

Augmented Interactions

Conor Hennessy & Erick Manuel, University of New Brunswick

Introduction

Drawing applications conventionally allow for drawing with mouse interaction and selecting tools from menus/toolbars. In this investigation, we implement and explore the use of two alternative ways to interact with a drawing application; gestures by mouse and keyboard shortcuts. With this, we implemented one way to use mouse gestures and the other way to use shortcut keys. With this we discussed and formed different gestures, to then select one for each of the stated drawing options and settings.

The significance of this study is that it explores the potential gestures and shortcuts which allow for this alternative way of using a drawing application. This has a number of applications including, making drawing applications more efficient or accessible for certain user groups. With this, we designed a range of input schemes and evaluated the effectiveness of these schemes throughout this report. We will gain an understanding of the criteria which makes a gesture effective and how each gesture/shortcut compares to one another. This understanding will be gained by the informal results of giving our implementation to another group and our own experience of using the system.

Input Scheme Designs

The set gestures and shortcut keys both allow for the selection of settings within the drawing app. They allow the user to determine the shape, stroke weight and colour settings for the drawing. For all drawing operations, there are two corresponding actions; one being a keyboard shortcut and one being the mouse gesture to change to that respective setting.

The keyboard shortcuts can be used by pressing the corresponding button at any time in between drawing any shapes. The setting would then be changed and the next drawing would have this change in place.

The gestures can be made at any time in-between drawings by right-clicking and making the corresponding gesture shape. As the gesture is made the trail of the gesture is shown to give feedback to the user. Upon recognition of a gesture, the appropriate setting would be changed ready for the next drawing.

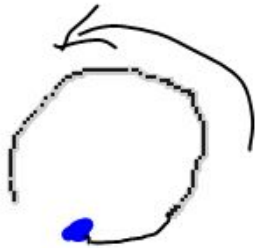

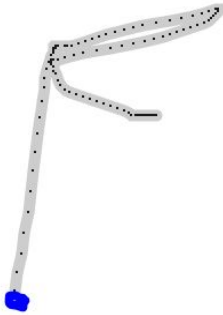

The cheatsheet on the next page and the associated figures show the specific keyboard buttons or gesture the user can make to change the settings.


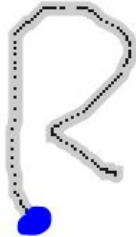

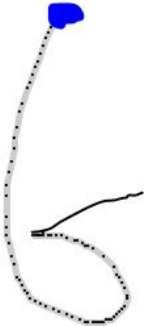
Cheatsheet

The following cheat sheet can be used to show a user what the selection schemes are, in order to introduce them to the drawing operations easily.

When sharing the application this cheat sheet and instructions on how to run the system should be included. Which were found in a file titled "READ ME - how to run.txt".

The blue dot shows the starting point of each gesture.

Shape	Keyboard Shortcut	Gesture
Rectangle	r	 <p>Anti-clockwise circle</p>
Oval	o	 <p>Clockwise circle</p>
Freeform Line	f	
Straight Line	l	

Colour	Keyboard Shortcut	Gesture
Black	SHIFT + 1 [!]	
Red	SHIFT + 2 [@]	
Green	SHIFT + 3 [#]	
Blue	SHIFT + 4 [\$]	

Assignment #2

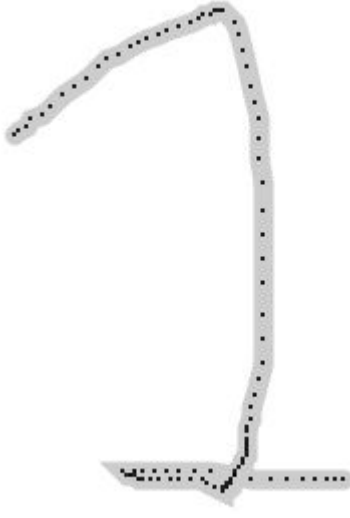


Faculty of Computer Science, UNB

CS4065

2020-02-01

Conor Hennessy [3681080]

Erick Manuel [3680151]

<i>Weight</i>	<i>Keyboard Shortcut</i>	<i>Gesture</i>
Thin	1	 A thin, dotted handwritten digit '1' with a slight curve at the top and a horizontal base.
Medium	2	 A medium-weight, dotted handwritten digit '2' with a curved top and a horizontal base.
Thick	3	 A thick, dotted handwritten digit '3' with a curved top and a horizontal base.

Design explanation

When it came to forming this assignments program we came together as a pair to decide what the selection scheme we would form. We opted to ensure that all drawing options would have both a keyboard shortcut and a unique gesture. We decided to do this to allow for a real user to have the choice between using the keyboard or making a gesture to change their option. With our own discussion however, we did not think that having a unique gesture for every option may prove difficult. As to have 11 different gestures which do not clash with one another on detection would be difficult.

So with the scheme we designed and outlined in the cheatsheet on previous pages we believe that the gestures are all intuitive for each of the drawing settings. As for the stroke gestures, the shapes correspond to arbitrary values for thin, medium or thick strokes; For the colours, the gestures are all first initials of the respective colours; Finally, for the shapes, the gestures are all relative to the shape that they make themselves. Overall the gestures are simple, easy to remember and can be said to be natural shapes or patterns which correspond to their function relatively easily.

It is also a benefit to the design that all the gestures are continuous within each group of settings (for shape, colour and stroke) as option follows one another. For example. the stroke settings are a set of numbers, where a larger number means thicker and a smaller number simply means it is a thinner stroke setting. This is intuitive for both the keyboard and gesture as the keyboard shortcut where the keyboard number corresponds to this setting and simply making the number as a gesture corresponds to this setting too.

With the design of the gestures, we decided (after some preliminary testing) that the gesture for a circle would go clockwise while the gesture of a rectangle would go anti-clockwise. This was as a result of some testing where the gesture library was sometimes incorrectly mistaking the gesture for one another. Adding this feature reduced the number of false positives greatly on gesture recognition.

We designed the keyboard shortcuts with simplicity and ease of learning in mind. For instance, the keys for the shape options are the corresponding initial of each shape. This is simple to use and easy to remember. Then the stroke weight keys for thin, medium or thick are mapped to the 1, 2 or 3 number keys respectively. We believed for this to be intuitive as a larger number means thicker and allows for the user to associate the thickness of shapes with an arbitrary unit of thickness.

Then the colour options (black, red, green, blue) are accessed by pressing *shift* than a number of 1 through to four. We decided to use numbers to represent the colours for two main reasons; to have all the colour keys next to each other and to reduce repetition/ambiguity. As if we did use the first initial of the colours we would have had an issue as two colours share the same first letter (blue & black) while another colour (red) shares the same first letter with a shape (rectangles) and so if we did use letters

to represent colours there would have been the potential of some colour letters being ambiguous from the colour they actually represent. For instance, red could not be *R* as *R* already corresponds to a rectangle.

The use of *shifts* was a design decision we took to allow for the numbers to have a second use. This is also an intuitive shortcut method for keyboard shortcuts and so we decided to use this so the selection would feel natural to a user.

We decided it would be better to have the colour options using the shift number keys rather than the shape options as we deemed it more natural to have the colours corresponding to numbers rather than shapes. As the numbers correspond to the RGB values which make up the colours.

Evaluation Results

Speed

The shortcuts key are faster than the gesture as our scheme for gesture got a lot of false positive leading to a lot of redrawing before getting the gesture right but, the shortcuts key use at most 2 keys at the same time and most of them are one key shortcut making it faster to get it.

Effort

In our scheme the gestures impose a great challenge to the user as we did a unique gesture for every choice that the program has in consequence our gesture has a lot of false-positive case when doing the gesture to the point of getting some gesture right on a fluke, especially in the freeform line gesture and the black colour gesture. Nevertheless, as our program also has shortcut keys for every choice it helps dealing with the usability of the program leaving leeway for changing the configuration.

Differentiability

In our program, the settings can be done by gestures or shortcut keys or a combination of both, so as long as they remember how to do the gesture or the shortcut they will be able to attain any configuration.

Memorability

Yes, as the shortcuts are related to the figure start letter and “l” for the straight line and the colour and weight of the line are related to numbers that are close going from 1 to 4 on colours and 1 to 3 on weight.

Naturalness and satisfaction

The naturalness of the program goes down as the gestures have a lot of false positive cases leading to abnormal flow while using the gesture to make the

Assignment #2

CS4065

2020-02-01

Faculty of Computer Science, UNB

Conor Hennessy [3681080]

Erick Manuel [3680151]

changes. Furthermore, this leads to frustration to the user causing it to be a not so pleasant experience while using the gestures.

Conclusion

In conclusion, the augmented input schemes can lead to better performance of the user if they are properly selected for the things they will be used. However, having our gesture scheme as an example trying to put a lot of gestures for doing a little fragment of the work will lead to a lot of false positive cases leading to the gesture becoming a problem instead of a solution. The shapes used for the gesture are also important as they would lead to better usability of the program. A poor selection of gestures would be difficult to associate with the action that the user wants to do. An example that can be seen in our gestures is the freeform line gesture, as aside from being a challenging gesture to do, the user would not associate initially the letter “F” with a freeform line.