Interface," Proc. of IFIP Conf. on HCI (Interact 2001), pp. 240-246, July 2001.
[10] Steve Benford, John Bowers, Lennart E. Fahlén, Chris Greenhalgh and Dave Snowdon, "User Embodiment in Collaborative Virtual Environments,", Proc. of CHI '95, http://www.acm.org/sigchi/chi95/Electronic/documnts/papers/sdb_bdy.htm
[11] Beverly L. Harrison , Hiroshi Ishii, Kim J. Vicente, and William A. S. Buxton,"Transparent Layered User Interfaces: An Evaluation of a Display Design to Enhance Focused and Divided Attention," http://www.acm.org/sigchi/chi95/Electronic/documnts/papers/blh_bdy.htm, Proc. of CHI '95.
[12] Vision Interface Seminar, Fall 2001, MIT, http://www.ai.mit.edu/~trevor/6.892/
[13] Bretzner, L., and T. Lindberg, "Use Your Hand as a 3-D Mouse, or, Relative Orientation from Extended Sequences of Sparse Point and Line Correspondences Using the Affine Trifocal Tensor," Proc. of the 5th European Conf. on Computer Vision, *(H. Burkhardt and B. Neumann, eds.), vol. 1406 of Lecture Notes in Computer Science*, (Freiburg, Germany), pp. 141--157, Springer Verlag, Berlin, June 1998.
[14] Laptev, I., and T. Lindberg, "Tracking of multi-state hand models using particle filtering and a hierarchy of multi-scale image features," Proc. of the IEEE Wksp on Scale-space and Morphology, Vancouver, Canada, in *Springer-Verlag LNCS 2106* (M. kerckhove, ed.), July 2001, pp. 63-74.
[15] Stotts, D., L. Wiliams, et al., "Virtual Teaming: Experiments and Experiences with Distributed Pair Programming," TR03-003, Dept. of Computer Science, Univ. of North Carolina at Chapel Hill, March 1, 2003.
[16] J. Smith, D. Stotts, and S.-U. Kum, "An Orthogonal Taxonomy for Hyperlink Anchor Generation in Video Streams using OvalTine," *Proc. of Hypertext 2000 (ACM)*, May, 2000, San Antonio, Texas, pp. 11-18.
[17] Nosek, J.T., "The Case for Collaborative Programming," *Communications of the ACM*, March 1998, pp. 105-108.
[18] Olson, G.M., and J.S. Olson, "Distance Matters," *Human-Computer Interaction*, vol. 15, 2000, pp. 139-179.
[19] Williams, L., "The Collaborative Software Process," *Ph.D. dissertation*, Dept. of Computer Science, Univ. of Utah, Salt Lake City, UT, 2000.
[20] JAWS, Windows screen reader, Freedom Scientific, http://www.freedomscientific.com/
[21] Hanks, B.,"Virtual Pair Programming," Doctoral Symposium at the International Conference on Software Engineering (ICSE 2003), May 3 - 10, 2003, Portland, OR.
[22] Roussel, N., The videoSpace Toolkit, http://www.lri.fr/~roussel/projects/videoSpace/, Feb. 2005.
[23] Hiroshi Ishii and Minoru Kobayashi. Clearboard: a seamless medium for shared drawing and conversation with eye contact. In Proceedings of the SIGCHI conference on Human factors in computing systems, pages 525–532. ACM Press, 1992.
[24] Patrick Baudisch and Carl Gutwin. Multiblending: displaying overlapping windows simultaneously without the drawbacks of alpha blending. In Proceedings of the 2004 conference on Human factors in computing systems, pages 367–374. ACM Press, 2004.
[25] Carl Gutwin, Jeff Dyck, and Chris Fedak. The effects of dynamic transparency on targeting performance. In Graphics Interface, pages 105–112. CIPS, Canadian Human-Computer Communication Society, A K Peters, June 2003. ISBN 1-56881-207-8, ISSN 0713-5424.
[26] Beverly L. Harrison, Gordon Kurtenbach, and Kim J. Vicente. An experimental evaluation of transparent user interface tools and information content. In UIST '95: Proceedings of the 8th annual ACM symposium on User interface and software technology, pages 81–90. ACM Press, 1995.
[27] Jie Yang, R. Stiefelhagen, U. Meier, A. Waibel, "Visual Tracking for Multimodal Human Computer Interaction," Proc. of CHI '98, Apr. 1998, pp. 18-23.