Design Of A New PumaPaint Interface And Its Use in One Year of Operation

Michael Coristine
Computer Science Student
Roger Williams University
Bristol, RI 02809 USA
michael_coristine@raytheon.com

Mathew R. Stein
Associate Professor of Engineering
Roger Williams University
Bristol, RI 02809 USA
mstein@rwu.edu

***Abstract*—The PumaPaint Project is an online robot that allows World Wide Web users to remotely create original artwork. The original site located at Wilkes University operated from June of 1998 to March of 2000 with approximately 25,000 unique-addressed machines downloading the interface to produce about 500 canvases. The new site at Roger Williams University opened to the public in August of 2002. After noticing that over fifty-percent of the machines downloading the interface were located outside of the United States we decided to create a more elegant and intuitive interface aimed at users from varied countries and varied ages. This paper addresses issues, concerns, and resolutions developed to provide this updated interface. The current site was updated integrating this new interface in November of 2002 and evaluation of the interface is currently underway.**

Keywords-Online robots, telerobotics, user interface

I. Introduction And Background

The PumaPaint Project [1] is an online robot that allows World Wide Web users to remotely create original artwork. The original site located at Wilkes University [2] operated from June of 1998 to March of 2000 with approximately 25,000 unique-addressed machines downloading the interface to produce about 500 canvases. The original Java interface was written by Peter DePasquale [5] and Matthew Stein [3,4] in 1998 and functioned unmodified for the life of the Wilkes site.

The new site at Roger Williams University opened to the public in August of 2002 [6]. The original Java interface did not take advantage of recent Java2 features and relied on English text for much of its labeling. As an undergraduate research project, Michael Coristine redesigned the PumaPaint interface in the Fall of 2002. This new interface was then continuously used for the first year of operation of the PumaPaint site. In sections two and three we will discuss the design issues associated with creation of the new interface. In section four we will present result of the first year of operation of the PumaPaint site. Section five will be summary and conclusions.

II. The New Interface

A PUMA 560 robot is equipped with four paintbrushes, four jars of paint (red, green, blue and yellow), and white paper attached to a vertical easel. The interface is specifically designed to allow WWW users to use the robot to create original paintings. It provides two windows showing live video views of the work site, various controls for connecting to the robot, getting help, task status, and controlling the painting task. The interface is a Java2™ applet with Swing components and requires compatible browser with the proper plug-in.

Because the applet is executed on the web user's machine, the user interacts directly with the applet and receives immediate feedback. The interface takes advantage of this feature, providing two channels of feedback: one immediate and virtual and the other time-delayed and real.

III. Design Issues

The overall layout of the interface is consistent with many existing paint applications (Adobe PhotoShop/Illustrator, Corel Draw, Paint Shop Pro, Macromedia Freehand). We did this in an effort to give some users a level of comfort and familiarity.

The help button is located in the upper right corner of the interface's window, the left panel contains the user tools for controlling the robot, while the center panel consists of the virtual canvas. We have made an attempt to rely less on written language by incorporating icons and visual cues throughout the interface design. For instance, when a user selects a paintbrush the cursor changes to a paintbrush when it is passed over the virtual canvas (Figure 3). The paintbrush cursor does not have a color until the user selects the dip brush button.

The PumaPaint Interface is achieved using a Border Layout. A Border Layout consists of five areas: North, South, East, West, and Center, corresponding to top, bottom, right, left and center.

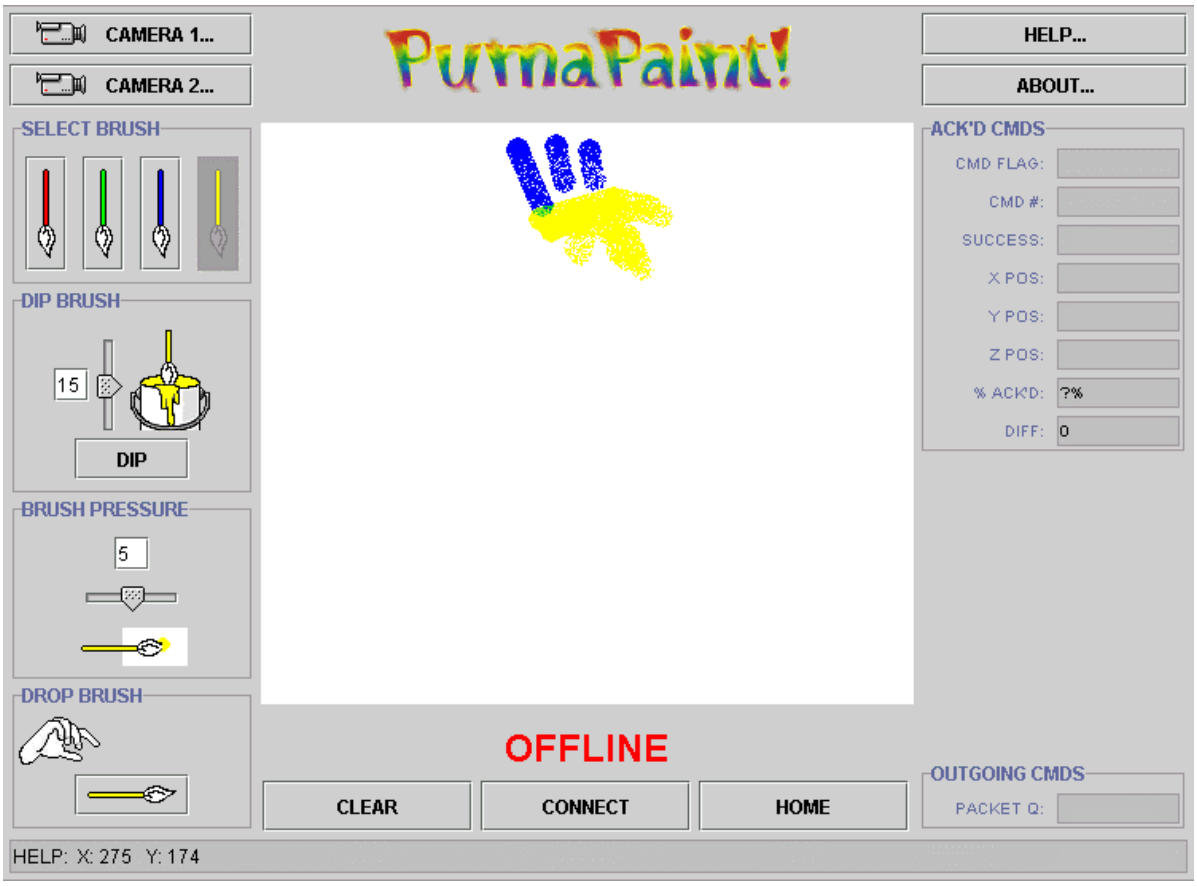

Figure 1. The PumaPaint Interface

The North Panel contains the camera buttons, the PumaPaint logo, and the "Help…" and "About…" buttons. The left side of the North Panel displays the two camera buttons (Figure 1). Selecting either of these buttons will spawn a new window containing an automatically updated image from the live video cameras. Video update rates depend on the quality of the image selected via a slider in the camera window (Figure 2). The maximum update rate approaches live video rates but typical update rates will be slower depending on image size and transfer time.

Selecting the about button spawns a window describing the origins of the interface, while the help button displays the help window (Figure 4). The help window is divided into two sections. The top section is a dynamically updated list of actions the user may perform. This list of actions informs the user of what painting actions currently make sense. For example, if the painter has used all of the paint on the brush, the list will remove 'Paint on the canvas' as a valid action, and inform the user that it is currently valid to select a new color or dip the brush back into the paint. The lower portion of the help window displays help topics that may be selected by using the left mouse button.

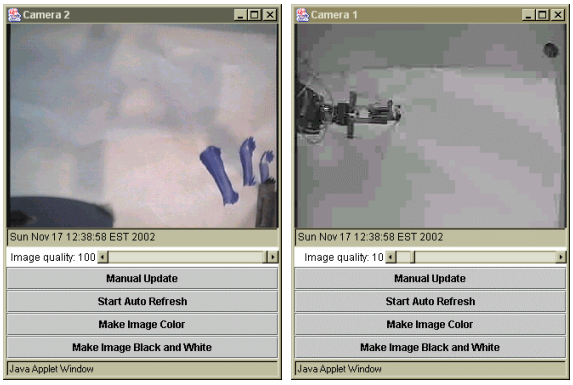

Figure 2. "Camera1" and "Camera2" Windows

Figure 3. Cursors: Default Clear, Red, Green, Blue and Yellow

The South Panel contains the user's Help/Status Text Field (Figure 5). The Help/Status area displays information to the user regarding the current coordinates of the mouse over the canvas and the status of the painting effort. For example, a message

warning the user that there is no more paint on the brush will appear in this area if all paint has been exhausted.

The East Panel shows Command Number, Success Flag, and current position (see Figure 1). Despite the typical user's lack of interest in this area, it shows the difference between packets sent and received and the packet queue. These fields are essential to understanding the issues of time-delayed teleoperation, and are included in the hopes that a user will show interest. The "Diff" field shows the difference, in command number, between the last command sent and the last acknowledgement received. This field will show that the robot is almost always behind the user, and sometimes quite far behind. If the user wishes to "wait for the robot", he or she is waiting for the robot to process every command sent, indicated by a zero in the field.

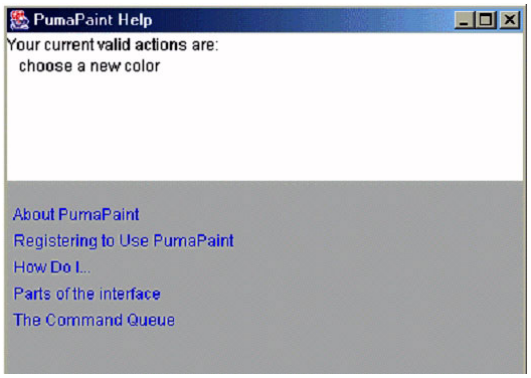

Figure 4. "Help…" Window

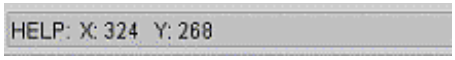

Figure 5. Help/Status Text Field

The "packet queue" field shows the length of the queue of commands waiting to be sent to the robot. Irrespective of communication delay, it takes much less time for the user to specify commands (via mouse actions) than it does for the robot to perform these commands. For example, it takes virtually no time for the user to select the "Dip" button, but about twenty seconds for the robot to perform this action. Thus, in a matter of minutes, it is easily possible for the user to specify hours of robot motion. This is not likely to be useful, so the length of the outgoing packet queue is limited to two hundred commands, equivalent to about ten minutes of robot motions. If the queue reaches this length, the interface notifies the user that he or she is "too far ahead" of the robot and has the option of either waiting or flushing the outgoing command queue.

The West and Center Panels work interactively along with the robot (Figure 1). The West Panel allows the user to control the robot's movements: to select brush color, choose the depth of the dipping motion into the paint jar, and the pressure with which the paintbrush is pressed against the canvas.

The Center Panel contains the virtual canvas; the main area of interaction. The virtual canvas is an attempt to represent what occurs on the actual canvas. By pressing down on the left mouse button, holding and dragging the mouse in this area the user issues commands to the remote robot to apply paint to the real canvas. These mouse actions also cause the selected color to appear on the virtual canvas. We suspected that simply turning virtual canvas pixels beneath the mouse to the selected color would mislead rather than assist the user; so several features are added in an attempt to increase the fidelity of the virtual canvas. The virtual canvas is colored as a blob, rather than a sharp line. The blobs contain randomly generated gaps and streaks, and the proportion of area turned to the selected color progressively decreases as the brush stroke continues. This simulates the effect of depleted paint in an attempt to remind the user to manually replenish the paintbrush. Another aid simulates colors mixing on the canvas. Should a blue brush stroke be made to overlap a yellow brush stroke on the virtual canvas, the resulting overlap will appear green.

The bottom of the Center Panel also contains three buttons and a panel that indicates weather the robot is "ONLINE" or "OFFLINE". The center button toggles between "Disconnect" and "Reconnect". Disconnect closes the communication socket, stops the spawned reader and writer threads, and flushes the command queue. Disconnect does nothing if there is not a connection currently established. Reconnect reopens the password window that invites users to enter a password and connect to the robot. The default password is provided, and this will work at all times unless another user has requested private access to the site.

The left button clears the virtual canvas (sets to white). Because the robot is handling real, physical objects, the position and movements of the brushes can never be modeled exactly. From the camera images, the user can tell if the real canvas even remotely resembles the virtual canvas. If not, the user has the option of clearing the virtual canvas and starting again. Lastly, the right button returns the robot to its home position, centered in front of the canvas.

Let's consider a typical scenario of a user just beginning to use the interface. The Help/Status field (Figure 5) initially informs the user to "To begin…Select a Color Button, then the Dip Brush Button." When the user selects a paintbrush color, several visuals cues are triggered (Figures 6a, 6b and 6c).

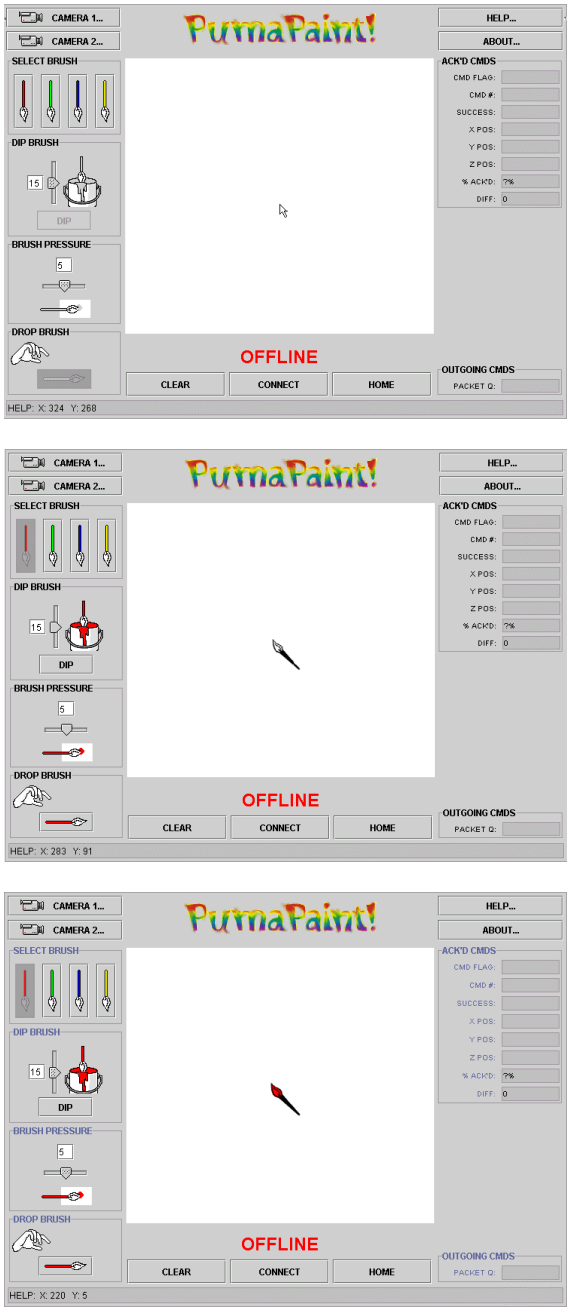

Figure 6. (A, B &C) PumaPaint Visual Queues

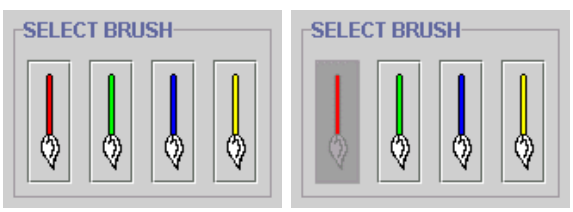

Figure 7. Select Brush Panel

First, the selected paintbrush is indicated within the Select Brush Panel (Figure 7).

Within the Dip Brush Panel the "DIP" button is activated and the dip image displays the paint color selected (Figure 8).

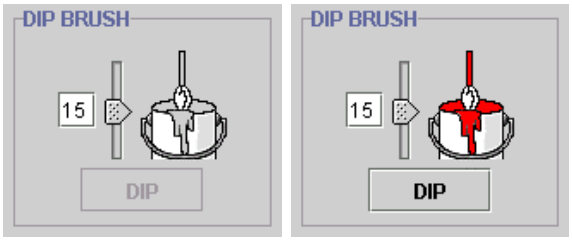

Figure 8. Dip Brush Panel

Within the Brush Pressure Panel the brush pressure image displays the color selected (Figure 9).

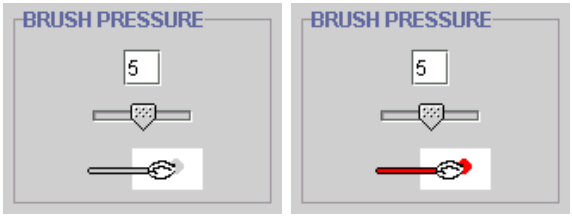

Figure 9. Brush Pressure Panel

Within the Drop Brush Panel the drop brush button is activated and the paintbrush image within the button displays the paintbrush color selected (Figure 10).

Figure 10. Drop Brush Panel

Lastly, the cursor is transformed to a paintbrush cursor when passed over the virtual canvas (Figure 6b). Until the dip brush button is selected, the paintbrush cursor will not display the paintbrush color selected (Figure 6c).

The paintbrush button images were achieved by creating an image using PhotoShop with a canvas size identical to the predetermined size of the buttons. The bristles of the brush were left transparent while the background was set to the same gray as used in the java interface. The image was then saved for the web as a transparent "GIF 128 Dithered" file. Each button within the panel is set with the same image and the desired color of each button is achieved by setting its background to the desired color while sizing the button to the exact size of the image.

The user has two parameter options for determining how the paint will be applied to the canvas by the robot. The first option is contained

within the Dip Brush Panel (Figure 11) and allows the user, through a slider, to control the depth that the robot dips the appropriate paintbrush into its paint jar, thus controlling the quantity of paint on the brush. The second option is contained in the Brush Pressure Panel and allows the user through a slider to control the pressure that the robot applies the brush to the canvas.

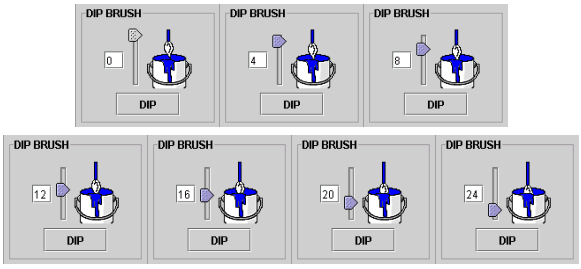

Figure 11. Sample renderings of Dip Brush Panel

The dip brush slider image is achieved in similar fashion to the previously discussed paintbrush buttons. In this case the image is rendered using layers. The layers consist of the gray background, the paint can image (split into two layers), and the paintbrush. The first layer contains the gray background. The second layer contains the top of the paint can (which is transparent). The third layer contains the paintbrush (with transparent handle). The last and foremost layer contains the front of the paint can (with the paint drip being transparent). The animation is achieved by duplicating the paintbrush layer while offsetting the image by one pixel each time and erasing the part of the image that overlaps the transparent paint drip.

We determined that the animation worked smoothly enough with increments of two, saving the amount of images needed to be rendered by half. Also, by using transparent "GIFs", we were able to utilize the background color of the Image Icon to display the selected color. This allowed us to use one set of images rather than draw a set for each color. This method also worked to display gray when no color had been selected.

The animation was perhaps too effective at immolating the robot's movement. Initially, the "DIP" button itself contained the animation, but we felt this might be confusing to the user. The button animation made it less apparent that the button was, in fact, a button. It appeared to us (and we conjectured, to would be users) that moving the slider initiated the robot's movement. We felt it might be confusing to potential users that it was still necessary to select the animated button and that the slider merely adjusted the dip brush depth. We decided that the "DIP" button should be separated from the animation.

The Brush Pressure Panel (Figure 12) was developed using identical methods. Because this slider only consisted of eleven values (0-10) there is a separate image rendered for each of these values. The offset of the animation's movement between the "zero" value and the "one" value is ten instead of one. This was done to make it clearer to the user that a setting of zero hovers in front of the canvas, while settings of "one" though "ten" adjusts the degree of pressure the brush applies to the canvas.

We created this animation in an attempt to improve the understanding of the function of the slider, but we again encountered unintended consequences of this change. In the many cycles of the robot grasping and releasing the brushes, there is always variation in the physical grasp of the brush by the robot. It is the user's central responsibility to accommodate for this variation by carefully controlling the physical interaction of the brush and the canvas using the visual feedback provided. As shown in Figures 12, the animation indicates that a particular pressure setting (e.g. '6') will produce a certain level of pressure of the brush on the canvas. It may be the case that a pressure setting of '6' would not cause the brush to touch the canvas as the animated icon shows. We are currently evaluating the benefits of the animation versus the potential to misinform the user.

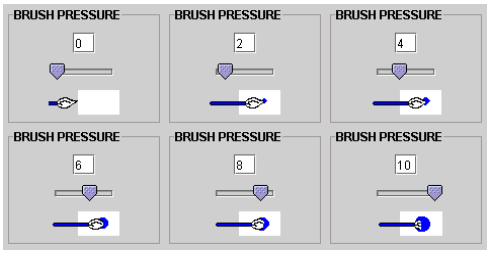

Figure 12. Sample renderings of Brush Pressure Panel

The last panel, within the West Panel, is the Drop Brush Panel. The drop brush button directs the robot to drop the brush and resets all the previously mentioned panels within the West Panel to their initial states and resets the cursor back to its default arrow as well (Figure 6a). Dropping the brush is only useful if the user wishes to view the canvas using the arm-mounted camera and does not want a brush in the way.

IV. First Year Results

We examined the **httpd** log from August 2002 to August 2003 to determine the usage statistics shown in Table 1. These figures suggest 9391 visitors to the main PumaPaint site with 4848 distinct hosts accessing the site. The latter figure can serve as an estimate of the number of users viewing the site in one year's time. The interface was downloaded 2720,

(2472 distinct) times, indicating users potentially gaining access to the robot, while 1716 unique hosts received a "Class not found" message indicating their browser was not Java2/Swing compatible. The intersection of these sets was 239 unique addresses, and this probably indicates users that first encountered the "plug-in required" message, obtained the plug-in and later downloaded the interface.

	Total	Distinct
Main Page	9391	4848
PumaPaint class	2720	2472
"Class not found"		1716
Intersection		239

TABLE I. PUMAPAINT USAGE AUG. 02 – AUG '03

We can draw some interesting interpretations from Table 1. We see that about 40% of users attempting to use the interface in 2002-2003 did not have Java2/Swing compatible browsers, and that of these only 14% apparently bothered to obtain the plug-in and return to download the interface. We also note a gap of about 800 users who viewed the main page but did not attempt to download the interface.

Figure 13. Collage Of Artwork Using the New Interface

We also briefly note that amongst the most interesting aspects of the PumaPaint project is seeing what people choose to paint. Figure 13 shows a collage of some of the most interesting canvases produced in the first year of operation, of the 110 canvases produced.

V. SUMMARY AND CONCLUSION

We realized, as is often the case, new improvements can lead to new problems. We upgraded the interface to Java2/Swing because it seemed like a good idea at the time, not realizing that many installed browsers do not support Java2/Swing. Instructions had to be added to the website plus a link to the plug-in, however it seems a minority of potential users chose to follow this link and obtain the necessary plug-in.

Another example, the new animations proved to work perhaps too well. In the case of the dip brush animation, we felt it might confuse the user. The animation visually seemed to imply the movement of the robot itself, rather than the level of the dip movement to be sent to the robot.

Shirking labor proved to be a motivation for innovation. Our desire not to have to draw hundreds of repetitive images brought us to seek the more creative method of using transparent gifs. Merging these images with the ability to set the background color of the various components allowed us to save both time and frustration. This solution also yielded savings in other areas: lines of code, storage space and efficiency.

We will continue to monitor the PumaPaint site to discover if the new interface achieved our desired results. The quality of the artwork produced using the new interface appears comparable to the original PumaPaint site. We currently have no strong indications that the interface is easier to use, but we see the Java2/Swing support as a significant barrier to potential users.

ACKNOWLEDGMENT

We would like to acknowledge the Roger Williams University Research Foundation for making this project possible. We would also like to thank A. Ryan Tiebout, Christopher P. Madden, Peter DePasquale, Lara Blatchford and Dr. John Lewis of Villanova University for their contribution to this project.

REFERENCES

[1] http://pumapaint.rwu.edu

[2] http://yugo.mme.wilkes .edu/~villanov

[3] M. R. Stein, C Ratchford, K. Sutherland, D. Robaczewski, "Internet Robotics: An Educational Experiment", *Telemanipulator and Telepresence Technologies II,* SPIE Volume 2590, 1995

[4] M. R. Stein, K. Sutherland, "Sharing Resources over the Internet for Robotics Education", *Telemanipulator and Telepresence Technologies IV,* SPIE Volume 3206, 1997

[5] P. DePasquale, J. Lewis, M. R. Stein, "A Java Interface for Asserting interactive Telerobotic Control", *Telemanipulator and Telepresence Technologies IV,* SPIE Volume 3206, 1997

[6] Matthew R. Stein, "The PumaPaint Project", *Journal of Autonomous Robots*, Fall 2003