

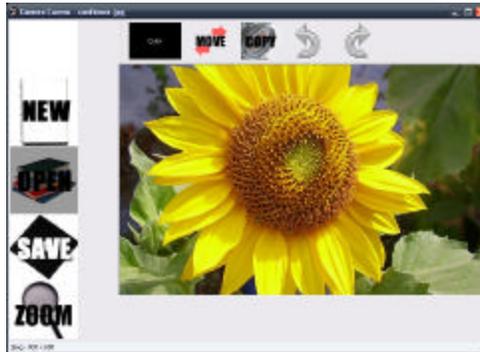
Camera Canvas: Image Editor for People with Severe Disabilities

Won-Beom Kim, Christopher Kwan, Igor Fedyuk, and Margrit Betke

Technical Report BUCS-TR-2008-010

Directed Study Project, CAS CS 492, Spring 2008

Department of Computer Science, Boston University, Boston, MA 02215, USA



Abstract:

Camera Canvas is an image editing software package for users with severe disabilities. It is specially designed for Camera Mouse, a camera-based mouse-substitute input system for people with limited mobility. Users can manipulate images through various head movements, tracked by Camera Mouse. The system is also fully usable with traditional mouse or touch-pad input. Designing the system, we studied the requirements and solutions for image editing and content creation when using Camera Mouse. Experiments with 20 subjects, each testing Camera Canvas with Camera Mouse as the input mechanism, showed that users found the software easy to understand and operate. User feedback was taken into account to make the software more usable and the interface more intuitive. We suggest that the Camera Canvas software makes important progress in providing a new medium of utility and creativity in computing for users with severe disabilities.

Keywords:

Human computer interaction, image editing, user interfaces, accessibility, camera interfaces, universal access, assistive software

Introduction:

Motivation:

Image editing is a powerful tool in expression. It has pervaded multiple industries and has also become a cornerstone of personal computing. However, the benefits of image editing have been unavailable to users with severe disabilities because of its reliance on handheld input systems. We hope to take the initial steps in bringing the power of image editing software to users with disabilities by designing an image editor that works with mouse pointer input provided by Camera Mouse system.

Camera Mouse:

The Camera Mouse computer interface is a camera-based mouse-replacement input system developed by Professors Margrit Betke and James Gips and their students, initially at Boston College, and then also at Boston University (Betke et al., 2002). It uses a camera to track a feature on a user's face and thus allows the user to move the mouse pointer on the screen by simply moving his head. Clicking events are registered when a user dwells the mouse pointer on an area for a certain amount of time. The interface is designed for users who cannot use their hands to move a mouse but can move their heads.

Project Goals:

- To develop an intuitive image editor interface specially geared towards users with severe disabilities.
- To study the requirements and solutions for image editing and content creation using the Camera Mouse input system.
- To develop a fully functioning image editing software package that also works with traditional computer interfaces.

Methods:

Overall Design:

The structure and design of Camera Canvas is outlined by the Unified Modeling Language in Fig. 1.

The Camera Canvas design is composed of two important classes: the Image class for handling loading, saving, manipulating, and displaying images and the UserInterface class for providing an accessible user interface.

The Image class provides the basic functionalities expected in common image editing software, serving as the backend of the application. It contains features such as zooming in and out, copying and pasting, drawing, and a history feature, which allows the user to revert back to older editions of the bitmap image.

The UserInterface class serves as the front-end of Camera Canvas and allows interaction with Camera Mouse to control the software features. The Toolbar class is the main component of

the `UserInterface` class, where users can select and use different tools by moving the mouse pointer.

The current version of Camera Canvas implements only a subset of the planned class design. The `UserInterface` class implements the components specified fully, but several planned features of the `Image` class remain unimplemented; namely, the ability to apply effects (such as blur or sharpen filters) and drawing text onto the image.

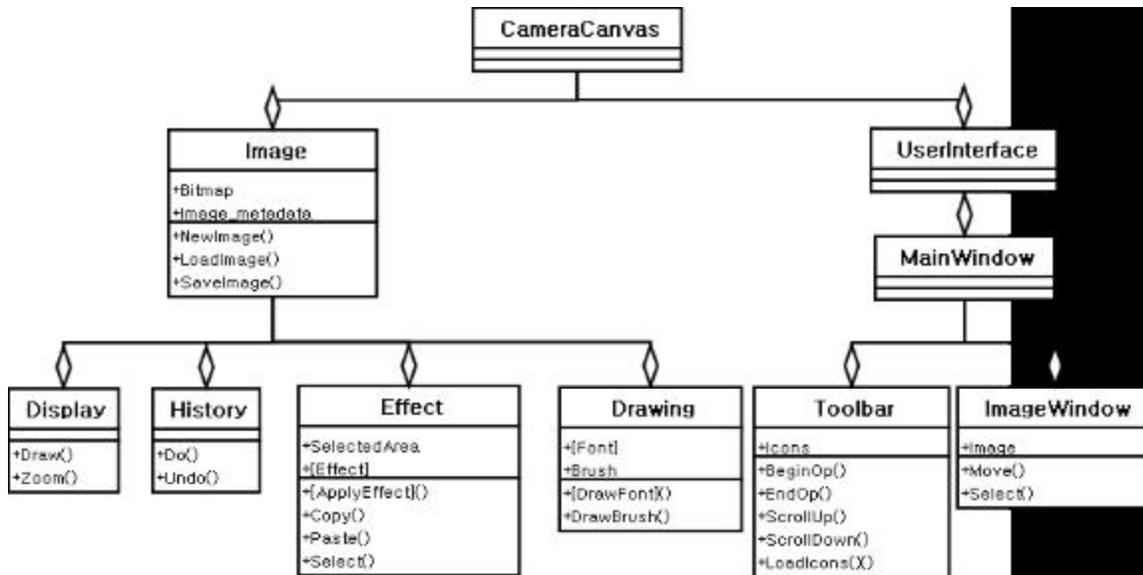


Fig. 1. Overall design and class structure of Camera Canvas. Unimplemented methods and fields are surrounded by "[]" signs.

Interface Design Goals:

- To provide a smooth and intuitive experience.
- To work well with the Camera Mouse input system.
- To minimize user head movement.
- To reduce accidental activation of features.

Interface Elements:

- Screen Layout:
 - The two main toolbars are pushed into the left and top of the screen to give the user as much space as possible to view and work with the image.
- Icons:
 - Easy to understand images were chosen for the feature icons. However, the name of each feature is also displayed in large bold letters in case the meaning

of the image is unclear. The large size and spacing of the icons makes it easier for the user to choose them using Camera Mouse.

- Feature Activation Indicators:
 - Upon selecting and using features, various confirmation messages, such as, “Selection Copied,” appear in large letters on the screen along with audible sounds. These help the user know that they have activated certain features and try to guide them on what to do next.
- Sliding Toolbar:
 - The goal of the sliding toolbar is to reduce long tiring head movements. The toolbar contains all the different features of the program. The center of the toolbar has a shaded gray square. Whatever feature lies under the shaded gray square is the feature that is selected. The user navigates the menu by moving the mouse pointer (via head movement) above or below the gray box. Making a movement above the gray box will cause the menu bar to slide down so that the features move down towards the gray box. Making a movement below the gray box will cause the toolbar to slide up so that icons below move up towards the gray box.
- Top Toolbar
 - The aim of the top toolbar is to provide an easy to access location for a few frequently used features: COLOR, MOVE, COPY, UNDO, REDO.
- Color Selection
 - Traditional color pickers require precise mouse control to pick a certain color from a color wheel. Since that level of precision is difficult with Camera Mouse, we chose to offer 8 preset colors with 8 different shades of each. The buttons are large and spaced apart to make them easier to select with Camera Mouse.
- CameraCanvas setup – Program Configuration Utility
 - This configuration manager allows users to change the folder where images are saved, the default image size, whether or not the user wants the MOVE feature to be inverted, and the speed of the MOVE feature. However, this setup has not yet been optimized for use with Camera Mouse.

Program Features:

Feature	Description
	<p>Allows the user to create a new image to edit. Contains 4 preset sizes for blank white canvases and also the option to create a new image from data on the system clipboard.</p> <p>This feature is located on the sliding toolbar.</p>

<p>OPEN</p> 	<p>Allows the user to open any image within the program's image directory.</p> <p>This feature is located on the sliding toolbar.</p>
<p>SAVE</p> 	<p>Allows the user to save the current image or to save a new copy of the current image.</p> <p>This feature is located on the sliding toolbar.</p>
<p>ZOOM</p> 	<p>Allows the user to view the image through 6 levels of zooming, ranging from 10% to 400% (see Appendix Fig. 8).</p> <p>Implemented using a built in .NET function for drawing a part of the image to an area defined by a rectangle.</p> <p>This feature is located on the sliding toolbar.</p>
<p>MOVE</p> 	<p>Allows the user to change the current view of the image.</p> <p>When this feature is used, four translucent red arrows appear on top of the middle of the image. By dwelling on any one of the arrows, the user can move the view of the image: Up, Down, Left, or Right.</p> <p>MOVE scans the position of mouse pointer to see if it is inside one of the move arrows. It checks it every 100 ms but also starts checking when the pointer is actually inside one of the arrows and ends checking when the pointer moves out.</p> <p>This feature is located on the sliding toolbar and the top toolbar.</p>
<p>COPY</p> 	<p>Allows the user to copy a selected portion of the image onto the system clipboard (see Appendix Fig. 9).</p> <p>This feature uses a specialized SELECTION interface. When selecting a portion of the image to copy, two sets of arrows similar to the MOVE arrows appear on screen along with a translucent blue rectangle. The rectangle represents the portion of the image that will be selected. The set of arrows to the upper left of the selection rectangle allow the user to move the selection box, while the set of</p>

	<p>arrows to the bottom right allow the user to resize the selection box.</p> <p>If the user switches to another feature while using the SELECTION interface, the location of the selection box is maintained so that the user can easily switch back and forth between features.</p> <p>This feature is located on the sliding toolbar and the top toolbar.</p>
<p>CROP</p> 	<p>Allows the user to crop a portion of the image. It selects a portion of the image using the same SELECTION interface as the COPY feature and then removes everything except for the selected portion.</p> <p>This feature is located on the sliding toolbar.</p>
<p>PASTE</p> 	<p>Allows the user to paste data from the system clipboard onto the current image.</p> <p>This feature is located on the sliding toolbar.</p>
<p>PENCIL</p> 	<p>Allows the user to draw free-hand lines on the image. The user clicks (dwells) on the point where he wants to begin drawing and then dwells again on the point where he wants to stop drawing. The color of the pencil-line can be changed using the COLOR feature.</p> <p>Implemented using the .NET GDI classes.</p> <p>This feature is located on the sliding toolbar.</p>
<p>ROTATE</p> 	<p>Allows the user to rotate the image clockwise or counterclockwise 45, 90, or 180 degrees. When selecting between different degrees of rotation, the user can see a small preview of what the image will look like after being rotated (see Appendix Fig. 6).</p> <p>Implemented using .NET image class transformations (via a rotation matrix).</p> <p>This feature is located on the sliding toolbar.</p>
<p>ADJUST IMAGE</p>	<p>Allows the user to modify the BRIGHTNESS and CONTRAST of the image (see Appendix Fig. 7).</p>

	<p>BRIGHTNESS: 20 degrees of brightness ranging from -100 to +100.</p> <p>CONTRAST: 20 degrees of contrast ranging from -100 to +100.</p> <p>Both are implemented by modifying the pixel values of the image.</p> <p>This feature is located on the sliding toolbar.</p>
<p>COLOR</p> 	<p>Allows the user to change the color of the PENCIL feature.</p> <p>When launched, a new smaller window pops up with a color palette. The “Main Colors” consist of 8 preset colors: Black, White, Red, Orange, Yellow, Green, Blue, and Violet. Below those are the “Shades,” consisting of 8 different shades for each of the Main Colors.</p> <p>When the user clicks on one of the Main Colors, its shades are automatically generated. The user can then choose to select one of the shades or to continue with the main color (see Appendix Fig. 11).</p> <p>Shades are generated by incrementing or decrementing the RGB values of the selected Main Color by multiples of an internal shade increment number.</p> <p>This feature is located on the top toolbar.</p>
<p>UNDO</p> 	<p>Allows the user to undo up to 5 of his recent actions in any feature.</p> <p>Implemented by using a buffer of 5 different versions of the image. To store and manipulate the images in the buffer, a new data structure called a circular stack was designed. When an element is pushed onto the stack over a certain amount, it is written to the beginning of the stack and that position becomes the new top of the stack.</p> <p>This feature is located on the top toolbar.</p>
<p>REDO</p> 	<p>Allows the user to redo up to 5 actions that he undid with the UNDO feature.</p> <p>Implemented in the same way as the UNDO feature.</p> <p>This feature is located on the top toolbar.</p>
<p>EXIT</p>	<p>Allows the user to exit from the program. If changes have been made</p>

	<p>to the image, the program will prompt the user to save. Otherwise, the program will simply close.</p> <p>This feature is located on the sliding toolbar.</p>
---	---

Past Features and Modifications:

Feature/Modification	Description and Reason for change
<p>Sliding Toolbar Acceleration</p>	<p>Description: The more the user moves on the sliding toolbar, the faster the toolbar slides.</p> <p>Reason for change: The real time polling that the implementation used was too heavy on system resources and did not work as well as intended. We decided that a constant speed for the sliding toolbar was sufficient.</p>
<p>SELECTION – Version 1 ("Click and Drag")</p>	<p>Description: To select a portion of the image, the user first dwells on the spot where the top-left corner of the selection box will be and then dwells on the spot where the bottom-right corner of the selection box will be. The selection box is then generated on those two points.</p> <p>Reason for change: It was too difficult to make the selection precise. Users frequently selected areas by accident.</p>
<p>SELECTION – Version 2 ("Pushing")</p>	<p>Description: When selecting a portion of an image, a translucent blue selection box with an arrow on the top-left corner and an arrow on the bottom-right corner appears. The top-left arrow moves the selection box and the bottom-right arrow resizes the selection box. The user activates one of the arrow's functions by actually "pushing" the mouse pointer against a particular side of that arrow (see Appendix Fig. 10).</p> <p>Reason for change: Although many users liked this interface and thought it was fun</p>

	<p>to use, other users found the concept of pushing against the arrows difficult to grasp. Also, the SELECTION feature was not as precise as we had wanted, although it was a significant improvement over version 1.</p>
MOVE – Version 1	<p>Description: The way that the user interacted with MOVE v1 is the same as the current MOVE, but the implementation is slightly different. Both implementations scan the position of mouse pointer to see if it is inside one of the move arrows. V1 checked it every 100 ms or so, while v2 checks it every 100 ms as well, but also starts checking when the pointer is actually inside the arrows and ends checking when the pointer moves out.</p> <p>Reason for change: MOVE v1 was a CPU hog and would sometimes slow down the whole program if the computer had other programs running or was on “Power Saver” mode. MOVE v2 maintains the same interface but is a much more efficient implementation.</p>
COLOR with color mixing	<p>Description: The older version of the COLOR feature allowed the user to select three colors from the color palette: Color 1, Color 2, and a Mix of Color 1 and Color 2 (mix is generated by averaging the RGB values of Color 1 and Color 2). We chose this method to give the user more variety in her color choices.</p> <p>Reason for change: The concept was too complicated and confused users. It was also difficult for users if they wanted to just select one color without mixing.</p>

Initial Experiments:

We asked users to play around with our software. Based on their comments, we made changes to Camera Canvas. Their comments are listed in Fig. 2.

Name:	Age:	Knowledge of Computers (1 = No Knowledge, 10 = Expert):	Comments:
Jack	20	7	Scrolling needs to be faster Cropping was good

			Laughing messes it up
John	19	6	Crop is broke
Tucker	18	8.5	Scrolling good in general Cropping prone to accidents Tools cannot move above screen bounds
Justin	21	10	Cannot read PASTE Moving the corners of selection tool is frustrating Forget which part of arrow drags No way to cancel ZOOM Rotation enlarges the image Why show RGB?
Jen	22	10	Color selection is unclear Pencil needs indicator Brightness needs a preview
Peter	20	10	Hard to control PENCIL
Patrick	22	10	GUI different from Photoshop COLOR box icon unclear
Sachit	22	10	Keeping mouse pointer still makes MOVE feature move faster? Need to smooth out mouse movement
Sharon	52	7	Arrows are confusing Head needs to be too tilted to use toolbar Draw (PENCIL and COLOR) selections should be on top instead of left Rotation icons are not clear
Walter	52	5	Order of icons is not clear

Fig. 2. User information and comments.

User Study:

Experiment Goals:

- To observe the intuitiveness and ease-of-use of the program.
- To spot bugs that may occur from users using the program in unforeseen ways.
- To gauge user input to improve and refine existing features.

Methodology:

We conducted tests with 20 users. It should be noted that the majority of users were between 18 and 26 years of age except for three users over 45. We asked users to rank their knowledge of computers on a scale of 1 to 10, 1 meaning having no knowledge and 10 meaning being an expert.

Users were given a brief introduction to Camera Mouse and to the nature of the Camera Canvas program. Users were asked to select a feature on their face for Camera Mouse to track. Once they became familiar with the kinds of motion they could do with Camera Mouse

and how those motions translated to motions of the mouse pointer on the screen, we began usability tests of Camera Canvas.

At the beginning of each test, a user was asked to open the test image for that test using the OPEN feature. The test was then explained and once the user acknowledged that she understood the test, we began timing with a stopwatch. Time was stopped and recorded once the user completed the test successfully. If the user needed to take a break or reset Camera Mouse during the test, the timer was paused and then resumed.

Usability tests involved testing four different software features. Each feature was associated with a specific image (see Appendix Fig. 12 - 15 for the test images):

Features Tested:

1. Copy/Paste (/Select) – Users had to copy a small cat and paste him into a small red circle.
2. Zoom (/Draw) – Users had to draw a circle using the pencil tool and then zoom in on that circle at 400%.
3. Move (/Crop) – Users had to move the image to see the cat hiding at the bottom and then crop him out.
4. Crop (/Select) – Users had to crop out the man in the center of an image containing multiple people.

Users were given an opportunity to repeat a test if they were unhappy with the initial result. Only in 6 instances did users redo a test. Overall, we have 4 tests * 20 users + 6 repeat tests = 86 data points.

Results:

The average rank of computer knowledge was 7.7. On average, it took 2 minutes and 11 seconds to complete a feature test (see Fig. 3).

Feature	Average time (minutes: seconds)
Copy/Paste	02:32
Draw/Zoom	01:23
Move/Crop	03:44
Crop	01:06
All Tests	02:11

Fig. 3. Average Times for Test Cases.

Also, for features like COPY and MOVE, it should be noted that the highlighted area that the user moved was slow moving. This means that even though the user may have known how to use a feature and was using it perfectly, the time might have been a little bit high because of the slow movement of the highlighted area.

The older users with ages over 45, usually had a harder time with our program. The main reason here was because they were just not used to using Camera Mouse. Although after adapting to the use of Camera Mouse, the tests went a lot quicker.

We compared the times for completing tests with the self-stated computer knowledge level of users and found that the time for a user to complete a task decreased as his/her computer expertise increased (see Fig. 4a). Inversely, the time for a user to complete a task increased with his/her age (see Fig. 4b).

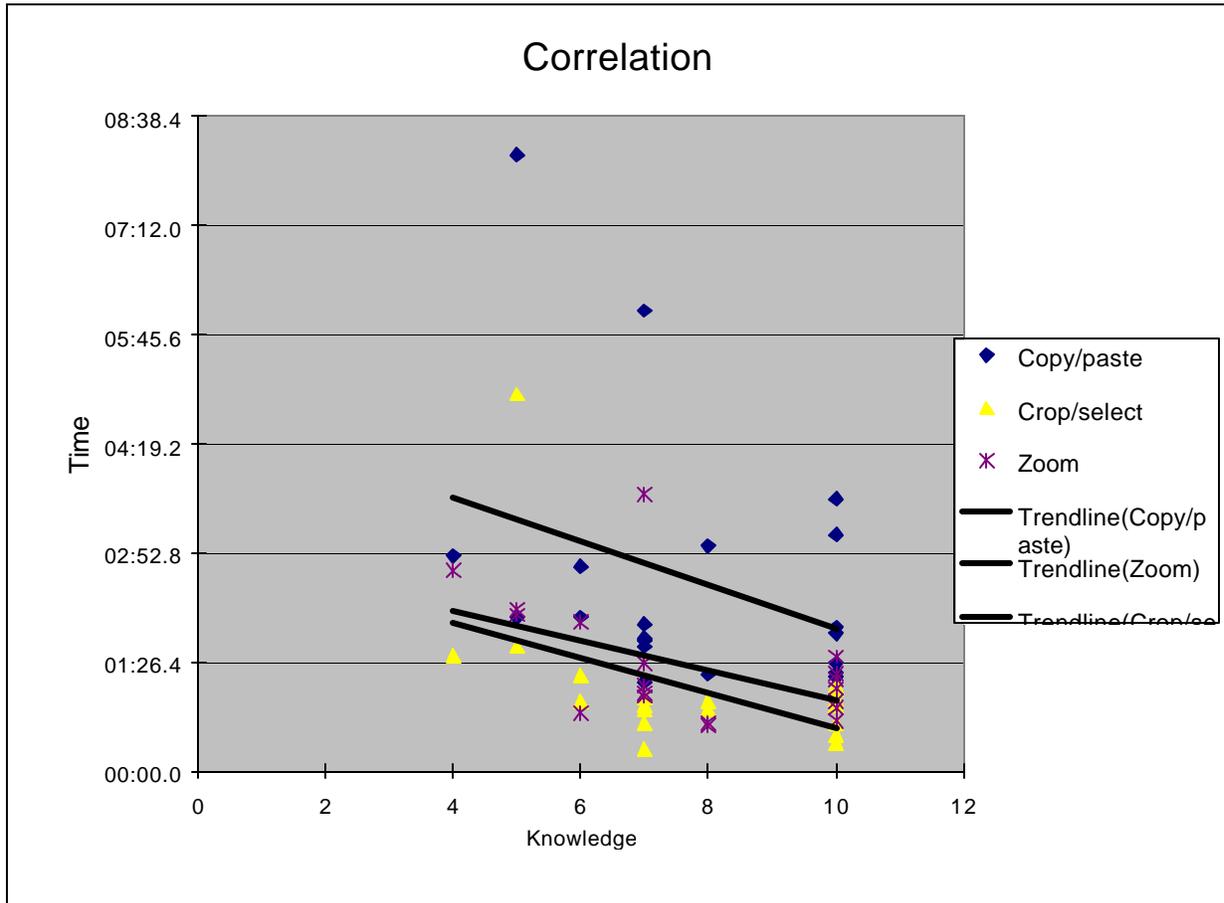


Fig. 4a. Graph of correlation between level of computer knowledge and time to complete 3 tests (using first data point for outlier cases). The linear trend line shows that the time for a user to complete a task decreased as his/her computer expertise increased.

<With Outlier>

Subject	Knowledge	Age	Copy/Paste(/Select)	Zoom(/Draw)	Move(/Crop)	Crop(/Select)	Net time
1	7	20	01:43.8	01:07.4	08:20.2	00:38.2	11:49.6
2	6	19	02:02.0	00:46.2	02:03.2	00:55.4	05:46.8
3	8	18	01:16.4	00:36.3	01:51.4	00:55.4	04:39.5
4	10	21	03:06.9	01:16.8	03:02.2	01:08.0	08:33.9
5	10	22	03:35.4	01:12.7	05:02.2	00:56.9	10:47.2
6	10	20	01:49.2	00:50.3	02:12.7	00:52.5	05:44.7
7	10	22	01:14.2	00:55.6	02:14.3	00:28.1	04:52.2
8	7	18	01:56.4	01:01.7	05:20.3	00:51.5	09:09.9
9	10	20	01:53.8	01:30.7	03:24.9	00:21.8	07:11.2
10	10	21	01:18.3	00:39.9	01:51.2	01:04.2	04:53.6
11	7	19	01:45.1	01:25.5	01:36.2	00:48.3	05:35.1
12	10	22	01:25.6	01:05.8	09:32.2	00:38.2	12:41.8
13	7	52	06:04.2	03:39.0	09:58.0	00:17.1	19:58.3
14	6	14	02:42.1	01:58.1	05:29.6	01:15.2	11:25.0
15	5	52	08:07.1	02:08.2	01:32.2	04:58.5	16:46.0
16	5	22	02:01.5	02:03.8	02:59.5	01:38.4	08:43.2
17	8	23	02:58.4	00:38.2	01:02.4	00:48.5	05:27.5
18	7	23	01:38.6	00:59.4	02:01.4	00:58.4	05:37.8
19	4	52	02:50.0	02:38.9	03:12.1	01:31.5	10:12.5
20	7	23	01:10.8	00:59.4	01:50.5	00:58.4	04:59.1
Average	8	25	02:31.99	01:22.69	03:43.84	01:06.23	08:44.74
Std	1.98	11.77	01:44.0	00:46.3	02:42.9	00:58.3	04:14.0
Correlation	with knowledge		-0.33	-0.50	0.06	-0.47	-0.29
	with age		0.76	0.76	0.16	0.50	0.67

Fig. 5a. Data from user tests with first data point used for outlier cases.

<Without Outlier>

Subject	Knowledge	Age	Copy/Paste(/Select)	Zoom(/Draw)	Move(/Crop)	Crop(/Select)	Net time
1	7	20	01:43.8	01:07.4	02:01.0	00:38.2	11:49.6
2	6	19	02:02.0	00:46.2	02:03.2	00:55.4	05:46.8
3	8	18	01:16.4	00:36.3	01:51.4	00:55.4	04:39.5
4	10	21	03:06.9	01:16.8	03:02.2	01:08.0	08:33.9
5	10	22	03:35.4	01:12.7	05:02.2	00:56.9	10:47.2
6	10	20	01:49.2	00:50.3	02:12.7	00:52.5	05:44.7
7	10	22	01:14.2	00:55.6	02:14.3	00:28.1	04:52.2
8	7	18	01:56.4	01:01.7	05:20.3	00:51.5	09:09.9
9	10	20	01:53.8	01:30.7	03:24.9	01:30.2	07:11.2
10	10	21	01:18.3	00:39.9	01:51.2	01:04.2	04:53.6
11	7	19	01:45.1	01:25.5	01:36.2	00:48.3	05:35.1
12	10	22	01:25.6	01:05.8	09:32.2	00:38.2	12:41.8
13	7	52	06:04.2	03:39.0	09:58.0	00:17.1	19:58.3
14	6	14	02:42.1	01:58.1	05:29.6	01:15.2	11:25.0
15	5	52	08:07.1	02:08.2	01:32.2	04:58.5	16:46.0
16	5	22	02:01.5	02:03.8	02:59.5	01:38.4	08:43.2
17	8	23	01:27.0	00:38.2	01:02.4	00:48.5	05:27.5
18	7	23	01:38.6	00:59.4	02:01.4	00:58.4	05:37.8
19	4	52	02:50.0	02:38.9	03:12.1	01:31.5	10:12.5
20	7	23	01:10.8	00:59.4	01:50.5	00:58.4	04:59.1
Average	8	25	02:27.42	01:22.69	03:24.88	01:09.65	08:44.74
Std	1.98	11.77	01:44.8	00:46.3	02:30.7	00:57.6	04:14.0
Correlation	with knowledge		-0.34	-0.50	0.12	-0.41	-0.29
	with age		0.77	0.76	0.23	0.48	0.67

Fig. 5b. Data from user tests with second data point used for outlier cases.

Conclusion:

Our ultimate goal for this project was to create a fully functional image editing software package specialized for people with disabilities. In order to meet this objective, we set out to study the needs of the target user base, came up with creative solutions addressing those needs, and implemented a working image editor incorporating our solutions.

We learned that classical solutions for known problems were not necessarily suitable for all users or input systems. Even for simple problems, like moving the image, entirely new solutions had to be thought up. Our early solutions were merely extensions of orthodox image editing interfaces and were met with limited success. Through trial and error, we found innovative designs that were more optimal for certain problems. For example, to address the problem of selecting a portion of an image, the traditional “click-and-drag” interface was replaced by our “pushing” interface, which in turn was replaced by an interface involving our MOVE feature.

In the end, due to the time constraints, we were only able to perform limited user testing. We were not able to conduct any user studies with users with disabilities, so our application’s degree of success in serving disabled users still requires further exploration. However, our effort was still fruitful in the sense that we were able to assess the needs of the user base and learn the difficulties involved, allowing us to come up with suitable solutions. The software that we created was robust and worked well with the Camera Mouse interface, as well as with handheld mice and touch-pads.

Since our program implements a significant subset of the common features of modern image editors, allowing the user to perform basic image manipulations, and considering the lack of other software for Camera Mouse that offer the same functionality, we believe that our project would be a valuable addition to the Camera Mouse family of applications.

We hope that our project will inspire others to take on projects that focus on users with specialized needs. In creating our program, we hope to bring more attention to developing applications around the Camera Mouse input system in order to further expand the computing options available to users with disabilities.

Future Plans:

Future work involves expanding the feature set and getting more user-input in order to refine the new features and make Camera Canvas a full-fledged image editor.

Features that we hope to implement in the future are:

- Varied brush sizes and shapes for the PENCIL feature.
- A painting component that includes eraser, line drawing, geometric shapes, and bucket fill.
- A file navigation system specialized for Camera Mouse for opening images in other directories.

- A text input system or integrating an existing Camera Mouse keyboard system into the program.
- A layering system to allow an image to have multiple layers.

Acknowledgments:

- The Camera Mouse team, without whose work this project would not have been possible.
- Boston University Computer Science Laboratory graduate students, for their valuable feedback at the infant stage of our project.
- Our friends and family, for being our guinea pigs.

References:

- M. Betke, J. Gips, and P. Fleming, "The Camera Mouse: Visual Tracking of Body Features to Provide Computer Access For People with Severe Disabilities." *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 10:1, pp. 1-10, March 2002.
- Bob Powell's GDI+ FAQ - http://www.bobpowell.net/gdiplus_faq.htm
- Camera Canvas Google Code Page - <http://code.google.com/p/cameracanvas/> (or cameracanvas.googlecode.org)
- Camera Mouse - <http://www.cameramouse.org/>

We also compared the features of our program with existing image editors:

- Adobe Photoshop - <http://www.adobe.com/products/photoshop/index.html>
- Gimp - <http://www.gimp.org/>
- Microsoft Paint - <http://windowshelp.microsoft.com/Windows/en-US/help/f5feb1df-8dd7-4ab0-9f65-3c1c89a329ab1033.mspx>
- openCanvas - <http://www.portalgraphics.net/en/>
- paint.net - <http://www.getpaint.net/>

Technologies Used:

- Microsoft Visual C#
- Microsoft .NET Framework
- Google Code Development Website

Appendix:

The appendix shows the images that we used to test the Camera Canvas system.



Fig. 6. Using the ROTATE feature.

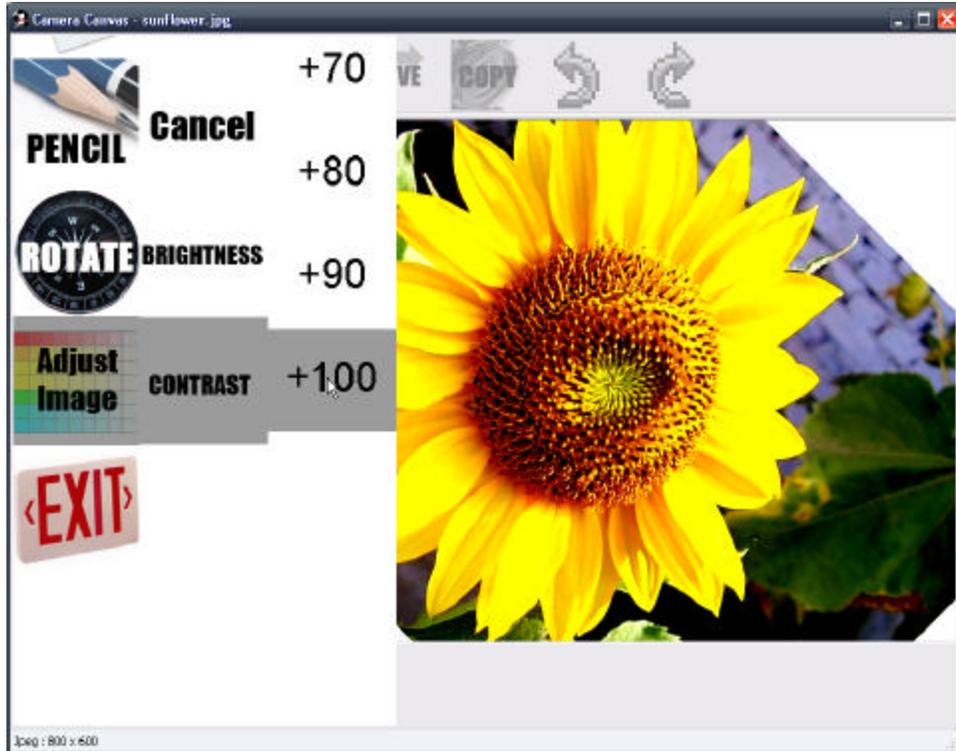


Fig. 7. Using CONTRAST in the ADJUST IMAGE feature.

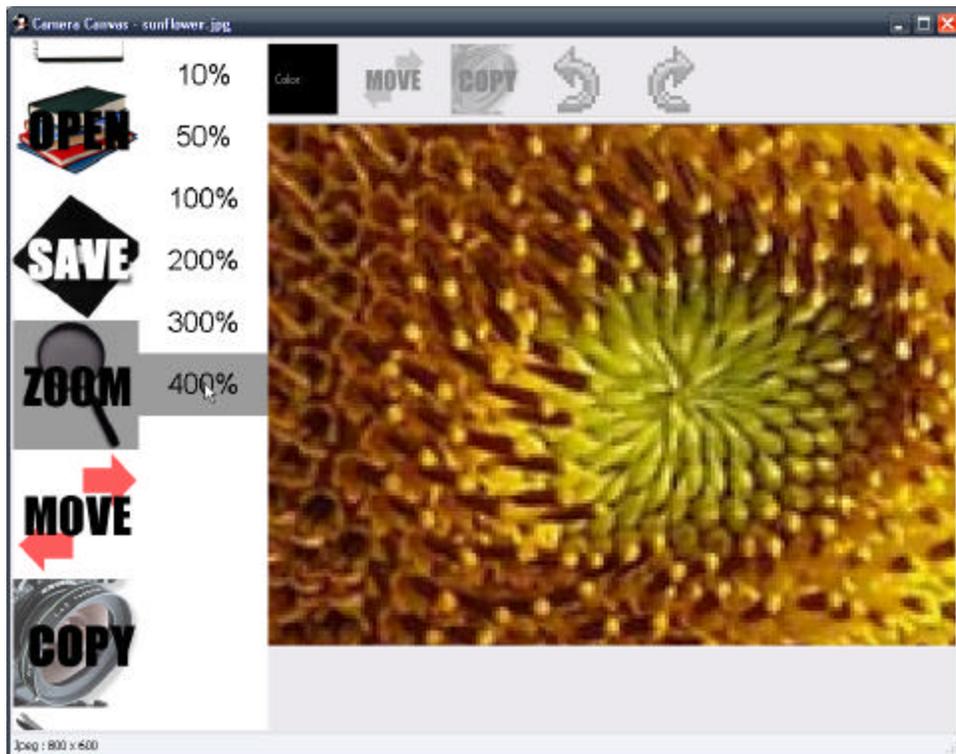


Fig. 8. Using the ZOOM feature.

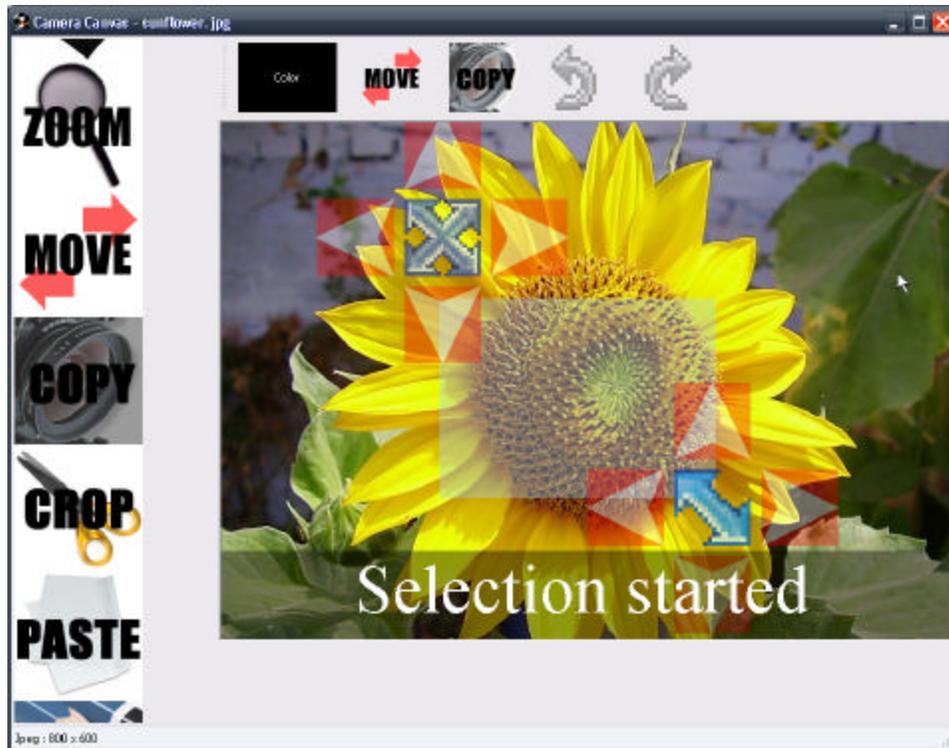


Fig. 9. Using the COPY and SELECTION features.

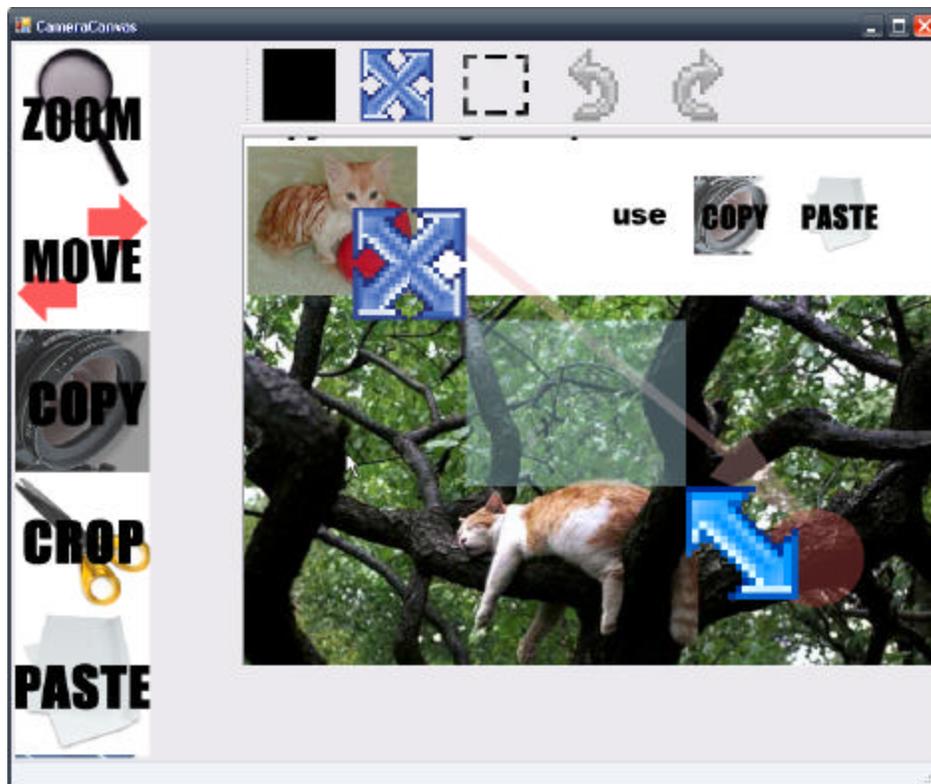


Fig. 10. The older “pushing” implementation of the SELECTION feature.

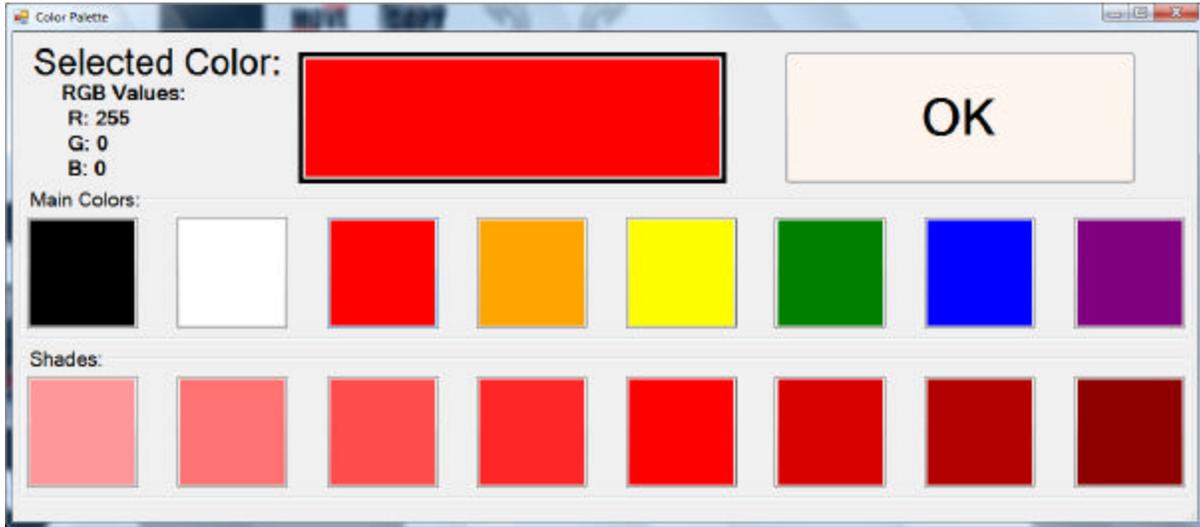


Fig. 11. The Color Palette window from the COLOR feature.

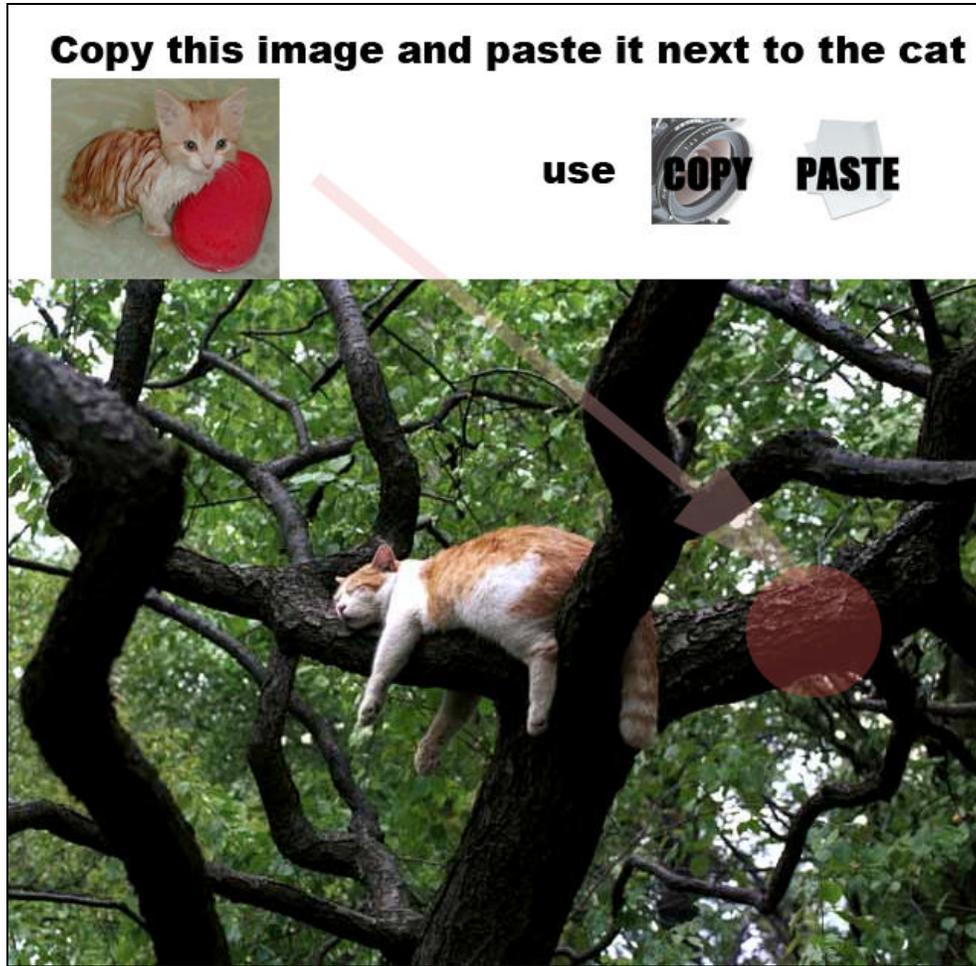


Fig. 12. The Copy/Paste (/Select) Feature test image.

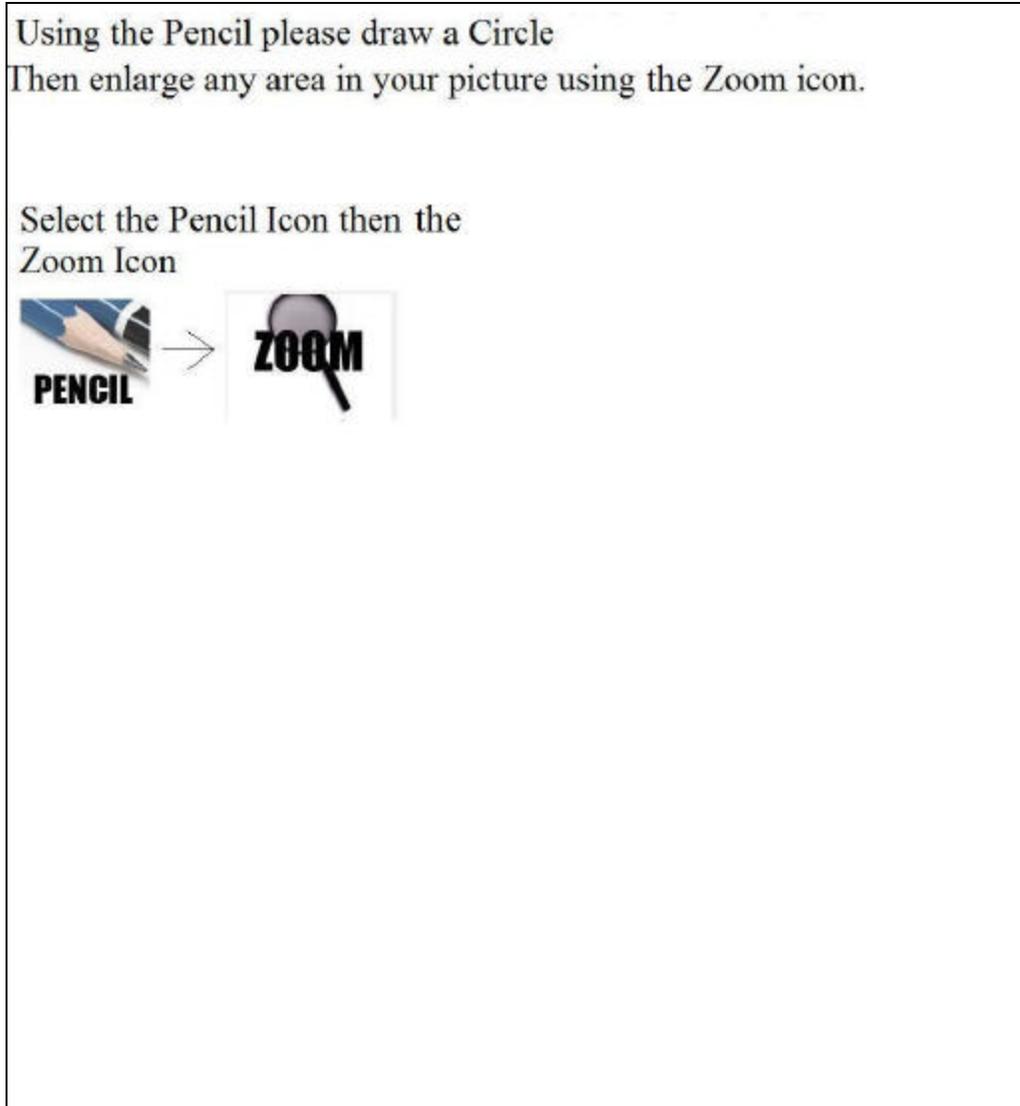


Fig. 13. The Zoom (/Draw) feature test image.

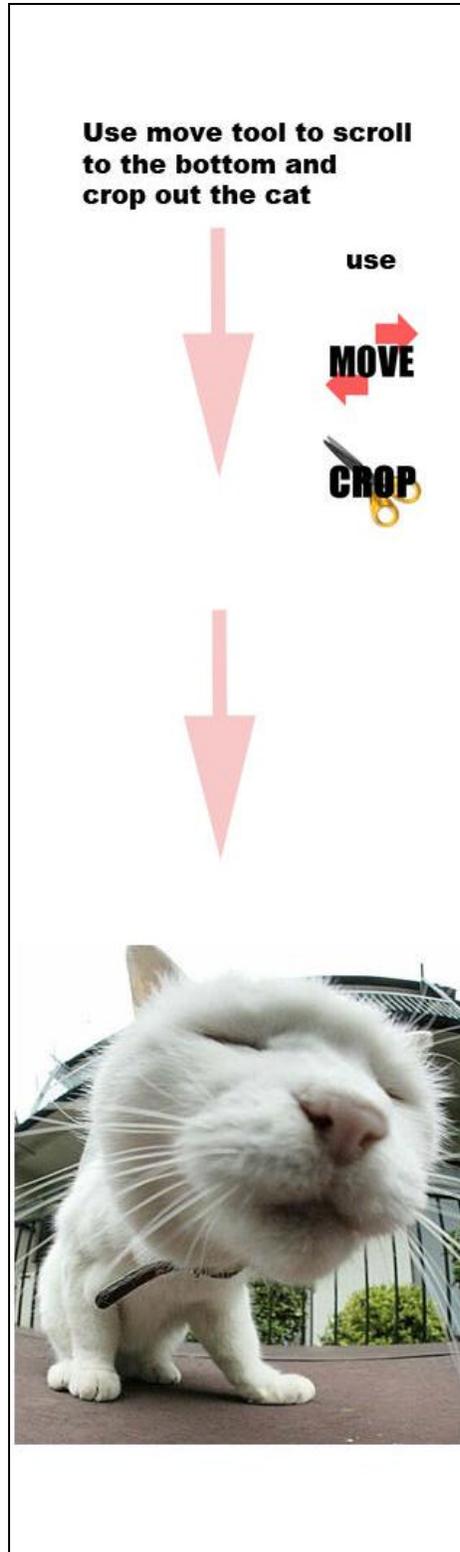


Fig. 14. The Move (/Crop) feature test image.

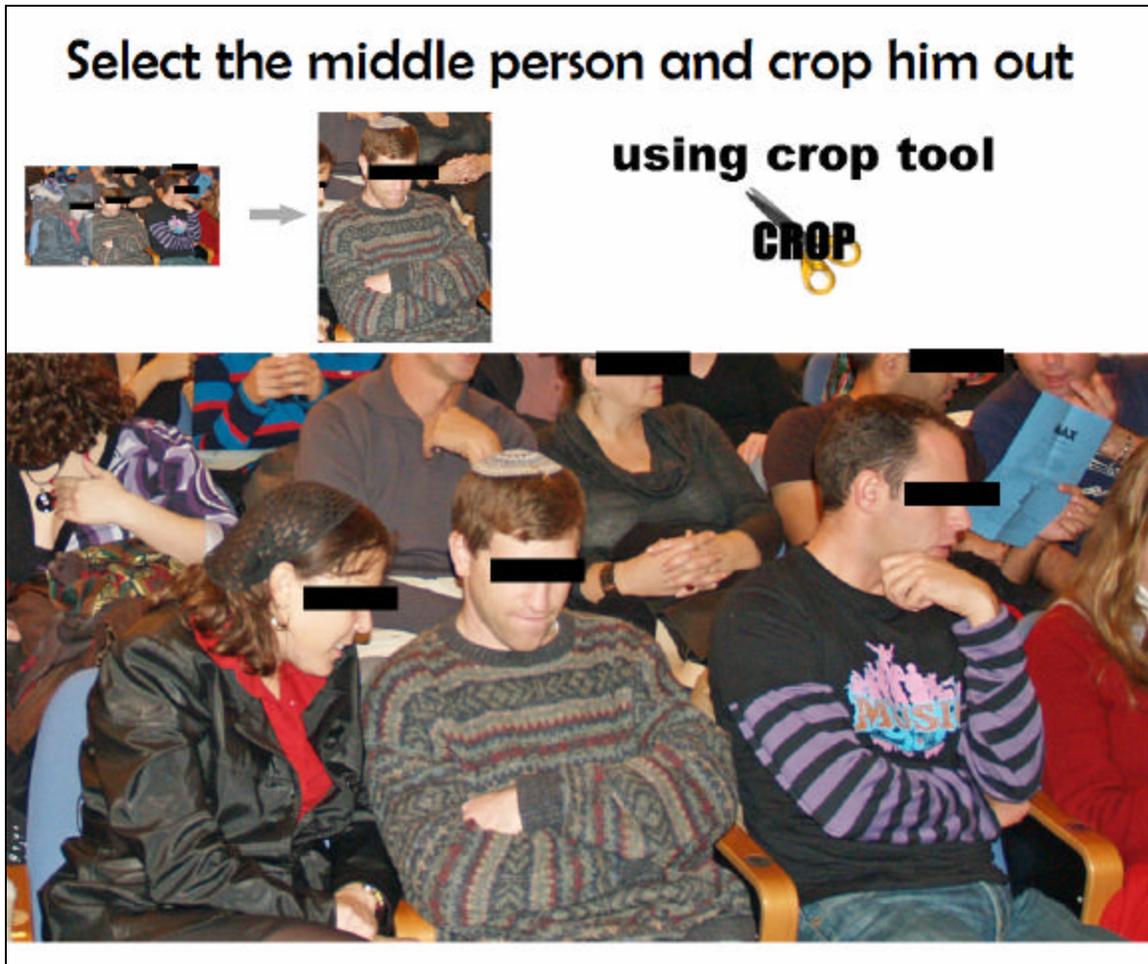


Fig. 15. The Crop (/Select) feature test image.