Author(s):	Heikkilä, Henna
Title:	EyeSketch: A Drawing Application for Gaze Control
Main work:	Proceedings of the 2013 conference on Eye Tracking South Africa (ETSA '13)
Year:	2013
Pages:	71-74
ISBN:	978-1-4503-2110-5
Publisher:	ACM
Series and number:	ACM Symposium on Eye Tracking Research & Applications
Discipline:	Computer and information sciences
School /Other Unit:	School of Information Sciences
Item Type:	Article in Conference Proceeding
Language:	en
DOI:	http://dx.doi.org/10.1145/2509315.2509332
URN:	URN:NBN:fi:uta-201311061562

EyeSketch: A Drawing Application for Gaze Control

Henna Heikkilä
Tampere Unit for Computer-Human Interaction
School of Information Sciences
University of Tampere
Finland
henna.heikkila@sis.uta.fi

ABSTRACT

We present a gaze-driven drawing application called EyeSketch. Contrary to earlier gaze-controlled drawing applications, our application utilizes drawing objects that can be moved and resized, and their color attributes can be changed after drawing. Tool and object selections are implemented with dwell buttons. Tools for moving and resizing are controlled with gaze gestures and by closing the eyes. Our gaze gestures are simple, one-segment gestures that end outside the screen area. The gestures are used to give the direction for moving and also for the command to make the object either smaller or larger. Closing the eyes signals to the application to stop a moving object. In our evaluations, these gaze gestures were judged as a usable interaction style for the moving and resizing purpose.

Categories and Subject Descriptors

H.5.2 [Information Interfaces and Presentation]: User Interfaces – *Graphical user interfaces (GUI), Input devices and strategies (e.g., mouse, touchscreen), Interaction styles (e.g., commands, menus, forms, direct manipulation).*

General Terms

Design, Human Factors.

Keywords

Gaze interaction, eye tracking, gaze gestures, drawing application, drawing with gaze

1. INTRODUCTION

After concentrating on day-to-day communication of users with special needs, the focus of the eye-gaze research community has slowly shifted to games, online communities, and artistic applications, such as music players and drawing applications. These leisure applications invite the disabled users to take part in such activities that are routine for able-bodied users.

In gaze-controlled drawing applications, the same sensory channel is used both to control the input and to study the output. This can cause the so-called Midas touch problem [9], if the two actions are not separated clearly enough from each other. Furthermore, the inaccuracy and the natural jitter of the gaze as well as the accuracy limitation and the possible calibration errors of the eye tracker make gaze a difficult input channel for the drawing task that often requires accurate pointing in order to make the picture look clean. Similar challenges are present in most gaze-controlled applications.

Our motivation was to design a new kind of gaze-controlled drawing application that enables the user to modify and edit the drawing afterwards. We searched for the best interaction method for the modifying tools, which led us to use gaze gestures for moving and resizing the objects.

In this paper, we describe our gaze-controlled drawing application called EyeSketch. First, we will go through the previous drawing applications for gaze control. Then, we move to describe our own application and its tools including the gaze gesture implementation. Last, we sum up the advantages of our drawing application and give some future directions to our work.

2. RELATED RESEARCH

Most previous drawing applications for gaze only have had two major limitations. First, the size and the position of a shape are determined when drawing it and editing after the drawing is not allowed. Second, when drawing shapes, their color has to be decided before drawing. Especially the first restriction causes inconvenience as the eye tracker's accuracy or calibration errors can make the accurate positioning very difficult.

Most gaze-controlled drawing applications let the user create their drawing from predetermined shapes, such as rectangles, ellipses, and lines. Another approach is so-called free-eye drawing [12], where the drawing appears on the screen according to the user's eye movements. Eye Painting for EagleEyes by Gips and Olivieri [1] is a drawing application that is based on the free-eye drawing method. In Eye Painting, a colorful line appears wherever the user looks at on the screen.

Drawing a representational picture with the free-eye drawing method is extremely difficult because of the following three limitations. Gaze is not well suited for doing precise movements. Gaze movements are also sensitive for distractions, as we tend to look away from the screen if we hear a noise or something flashes in the corner of one's eye. The most profound limitation in free-eye drawing is that it does not separate drawing and looking. This means that the user cannot use the most basic function of the eyes, that is looking, without causing a drawing to appear as well.

EyeDraw [4, 5, 6] was the first gaze-controlled drawing application that separated the drawing process from looking by utilizing the aforementioned predetermined shapes. In EyeDraw, a two-step process was used to determine whether the user wants to draw or to keep looking around. After the user's gaze stays relatively still for 500 milliseconds, the application starts to wait for the drawing command, which is issued when the user's gaze stays fixated for another 500 milliseconds. When the two-step process is followed through for the first time, the gaze point is marked as the starting point for the shape. The second completion of the process marks the ending point and finishes the drawing of the shape.

EyeArt is a more recent drawing application for gaze control [10]. EyeArt and EyeDraw have many similar design solutions, but in EyeArt, the user has a wider selection of tools to create the drawing. The new tools in EyeArt include a text tool, an eraser, and a paint can tool for coloring already drawn shapes. The paint can tool is the first tool in gaze-controlled drawing applications that lets the user do something to already drawn shapes besides delete them.

The weakness in EyeDraw and in EyeArt is that the user cannot move or resize the drawn object afterwards. This means the pictures most probably will have unwanted gaps between shapes and unintentional overlapping of shapes. Making the shapes fall into the right place in the right size when drawing them is difficult, as the gaze is a more inaccurate input channel than for example a mouse or a finger.

Both in EyeDraw and in EyeArt looking is distinguished from drawing on the basis of dwell time. Yeo and Chiu [14] created a Gaze Estimation Model for recognizing whether the user wants to draw or to keep looking. The model calculates several metrics which are used in the model to decide whether to start the drawing process or not and also where the user wants to draw the shape. The model is intended to solve the problem of unwanted gaps and overlapping of the shapes. However, no evaluations of its behavior in practice have been published.

Van der Kamp and Sundstedt [13] replaced dwell time with voice commands. In their application, gaze was used to control the cursor and voice commands were used to open menus, to select tools from menus, and to control drawing. The method was assessed as enjoyable even though the application had trouble to recognize some female voices.

The previous work shows how two issues can make drawing a very difficult task. First, the accuracy of the gaze points may vary, so the application has to manage the possible inaccuracy. Second, the separation between looking and drawing is the key for drawing clean, representational pictures.

3. IDEAS BEHIND EYESKETCH

We designed and implemented a new drawing application called EyeSketch (Figure 1). In EyeSketch, the user creates the pictures by drawing objects that they can move and resize. The color and border properties of the objects are also modifiable. This approach was selected, as it was evident from pervious drawing applications that drawing perfectly placed shapes was troublesome.

Tools in EyeSketch are controlled by dwelling or with gaze gestures. We use dwell time for selecting tools and objects. Dwell times are kept relatively short and a feedback sound is played when the dwell time is reached. Gaze gestures are used to avoid cluttering the drawing canvas excessively with tool buttons. Gaze gestures are also less sensitive for calibration errors and usually can be used when hitting dwell buttons has became difficult due to increasing jitter in the eyes or calibration errors.

COGAIN ETU Driver[1] is integrated to our application and it retrieves the raw gaze point data from the eye tracker and delivers it to EyeSketch. ETU Driver ensures that EyeSketch can be used with several different eye trackers.

3.1 Tool Buttons

Tool buttons are arranged around the drawing canvas (Figure 1). Buttons are selected by dwelling on them for 400 milliseconds. The size of each tool button is 80 × 80 pixels. A small crosshair is shown on the user's gaze point. The color of the crosshair changes to give visual feedback to the user during drawing and gesture processes.

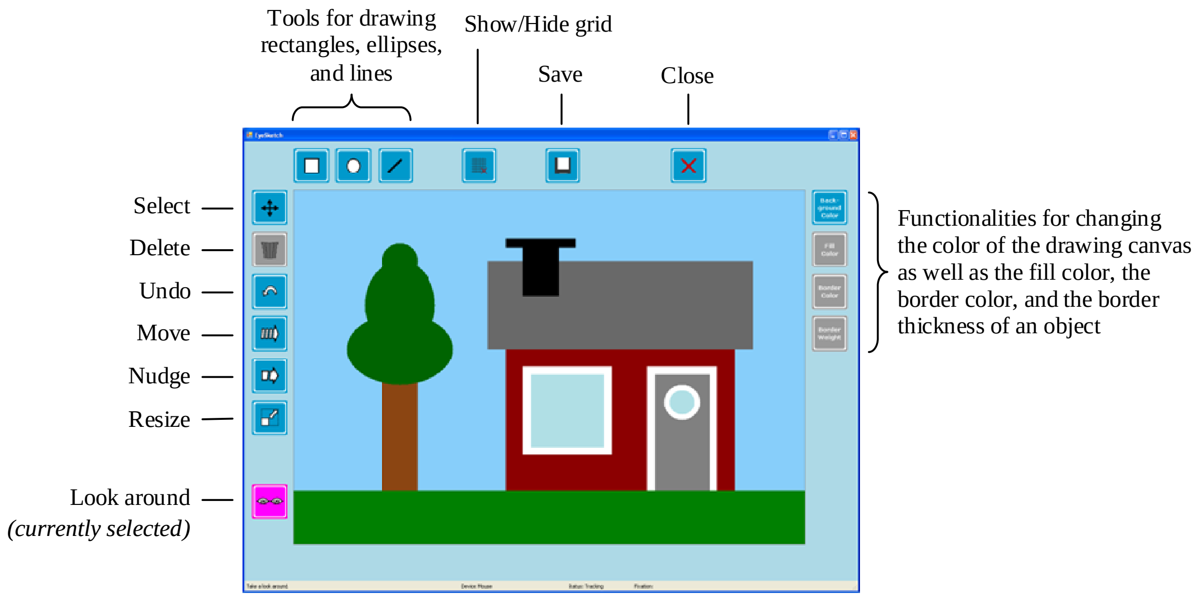
Tools for drawing rectangles, ellipses, and lines
Show/Hide grid
Save
Close
Select
Delete
Undo
Move
Nudge
Resize
Look around
(currently selected)
Functionalities for changing the color of the drawing canvas as well as the fill color, the border color, and the border thickness of an object

Figure 1. A screen shot from EyeSketch.

[1] COGAIN ETU Driver can be downloaded from http://www.sis.uta.fi/~csolsp/projects.php#ETUDriver

With EyeSketch, the user draws objects to create the picture. The application includes tools for making rectangles, ellipses, and lines. The drawing happens by first dwelling on the button of a drawing tool, then dwelling on the starting point of the object on the drawing canvas, and then on the ending point of the object. After the starting point is defined, the object is drawn based on the user's gaze point until they dwell again on a point in the drawing canvas (giving the ending point) or until they stop the drawing process by dwelling on any of the tool buttons.

We also implemented some supporting tools and functionalities. *Save* and *Open* functionalities are implemented to enable the user to continue the same drawing later. All objects can be deleted with the *Delete* button and all actions can be undone with the *Undo* button. The *Select* tool is used when the user want to delete an object or to modify an object's color or line properties. When the *Select* tool is in use, the user can select any object by looking at it for 100 milliseconds. When the *Look around* tool is selected, it ends all processes and lets the user look at the drawing without fear that something would happen to it.

A grid was added on the drawing canvas to help in positioning the objects. All objects snap to grid, which makes the drawing of connecting objects easier. The user can adjust the visibility of the grid with the *Show* / *Hide Grid* button.

3.2 Gaze Gestures

We designed a set of eight gaze gestures for the needs of our drawing application. The tools for moving and resizing utilize these gestures. Our gaze gestures are simple, one-segment gestures. They always start on the object to be moved or on a resizing handle of the object to be resized, and end outside the screen area (Figure 2).

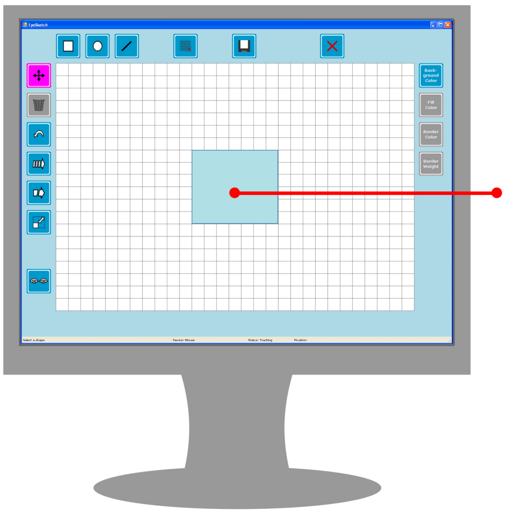

Figure 2. Gaze gestures start on the movable object (or on a resizing handle) and end outside the screen area.

Our gaze gestures combine the ideas from several previous studies. Møllenbach et al. [11] presented their simple gaze gestures in 2010. Their gestures are done from one side of the screen to another side whereas ours start from limited space and need to move relatively straight outside the screen. The use of the space beyond the screen area was originally suggested by Isokoski [7]. He used off-screen targets for text input via gaze. In a way, Istance et al. [8] already combined simple gaze gestures and the use of off-screen space in their SnapClutch application. However, in their implementation, the area where the eyes end was meaningful, whereas in our gaze gestures, the path of the gesture is more meaningful, and the off-screen space is only used to help to differentiate gestures from the gaze data.

The gesture process is started by staying on the object or on the resizing handle for 100 milliseconds. As the cursor changes its color to green, the user knows they can proceed. Now, the user can move their gaze outside the screen area. When the gaze has stayed outside the screen area for 100 milliseconds, a feedback sound is played, the command is performed, and the user may return their gaze back to the screen.

When making the gesture from the object or handle to outside the screen, three gaze points must fall into the same gesture area (see Figure 3) before exiting the screen. With 60Hz eye trackers, this means that the movement from object to outside the screen must take at least 64 milliseconds the gesture to be recognized. If the movement takes more than 1500 milliseconds, the gesture process is stopped.

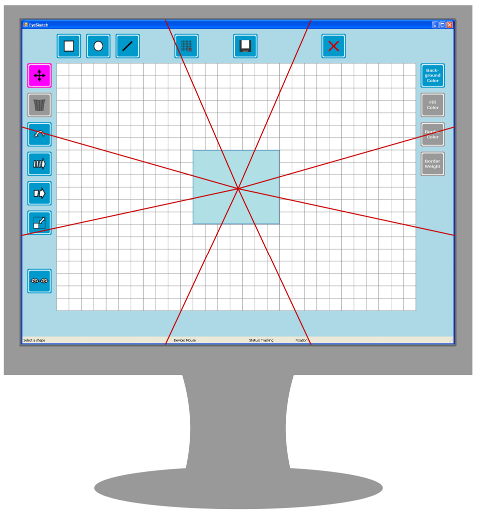

Figure 3. For moving, eight gesture areas are used to determine the moving direction.

The overall design of EyeSketch and its gaze gestures based on our previous studies [2, 3]. When evaluated against dwell time, the gestures were found to be a very usable method for moving and especially for resizing the objects. In a moving task, gaze gestures cannot beat dwell in performance time, but they are easier to use if the gaze point has any error due to calibration or posture shifting. For resizing, gaze gestures were less error-prone than dwell time, as the resizing buttons were situated next to each other in the dwell-based design and it was very common to click accidentally the wrong one. The results from the evaluation encouraged us to proceed with the gaze gestures as an interaction method.

3.3 Tools for Modifying Objects

The most profound difference to previous drawing applications is that the drawn objects can be moved and resized afterwards. In addition, their fill and border color as well as the border thickness can be modified before and after drawing. The user can also change the color of the canvas anytime.

When using the *Move* and the *Nudge* tools for moving, the user first needs to dwell on the tool button, and then they can move their gaze on the object they want to move. To move the object, the user has to do a gaze gesture (described on the previous section) to make the object move. The gesture direction itself determines to which direction the object will be moved.

With the *Nudge* tool, the object moves one step to the direction given through the gaze gesture. When using the *Move* tool, the object is put in motion with the gaze gesture. The object moves gradually towards the given direction until the user closes both eyes, or until the object reaches the border of the drawing canvas. The user needs to keep the eyes closed for at least 300 milliseconds. A feedback sound is played when the time limit is reached and the user may open their eyes.

When using the *Resize* tool, the user first needs to select the resizing tool by dwelling on its button and then move their gaze on the object they want to resize. After a brief dwelling (i.e. for 100 milliseconds) on the object, the resizing handles (see Figure 4) appear, and then the user can perform the gesture process as described earlier. The size of the resizing handle is 50 × 50 pixels.

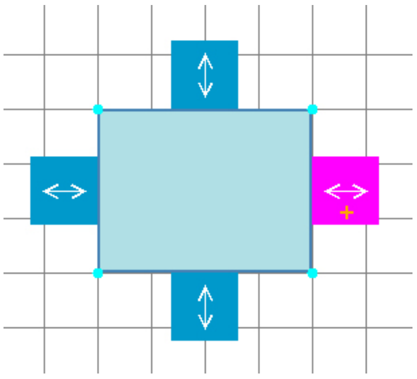

Figure 4. When using the Resize tool, resizing handles appear after dwelling on the object for 100 milliseconds.

The resizing handles are used to determine whether the user wants to make the object smaller or larger. If the user wants to make the object smaller from one side, they start the gesture from the handle on that side, move through the object, and end outside the screen. If the user wants to make the object larger, they start on the handle and move away from the object ending up outside the screen. The resizing handles have arrows to show the available gesture directions.

4. CONCLUSION

We have presented EyeSketch, a drawing application for gaze only use. Our approach is new for gaze-controlled drawing applications for two reasons. First, we let the user adjust the drawing afterwards, which helps to avoid the unwanted gaps and overlapping of shapes. Second, we use gaze gestures for some tools to make the interaction easier and to free space for drawing.

The development of EyeSketch has been done in conjunction with experimental research on gaze gestures and modifying tools. In the near future, we will evaluate and refine the application with our target user group, i.e. disabled users, and then make it available for public.

5. ACKNOWLEDGMENTS

This work was supported by the Finnish Doctoral Program in User-Centered Information Technology (UCIT).

6. REFERENCES

[1] Gips, J. and Olivieri, P. 1996. EagleEyes: An Eye Control System for Persons with Disabilities. In *Proceedings of 11th International Conference on Technology and Persons with Disabilities* (CSUN'96). Los Angeles, USA.

[2] Heikkilä, H. 2013. Tools for a Gaze-Controlled Drawing Application – Comparing Gaze Gestures against Dwell Buttons. In P. Kotzé et al. (eds.) 2013. *INTERACT 2013, Part II, LNCS 8118*, 187–201.

[3] Heikkilä H. and Räihä, K.-J. 2012. Simple Gaze Gestures and the Closure of the Eyes as an Interaction Technique. In *Proceedings of 2012 Symposium on Eye Tracking Research & Applications* (ETRA'12). ACM, New York, USA, 147–154.

[4] Hornof, A. J. and Cavender, A. 2005. EyeDraw: Enabling Children with Severe Motor Impairments to Draw with Their Eyes. In *Proceedings of SIGCHI Conference on Human Factors in Computing Systems* (CHI'05). ACM, New York, USA, 161–170.

[5] Hornof, A., Cavender, A., and Hoselton, R. 2004a. EyeDraw: A System for Drawing Pictures with Eye Movements. In *Proceedings of 6th International ACM SIGACCESS Conference on Computers and Accessibility* (ASSETS'06). ACM, New York, USA, 86–93.

[6] Hornof, A., Cavender, A., and Hoselton, R. 2004b. EyeDraw: A System for Drawing Pictures with the Eyes. In *Extended Abstracts on Human Factors in Computing Systems* (CHI'04). ACM, New York, USA, 1251–1254.

[7] Isokoski, P. 2000. Text Input Methods for Eye Trackers Using Off-Screen Targets. In *Proceedings of 2000 Symposium on Eye Tracking Research & Applications* (ETRA'00). ACM, New York, USA, 15–21.

[8] Istance, H., Bates, R., Hyrskykari, A., and Vickers, S. 2008. Snap Clutch, a Moded Approach to Solving the Midas Touch Problem. In *Proceedings of 2008 Symposium on Eye Tracking Research & Applications* (ETRA'08). ACM, New York, USA, 221–228.

[9] Jacob, R. J. K. 1991. The Use of Eye Movements in Human-Computer Interaction Techniques: What You Look At is What You Get. *ACM Transactions on Information Systems* 9 (2), 152–169.

[10] Meyer, A. and Dittmar, M. 2009. *Conception and Development of an Accessible Application for Producing Images by Gaze Interaction - EyeArt.* Retrieved from http://wiki.cogain.info/images/d/da/EyeArt_Documentation.pdf.

[11] Møllenbach, E., Lillholm, M., Gail, A., and Hansen, J.P. 2010. Single Gaze Gestures. In *Proceedings of 2010 Symposium on Eye Tracking Research & Applications* (ETRA'10). ACM, New York, USA, 177–180.

[12] Tchalenko, J. 2001. Free-Eye Drawing. *Point* 11, 36–41.

[13] Van der Kamp, J. and Sundstedt, V. 2011. Gaze and Voice Controlled Drawing. In *Proceedings of 1st Conference on Novel Gaze-Controlled Applications* (NGCA'11). ACM, New York, USA.

[14] Yeo, A. W. and Chiu, P.-C. 2006. Gaze Estimation Model for Eye Drawing. In *Extended Abstracts on Human Factors in Computing Systems* (CHI'06). ACM, New York, USA, 1559–1564.