# Multitouch Radial Menu Integrating Command Selection and Control of Arguments with up to 4 Degrees of Freedom

Shrey Gupta
École de technologie supérieure
Montreal, Canada
shrey.gupta.1@ens.etsmtl.ca

Michael J. McGuffin
École de technologie supérieure
Montreal, Canada
michael.mcguffin@etsmtl.ca

## ABSTRACT

We design and evaluate a multitouch radial menu for large screens with two desirable properties. First, it allows a single gesture to select a command and then continuously control arguments for that command with unbroken kinesthetic tension. Second, arguments are controlled with 1 or 2 fingers for up to 4 degrees of freedom (DoF). For example, the user may select one command for 4 DoF direct manipulation (translation + scaling + rotation), or another command for 3 DoF camera operations (pan + zoom), using the same two-finger pinch gesture, but with different initial orientations of the gesture to disambiguate. We present a taxonomy to classify previous menuing techniques sharing the first property, and discuss how very few techniques have both of these properties. Our work also extends previous work by Banovic et al. in the following ways: our menu supports submenus and a fast default command, and we experimentally evaluate the effect of varying the number of rings in the menu, the symmetry of the menu, and the use of one hand vs. two hands vs. a stylus and hand.

## CCS Concepts

•**Human-centered computing** → **Gestural input;**

## Keywords

multitouch; popup menu; direct manipulation; pen; stylus

## 1. INTRODUCTION

Popup menus for invoking commands have several advantages over toolbars, pulldown menus, or tool palettes: they require no screen space when not in use; they eliminate back-and-forth motion between the main work area and widgets located in the periphery (which is important on large display surfaces); and they can be designed to only remain open if a finger, button, keyboard key, or stylus tip is held down, creating kinesthetic feedback. The resulting *kinesthetically-held* mode (also called spring-loaded mode or quasimode [26]) can help a user remember which command mode they are in, reducing rates of mode errors [27]. A subset of menuing techniques (e.g., [25, 12]) also allow the user to specify an object, command, *and arguments*, all in a single drag. For example, a user might press down on an object to be modified, causing the menu to pop open; then the user drags through the menu to select a command; and then somehow transitions to dragging out the arguments for that command. In a 2D drawing application, a "Move" command would have a 2 DoF (degrees of freedom) argument, whereas a "Rotate" command would have a 1 DoF argument.

Unfortunately, most previous menuing techniques have a limited compatibility with two-finger pinch gestures in multitouch interfaces. For example, with MultiTouch Menu [1], HandyWidgets [29], SPad [8], and Pin-and-Cross [22], the pinch gesture "coexists" with the menu widget and is invoked separately, outside the menu. Such a singular pinch-controlled command can be invoked as a special case whenever two fingers touch the screen, however a problem arises if more than one command requires pinch-controlled arguments with 3 or 4 DoF. In a 2D application, the user may sometimes want to invoke 4 DoF *direct manipulation* (translation + scaling + rotation), and other times invoke 3 DoF *camera* operations (pan + zoom), or even change RGB or RGBA colors using the same two-finger pinch gesture. Frisch et al. [9] found that users suggested the same (ambiguous) pinch gesture for "Scale size of node" and "Zoom whole diagram" commands. Similarly, in a 3D application, there could be multiple commands for camera and object transformations, each with more than 2 DoF, and each controlled with two fingers. How can such commands be disambiguated? One approach would be to use a toolbar or menu to enter a mode (e.g., [13]), but this lacks the advantages listed earlier of popup menus and unbroken kinesthetic feedback. Another approach would be to invoke direct manipulation whenever the user presses their fingers down on an object, and invoke camera operations when the fingers press down on empty space. However, this only allows for *two* pinch commands, and if the user zooms in on a single object that fills the viewport, then empty space is no longer accessible to invoke camera operations.

Instead, we propose disambiguating the command by making use of (1) the initial orientation of the two fingers when they both press down, and also (2) the order in which the fingers press down and (3) the distance between the fingers. The command set is displayed as a multi-ring pie menu, similar to Banovic et al.'s 3-ring pie menu [3]. We extend Banovic et al.'s work by (1) allowing the number of



*AVI 2016 June 7–10, 2016, Bari, Italy*



DOI: http://dx.doi.org/10.1145/2909132.2909266

rings to be varied, (2) allowing the use of both hands and therefore all directions (sectors) in the menu, (3) supporting submenus and a quickly-accessible default command in the center of the menu, (4) allowing command arguments to be provided immediately after the fingers touch *and before they release*, to maintain kinesthetic tension. We also experimentally compared variants of our menu, including versions with a single ring and/or a symmetrical layout, allowing the user to make faster selections but at the cost of having fewer commands accessible in the top level of the menu.

## 2. BACKGROUND

The previous work on menus can be divided into four subsets. The first contains menus that require multiple clicks or actions to select a command, such as bezel swipes on a tablet, or right-click menus where the user click-releases to open the menu and then click-releases to select a command.

The second subset, which has the potential to be faster, contains menus where a command (and possibly an object) are selected with a single press-release or press-drag-release gesture involving one or more fingers (with contact maintained by at least one finger throughout the gesture). These include traditional toolbars, as well as Marking menus [18], FaST Sliders [23], multitouch marking menus [21], Banovic et al.'s 3-ring menu [3], Kin et al.'s two-handed marking menus [17], HandyWidgets [29], Arpège [10], and FastTap [13]. The selected command in these menus may switch the user into a command mode, afterwhich a second press-drag-release can be used to provide arguments for the command. However, with such a design, the user may be sometimes be confused about the mode they are in [26], because they may lose track or be distracted in between gestures. Such mode errors have been shown to be reduced when kinesthetic tension is maintained [27]. On the positive side, most of the menus within this second subset are popup menus, and therefore eliminate back-and-forth movements and do not appear unless needed. Many of them are also designed to be operated with gestures that can be performed with little or no visual attention once the gestures have been learned. To make it easier for novice users to learn these gestures, the menus provide visual feedback to guide their use, easing the transition from novice to expert use. The gestures are thus *self-revealing* and ideally lead to *eyes-free* operation.

The third subset builds on the second subset with menus that allow command, *arguments*, and possibly an object to all be selected with a single gesture (using one or more fingers) with uninterrupted kinesthetic tension. These menus are classified and compared in the taxonomy in Figure 1. This group includes Control Menu [25], FlowMenu [12], Scriboli's menu [14], Springboard [15], Tracking menu [6], Hover widgets [11], PieCursor [7], MultiTouch Menu [1], SPad [8], and Pin-and-Cross [22]. Also in this group is the hotbox as described in [24] which extends [20] to enable the specification of arguments with a command in a single drag. We also include the "bimanual marking menu" as described by Chen et al. [5], where the non-dominant hand performs a marking menu gesture to select a command while the dominant hand controls the argument to provide with the command.

The remaining menus in Figure 1 make up the fourth and final subset of menus, which again builds on the properties of the previous subset. These menus allow for object, command, and *4 DoF* arguments to be specified without breaking kinesthetic tension. They are the Toolglass, FingerCount menu, and the Multitouch Radial Menu proposed in the current work (last column of Figure 1). We discuss these menus more at the end of this section.

Returning to Figure 1, the form of the taxonomy was settled on after several iterations of organizing previous work. Columns are grouped according to the hardware platform necessary for each technique. Thus, a designer might look-up the available techniques for a given target hardware platform. Rows correspond to criteria by which techniques could be evaluated or chosen. These criteria were carefully chosen to be as objective as possible, while covering a range of design issues. All of these criteria are phrased in a *positive* way, so that satisfying more criteria is desirable or at least neutral. For example, the 2nd last row is labelled "Hardware detection of hover is optional"; techniques that satisfy this criteria don't *require* hover detection and therefore work on a wider variety of hardware platforms. Cells containing the word "Yes" are colored green to show that a criterion is satisfied. Some cells only partially satisfy a criterion; these are partially shaded green. Notice that no single technique satisfies *all* criteria: there are many tradeoffs.

Some of the previously published techniques could be modified to change their properties. For example, Tracking menus could be enhanced with a button for the non-dominant hand allowing the user to "escape" from the menu and access a default command, or SPad could tradeoff eyes-free selection of commands to allow for more menu items. Also, Bimanual marking menus [5], MultiTouch menu [1], SPad [8], and Pin-and-Cross [22] could all be modified to allow for *multiple* commands with *4 DoF* arguments, but this is not how they were presented in previous work. For simplicity, we populated Figure 1 with the menu techniques as they were presented in the literature.

[[Some cells in Figure 1 are numbered for detailed notes, provided here. Readers may skip these. **Note 1:** a typical way to invoke a Control Menu or FlowMenu is over an object, in which case the selected command operates on that object. This requires first pointing at that object, incurring some temporal cost. The same press event over empty space could be used in two ways: first, as a faster way to open the menu (possibly to select a command with global scope, or to apply a command to a previously selected object), or second, as a way to invoke some default command (such as lasso selection, inking, object creation, or camera operations) without going through the menu. It is not possible to allow for *both* of these with Control Menu nor FlowMenu without using additional hardware buttons. However, many other menuing techniques *do* allow for both, such as the Scriboli menu, which can be opened over empty space with a pigtail gesture, and which also allows for a default command by simply pressing down on empty space and dragging. **Note 2:** Scriboli is exceptional in that *two* default commands can be executed without the menu simply by dragging, and by terminating either with a pigtail gesture (for lasso) or not (for ink). **Note 3:** eyes-free targeting can be performed if the target is in the center of the hotbox, or in one of the 4 cardinal directions, enabling a ballistic motion before clicking. **Note 4:** SPad, as presented in [8], is limited to 4 submenus × 3 items/submenu = 12 commands in total.]]

In general, Figure 1 could be used by designers to choose an appropriate popup menu technique based on their hardware constraints (e.g., is hover detected on the target platform?) and/or based on other criteria (the rows).

| | | 1-point input (no hover required) | | | 1-point input with hover | | | | 2-point input with hover | multipoint input (no hover required) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Button for non-dominant hand | | | | Button for non-dominant hand | | | | | | | |
| | | Control Menu, FlowMenu [Pook 2000, Guimbretière 2000] | Scriboli menu [Hinckley 2005] | Springboard [Hinckley 2006] | Tracking menu [Fitzmaurice 2003] | Hover widgets [Grossman 2006] | PieCursor [Fitzmaurice 2008] | Hotbox [Kurtenbach 1999, McGuffin 2009] | Toolglass [Kurtenbach 1997] | Bimanual marking menus [Chen 2008] | MultiTouch menu [Bailly 2008] | Finger-Count [Bailly 2012] | SPad [Foucault 2014] | Pin-and-Cross [Luo 2015] | Multitouch radial menu |
| Speed | Menu always accessible nearby | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| | Menu can be opened without first selecting object | Only one or the other[1] | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| | Default command is possible | | Yes[2] | Yes | No | Yes | No | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| | Eyes-free selection of commands | Yes | Yes | No | Only for commands on the edge of the menu | Yes | Yes | Only for certain commands[3] | No | Yes | Yes | Yes | Yes | Yes | Only in the root menu |
| | Menu can stay open for invoking multiple commands | No | No | Yes | Yes | Only submenus appearing after the initial click | Yes | Yes | Yes | No | Yes | Yes | Yes | Yes | Only in one submenu at a time |
| Scalability | Submenus | Yes | Yes | Yes | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes | Yes[4] | No | Yes |
| | Unconstrained 2D menu layout | No | No | Yes | Yes | Only submenus appearing after the initial click | No | Yes | Yes | No | No | Yes | No | No | Only in submenus |
| Space management | Only appears when needed | Yes | Yes | No | No | Yes | No | Yes | No | Yes | Yes | Yes | Yes | Yes | Yes |
| Hardware | Hardware detection of hover is optional | Yes | Yes | Yes | No | No | No | No | No | Yes | Yes | Yes | Yes | Yes | Yes |
| Expressivity | Max degrees of freedom of command arguments | 2D | 2D | 2D | 2D | 2D | 2D | 2D | 4D | 2D | 2D | 4D | 2D | 2D | 4D |

**Figure 1: Menuing techniques (columns) compared according to multiple criteria (rows). Green cells show desirable properties, and partially fulfilled criteria are shaded. See main text for Notes 1 - 4.**

Returning now to the few menus with 4 DoF in the last row of the taxonomy, we first have Toolglass [4, 19]. Toolglass is a two-point bimanual technique, usually implemented with two mice or similar devices, that allows for simultaneous selection of object and command by clicking *through* a command onto an object or location. This has been shown to be either faster [16] or slower [5] than other menus, depending on conditions. We found that there were problems with adapting the Toolglass to a multitouch surface (more on this in section 6), motivating the current work.

Second is Finger-Count [2], which involves placing $N$ fingers of the non-dominant hand (NDH) followed by $M$ fingers of the dominant hand (DH) to select the $M$th item of the $N$th submenu. This is easily understood by novice users, and integrates nicely with a single 4 DoF command like direct manipulation if this command is the 1st item of the 1st submenu. For other commands, however, more than 2 fingers must be placed, which limits precise positioning. The menu also *requires* the use of both hands, whereas users sometimes wish to be more relaxed and use a single hand.

Our proposed menu enables a unique tradeoff in design options. Unlike Finger-Count menu, ours can optionally be used with one hand to access the most frequent commands in the top level, and does not require more than two fingers to control arguments. And unlike the other multitouch menus in the taxonomy, we support submenus with unconstrained layout of items, and explicitly designed for multiple 4 DoF commands. The next section presents our design.

## 3. MULTITOUCH RADIAL MENU

In designing our menu, we sought a way to enable the user to, first, select a command, and subsequently, provide an argument for that command using 1 or 2 fingers (for up to 4 DoF), all with a single drag (i.e., with unbroken kinesthetic feedback). The *transition* from the first stage (command selection) to the second (argument control) could be triggered by various means, such as a drag distance threshold (as in Control Menus [25]), or a pigtail gesture (as in Scriboli [14]), or a timeout, or a release-and-repress event, however all of these incur some temporal cost, and the last one breaks the kinesthetic feedback. We realized that if the user is planning to use two fingers in the second stage anyway, as soon as they touch both fingers on the screen, and just before they start to drag, the *relative positions* of the fingers already provides some information to the system that could be used to select the command and complete the first stage. If the user can learn to plan the initial positioning of their fingers to properly select the desired command, this could theoretically be performed very quickly (as fingers could be prepared "on their way" to the screen), with the transition to the second stage occurring "for free" as the user starts to drag. Taking inspiration from Marking Menus, the easiest relative positions to learn and remember are probably based on angles: two finger tips can lie on a horizontal, vertical, or diagonal line. In addition to this, the order in which the two fingers arrive on the screen, and the distance between them, could also disambiguate commands. (This distance between fingers will be used to select among different rings of a pie menu, whereas orientation and touch order are used to select a sector of the menu.)

We also sought to preserve the ability to have one (2 DoF) default command accessible with a single finger, and to make this command as easy as possible to get to, since it is probably the most frequently used. Depending on the application, this default command might be lasso selection, or inking, or object creation, or camera pan / 2D scroll. We realized that this would be easy to allow: when the system sees a single finger press, it waits to see if a second finger will touch (causing some non-default command to be selected) or if, instead, the first finger starts to drag (causing the default command to execute).

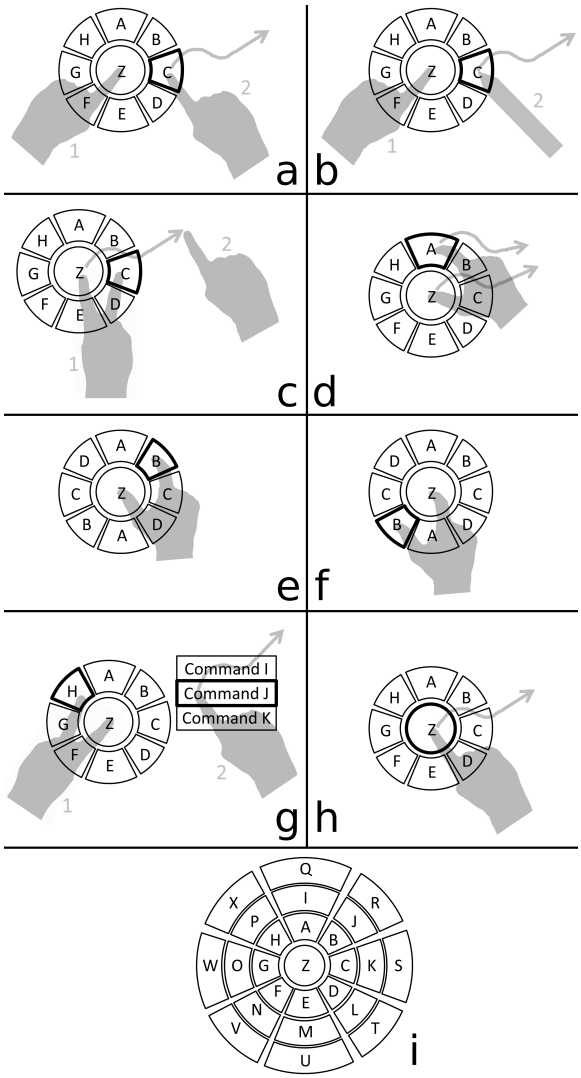


**Figure 2: Using the multitouch radial menu.**

These ideas led to the variants illustrated in Figure 2. Figure 2a shows one way of selecting a (non-default) command: the left hand presses to open the menu, and then the right hand presses on item C to select it and drag to provide a 2 DoF argument with it. Item C could be, for example, a command to translate a previously selected shape in 2D. Note that as soon as the right hand begins to drag, it is no longer necessary for the left hand to hold the menu open, since the right hand is providing kinesthetic tension.

Figure 2b shows that this could also be done with a hand and a stylus. If the hardware can distinguish between finger and stylus, this opens the possibility to having two different menus: one opened when a finger first presses, and another opened when a stylus first presses. We evaluated this possibility in our experiment, discussed later.

Figure 2c shows how to operate the same command with a single hand. First, the index finger opens the menu, then the middle finger selects the command C, then the user is free to use either finger to drag out the 2 DoF argument for the command, *lifting up the other finger in the process*.

Figure 2d shows the use of a single hand to select a command and control a 3 or 4 DoF argument. The thumb first opens the menu, then the index finger selects command A, then both fingers drag out the argument. Command A could be, for example, a 2D camera control, allowing simultaneous pan and zoom. Alternatively, in a 3D application, command A could be for translation up/down, left/right, forward/backward (controlled by pinch distance), as well as camera roll (controlled by pinch angle), for a total of 4 DoF.

A rotation of the hand enables the same kind of interaction for command B (Figure 2e). In Figures 2e and f, the menu is now **symmetrical**: the items A, B, C, and D are repeated. This allows the user to select any of these 4 commands without paying attention to the order in which their two fingers touch the screen. For example, command B is selected whether the thumb touches first (Figure 2e) or the index does (2f).

Some items may open up submenus. Figure 2g shows item H doing this: the submenu is held open with the left hand, allowing the right hand to select one or more commands in the submenu. In the figure, the right hand selects command J and then drags out a 1 or 2 DoF argument. For example, command J might control the opacity or color component of a selected object. This could be followed by multiple other command invocations within the submenu, all operating on the same previously selected object, while the left hand continues to hold the submenu open. This idea for a submenu was inspired by the way the hotbox [20, 24] works, where the NDH holds down a physical keyboard key to keep a virtual menu open and accessible to the DH's pointing device. Like the hotbox, this kind of submenu in our multitouch radial menu increases the scalability of our technique, especially since the submenu is not constrained to a radial layout and may contain several rows and columns of commands.

Figure 2h shows that the center of our menu contains a default command Z that the user can invoke very easily by simply touching and dragging with a single finger. Note that this requires the menu to have a spatial threshold defined: if the user presses down with 1 finger and moves less than this threshold, the menu awaits a 2nd finger, but if the 1st finger moves more than this threshold, the center command is invoked. In a 2D application, a typical default command Z might be to pan or to draw ink; and in a 3D application, a plausible default command Z might be to "orbit" (rotate) the 3D camera (these examples all require 2 DoF each).

Figure 2i shows a 3-ring variant of the menu. Having more rings allows for more commands in the root level of the menu, but may slow down the user by requiring more precise positioning of the fingers.

We thus have three ways to increase the breadth of the top level of our menu: by using asymmetry rather than symmetry, by using more rings, or by using a stylus that opens up a menu different from the one opened by the finger. These

will all presumably slow down the user, thus an experimental comparison is needed to reveal which of these incurs the smallest temporal cost.

# 4. EXPERIMENTAL EVALUATION

Banovic et al. [3] proposed an earlier 3-ring menu for use with a single hand, and experimentally compared variants of their own menu against each other. However, their design did not consider controlling arguments with unbroken kinesthetic tension, nor how to integrate a default command, nor submenus. The experimental evaluation of their menu also did not test all 8 directions, nor did it vary the number of rings, use of symmetry, use of a stylus or 2nd hand. We therefore performed our own experimental comparison of menu variants. Another difference between our experiment and Banovic et al.'s is that we did *not* have users wear cots on their fingertips, and we allowed users to use whichever fingers they wanted to, making our experiment less controlled but more realistic, at the risk of suffering higher error rates due to inadvertent touch events. The goals of our experiment are to (1) determine which factor for increasing menu breadth incurs the smallest cost in time, be it number of rings, asymmetry, or stylus; (2) determine which sector or direction within the menu is slowest or most error prone, and therefore best suited for use as a submenu rather than as a frequently-accessed top-level command; and (3) check if the error rates of the user are within reasonable bounds.

We used a Wacom *Cintiq 24HD touch* display (hereafter simply "Cintiq") with 1920×1200 pixels, 94 dpi, supporting 10 finger multitouch and simultaneous stylus input. Source code was written in Java (using usb4java/libusb to read data from the Cintiq) and ran on a Dell Precision M4700 laptop running Linux. The user was allowed to incline and adjust the level of the Cintiq for comfort.

The task was to select a target menu item as fast as possible from a multitouch radial menu. Users did not provide any subsequent arguments for the command in the form of a drag. We also did not use any submenus nor any central default menu item in the experiment. For each trial, the system first displayed a scaled-down duplicate of the menu at the top of the screen with the target item highlighted in orange. A square selection area in the center of the screen was also highlighted in orange. The user then had to place their finger(s) (and possibly stylus) in octagon-shaped starting positions (60 pixel radius) (Figure 3), and hold them there on the display for a randomly chosen foreperiod lasting between 500 and 900 ms — this prevented the user from anticipating when the trial would begin. Then the highlighting color changed to green, signaling the start of the trial, afterwhich the user had to move their finger(s) (and possibly stylus) into the square selection area to invoke the menu and select the target item as quickly and as accurately as possible. Because users had to move from the starting positions to the square selection area, the distance to travel with the hands was always the same. Every target item required two contacts: one finger or stylus to open the menu, followed by one finger or stylus to select the menu item.

If the user lifted a finger or stylus before the end of the foreperiod, the foreperiod had to be restarted. Users were instructed to keep fingers/stylus inside the square selection area to complete the trial. If a finger/stylus touched outside, the input was ignored.

The time taken to complete each trial was measured from

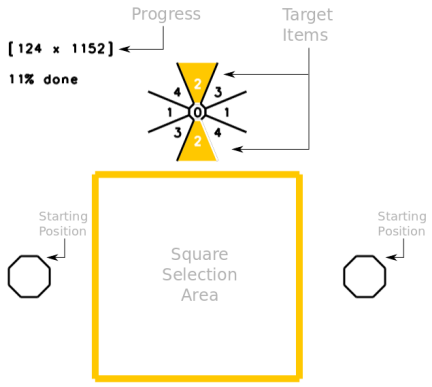


**Figure 3: The display, before the start of a trial.**

the start of the green highlighting to the time that the participant had put their finger down on a menu item. At the end of the trial, the user was given visual feedback in green if the trial had been successful, and red if there was an error (incorrect menu item selected). In the case of such an error, the user had to redo the trial.

A pilot study found that users sometimes lifted their first finger before their 2nd finger landed on the target menu item, causing an error. Thus, in the full experiment, we added a 200ms timeout in the code to allow for the user to do this without causing an error.

In our full experiment, 16 users (5 female, 2 left-handed) aged 19 to 38 (mean 26.8, sd = 5.69 ) participated. 14 used touch screens daily (on phones), 10 had prior experience with a stylus, 3 had prior experience with a radial menu. Each user session lasted approximately 1.5 hours, and users were given a $20 gift card.

Each user's hand size was measured by having them stretch their hand and touch the Cintiq with their thumb tip and tip of middle finger. The measured distance was on average 625 pixels = 6.6 inches (sd = 52 pixels, min = 518, max = 715). The menu was scaled to this handSize, such that the central region of the menu was centerRadius = 30 pixels in radius, and each ring of the menu had a radial thickness of (handSize - centerRadius) / numberOfRings, where numberOfRings varied from 1 to 3 depending on the current experimental conditions.

The experiment varied the following factors: **SYMMETRY**, which was either Sym (symmetrical) or Asym (asymmetrical); **HANDS**, which could be 1HAND (one hand), 2HAND (two hands), HAND+STYLUS (one hand opens the menu with a finger, the other hand completes with the stylus), or STYLUS+HAND (one hand opens with the stylus, the other hand completes with a finger); and **RINGS**, which could be 1RING, 2RING, 3RING. Each menu instance could have anywhere between 4 items (Sym, 1RING) and 24 items (Asym, 3RING), and for each menu instance, the user performed 48 trials, covering each item an equal number of times. The three factors were fully crossed, yielding 16 participants × 2 levels of SYMMETRY × 4 levels of HANDS × 3 levels of RINGS × 48 trials = 18432 trials in total. Because we were mostly interested in the effects of SYMMETRY and HANDS, the ordering of their 8 combinations of levels was counterbalanced with an 8×8 Latin square. The ordering of RINGS was fixed and increasing in number of rings.

## 4.1 Results and Discussion

The main effect of the time was analyzed using non parametric repeated measures ANOVA using the Aligned Rank Transform [28]. All post-hoc comparisons were made using a pairwise $t$-test with Holm's sequential Bonferroni correction. Error count measures were compared using the $\chi^2$ test.

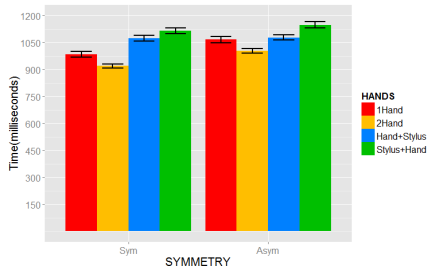


**Figure 4: Time by SYMMETRY and HANDS, with 95% confidence intervals.**

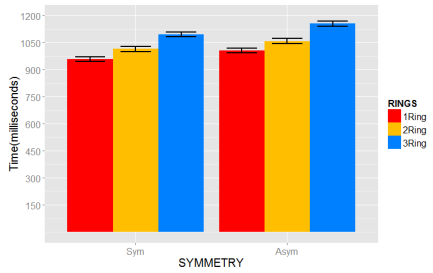


**Figure 5: Time by SYMMETRY and RINGS, with 95% confidence intervals.**

Each of the main factors had a significant effect on time (Figures 4 and 5). SYMMETRY had a significant effect on time ($p < 0.005$, $F_{1,15} = 12.45$) with symmetrical menu (mean time 1023ms) significantly faster than asymmetrical menu (1074ms). RINGS also had a significant effect on time ($p < 0.001$, $F_{2,30} = 71.6$ ). The 1RING menu (982ms) was faster than the 2RING menu (1037ms) which was faster than the 3RING menu (1126ms) (all differences significant). HAND also had a significant effect on time ($p < 0.001$, $F_{3,45} = 19.1$). 2HAND (962ms) was faster than 1HAND (1026ms), which was faster than HAND+STYLUS (1075ms), which was faster than STYLUS+HAND (1132ms) (all differences significant). We see in these results a classic tradeoff: increasing menu breadth slows down the user. From a design perspective, the number of items in the menu may be doubled by (1) using asymmetry rather than symmetry (corresponding to a cost of 1074ms - 1023ms = 51ms), or (2) using 2 rings rather than 1 (costing 1037ms - 982ms = 55ms), or (3) using a stylus (costing anywhere between 1075ms - 1026ms = 49ms to 1132ms - 962ms = 170ms). By this comparison, it seems clear that using a stylus is *not* the best way to increase menu breadth. Furthermore, without a stylus, the user is free to choose between using 1HAND or 2HAND (the latter being 64ms faster, on average, at the cost of requiring more effort). Thus, using asymmetry or more rings are better ways to increase breadth.

It is interesting that STYLUS+HAND was significantly slower than HAND+STYLUS. This is explained by two observations. First, users found the stylus slippery against the Cintiq surface, forcing them to slow down so as to stay within the menu center when opening the menu with the stylus. Second, the position at which the menu was opened did not much matter, whereas the selection *within* the menu required more precise pointing and is thus best done with the dominant hand (i.e., the hand holding the stylus).

We suspect 1HAND was slower than 2HAND because 1HAND required rotating the wrist to sometimes less comfortable angles.

We also analyzed data according to a SECTOR factor, which varied from target to target, with 8 levels (N, NE, E, SE, S, SW, W, NW). SECTOR had a significant effect on time ($p < 0.001$, $F_{7,105} = 12.0$), with SE and NW significantly slower than the other directions. This is likely a consequence of most users being right-handed, and having to tuck their thumb under their hand to reach SE and NW targets in the 1HAND condition.

From a design perspective, if all but one direction is occupied with frequently-accessed commands, the best direction for a submenu is SE-NW, since the left hand can hold open the submenu while the right hand accesses submenu items, similar to a Hotbox. (These are of course reversed for left-handed users.)

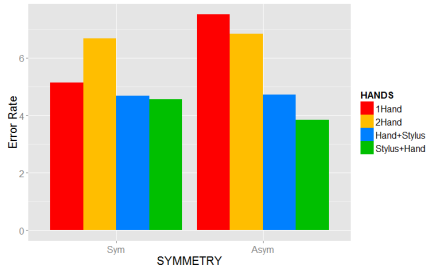


**Figure 6: Error rates (in %) by SYMMETRY and HANDS.**

RINGS had a significant effect on error ($p < 0.05$), with 1RING having the lowest error rate, followed by 2RING followed by 3RING which was worst (all differences significant). The symmetrical menu had a lower error rate (5.28%) than the asymmetrical menu (5.75%), however this difference was not significant. Notice that all mean error rates are below 10%, despite our not asking users to wear cots or control which fingers they used. However, (Asym,1HAND) and (Asym,2HAND) have the highest measured error rate of the 8 conditions in Figure 6, suggesting that these combinations should be avoided in favor of symmetrical layouts.

Users were asked to score the conditions on Likert scales. (Sym,1HAND) was rated as the easiest and the least tiring, whereas (Asym,STYLUS+HAND) was rated as the most difficult and most tiring. (Sym,1HAND) was also the most preferred technique, followed by (Sym,2HAND).

Given the above results for time, error rates, and the subjective preferences, we recommend the following. If more than 4 commands are desired in the top-level menu, to increase the top-level menu breadth, we recommend adding a 2nd ring rather than using asymmetry or requiring a stylus. If the user will be using a stylus anyway, e.g. as part of a drawing application, it should be possible to open the

menu with a finger (e.g. on the non-dominant hand). For right-handed users, frequently used commands should not be placed in the NW-SE directions, although these directions are useful for a submenu held open by the left hand.

At the end of each session, participants in the experiment were shown a simple drawing program (described in the next section) that uses the Multitouch Radial Menu, and asked to interact with the program. 10 out of 16 participants stated they would like to use the menu in real-world applications, such as in 3D applications or drawing programs.

# 5. EXAMPLE APPLICATION

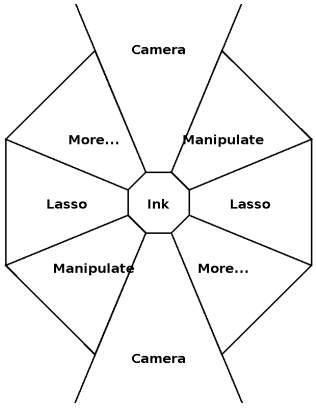


**Figure 7: An implementation of a symmetrical multitouch radial menu for a drawing application.**

Figure 7 shows a screenshot of a multitouch radial menu implemented in our drawing program (see companion video for demonstration). Notice that the menu is symmetrical, like in Figure 2e. Consider a hypothetical right-handed user. Once this user has learned that, for example, "Manipulate" lies on a North East – South West axis, they can place their thumb and index finger on the screen, with fingertips aligned on a diagonal line, and it won't matter which finger touches down first: if the thumb touches first, the menu will be centered on the thumb, and the index finger will touch the North East "Manipulate". Alternatively, if the index finger touches first, the menu will be centered on the index finger, and the thumb will land on the South West "Manipulate".

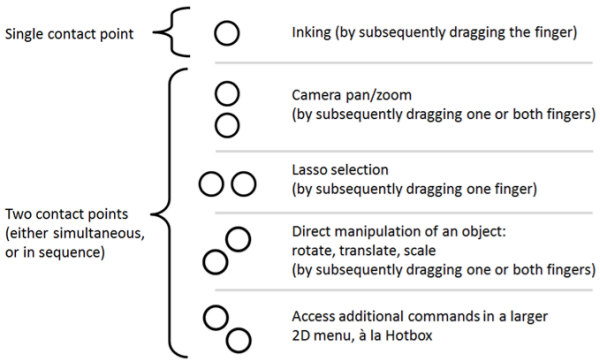


**Figure 8: An alternative way of explaining the gestures available in the menu of Figure 7.**

This symmetrical layout also enables an alternative way to explain or present the gestures available to a new user. Figure 8 was designed based on the menu in Figure 7, and may be a more effective way of explaining the available gestures to a user. One finger maps to the default command, and two fingers map to other commands depending on the orientation of these two fingers. Note that, at present, Figure 8 is only a static image, and we have not yet found an effective way to incorporate similar visual feedback into an interactively explorable popup menu.

Observe also that the North/South "Camera" command in Figure 7 has pie slices that extend outward more than the other slices. The Camera command in this case is for controlling 2D zoom and pan. We observed that when users wish to zoom *out*, they naturally begin with their fingers far apart to prepare for pinching their fingers together. This requires that the corresponding pie slices be long enough to "catch" the fingers. Another way to address this issue would have been to make all pie slices extend infinitely outward (though not necessary drawing them as infinitely large), ensuring that finger events are always "caught" by the menu; we chose not to do this to instead allow the default command to be invoked on multiple fingers when they press *outside* the menu. In our application, this allows for inking with multiple fingers: the user first uses their NDH's index finger to open the menu, then presses down with their DH's multiple fingers *outside* the menu to begin inking with multiple fingers, making it easier to draw grass, hair, or textures.

# 6. INFORMAL OBSERVATIONS

Before designing the Multitouch Radial Menu, we implemented a Toolglass for a multitouch screen, and found it awkward for two reasons. First, the dominant hand's (DH's) finger causes occlusion during click-through operations, unless an offset is introduced between the finger and the cursor. However, without detection of hover, such an offset makes it difficult to know where to aim before the finger has landed on the screen. Second, dragging the non-dominant hand's (NDH's) finger across a large screen feels tiring and slow compared to moving a mouse.

We asked a group of 12 users to perform a series of commands (requiring 2, 3, or 4 DoF) with either the Multitouch Radial Menu or with our implementation of Toolglass on the Cintiq. Users found the Toolglass tiring because it required them to use both hands at all times, whereas the Multitouch Radial Menu could be used with one or both hands. All users subjectively preferred the Multitouch Radial Menu over the Toolglass.

# 7. LIMITATIONS

Because the North East – South West direction is more difficult to access with only a right hand, and both hands are required to access a submenu, our menu is not well suited to mobile devices held in one hand. A 10-inch-or-larger screen on a stable platform is best.

When trying to manipulate an object close to the edge of the screen, users may sometimes lack room for both fingers, however this can be addressed by having the user first invoke a camera command to pan the object away from the screen's edge before manipulating the object.

The fastest version of our menu, with a single ring and symmetrical layout, has only room for either 5 commands

(including the default command), or 4 top-level commands with one submenu, or 3 top-level commands plus two submenus, where each submenu might contain up to 20 commands. More commands could be accessed by sacrificing kinesthetic continuity and introducing modes, such as with a toolbar to switch between different menus.

Users often needed 10 minutes of explanation and practice to understand how to operate the menu. Future work could develop better visual feedback or metaphors to explain the menu to users.

# 8. CONCLUSION

We have presented the Multitouch radial menu, which supports continuous control over 4 DoF command arguments, submenus, and a default item easily accessed at the center of the menu. Our experimental evaluation examined the effects of number of rings, symmetry (or lack thereof), use of a 2nd hand and stylus, and enables a better understanding of the tradeoffs associated with speed, error rate, and number of items in the root menu. The taxonomy in Figure 1 also summarizes several ways of comparing previous techniques, and may guide designers in choosing a technique according to hardware constraints.

# 9. ACKNOWLEDGMENTS

Thanks to Tovi Grossman, Ken Hinckley, and Nathalie Henry Riche for advice. Funding was provided by NSERC.

# 10. REFERENCES

[1] G. Bailly, A. Demeure, E. Lecolinet, and L. Nigay. Multitouch menu (MTM). In *Proc. IHM*, 2008.

[2] G. Bailly, J. Müller, and E. Lecolinet. Design and evaluation of finger-count interaction: Combining multitouch gestures and menus. *IJHCS*, 70(10), 2012.

[3] N. Banovic, F. C. Y. Li, D. Dearman, K. Yatani, and K. N. Truong. Design of unimanual multi-finger pie menu interaction. In *Proc. ACM ITS*, 2011.

[4] E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. D. DeRose. Toolglass and magic lenses: The see-through interface. In *Proc. SIGGRAPH*, 1993.

[5] N. Chen, F. Guimbretière, and C. Lockenhoff. Relative role of merging and two-handed operation on command selection speed. *IJHCS*, 66, 2008.

[6] G. Fitzmaurice, A. Khan, R. Pieké, B. Buxton, and G. Kurtenbach. Tracking menus. In *Proc. ACM UIST*, pages 71–79, 2003.

[7] G. Fitzmaurice, J. Matejka, A. Khan, M. Glueck, and G. Kurtenbach. PieCursor: merging pointing and command selection for rapid in-place tool switching. In *Proc. ACM CHI*, pages 1361–1370, 2008.

[8] C. Foucault, M. Micaux, D. Bonnet, and M. Beaudouin-Lafon. SPad: a bimanual interaction technique for productivity applications on multi-touch tablets. In *Extended abstracts of CHI*, 2014.

[9] M. Frisch, J. Heydekorn, and R. Dachselt. Investigating multi-touch and pen gestures for diagram editing on interactive surfaces. In *ITS*, 2009.

[10] E. Ghomi, S. Huot, O. Bau, M. Beaudouin-Lafon, and W. E. Mackay. Arpège: learning multitouch chord gestures vocabularies. In *Proc. ACM ITS*, 2013.

[11] T. Grossman, K. Hinckley, P. Baudisch, M. Agrawala, and R. Balakrishnan. Hover widgets: using the tracking state to extend the capabilities of pen-operated devices. In *Proc. ACM CHI*, 2006.

[12] F. Guimbretière and T. Winograd. FlowMenu: Combining command, text, and data entry. In *Proc. ACM UIST*, pages 213–216, 2000.

[13] C. Gutwin, A. Cockburn, J. Scarr, S. Malacria, and S. C. Olson. Faster command selection on tablets with FastTap. In *Proc. ACM CHI*, 2014.

[14] K. Hinckley, P. Baudisch, G. Ramos, and F. Guimbretière. Design and analysis of delimiters for selection-action pen gesture phrases in Scriboli. In *Proc. ACM CHI*, pages 451–460, 2005.

[15] K. Hinckley, F. Guimbretière, P. Baudisch, R. Sarin, M. Agrawala, and E. Cutrell. The Springboard: multiple modes in one spring-loaded control. In *Proc. ACM CHI*, pages 181–190, 2006.

[16] P. Kabbash, W. Buxton, and A. Sellen. Two-handed input in a compound task. In *Proc. ACM CHI*, 1994.

[17] K. Kin, B. Hartmann, and M. Agrawala. Two-handed marking menus for multitouch devices. *ACM TOCHI*, 18(3):16, 2011.

[18] G. Kurtenbach and W. Buxton. The limits of expert performance using hierarchic marking menus. In *Proc. ACM CHI*, pages 482–487, 1993.

[19] G. Kurtenbach, G. Fitzmaurice, T. Baudel, and B. Buxton. The design of a GUI paradigm based on tablets, two-hands, and transparency. In *CHI*, 1997.

[20] G. Kurtenbach, G. Fitzmaurice, R. Owen, and T. Baudel. The Hotbox: Efficient access to a large number of menu-items. In *Proc. ACM CHI*, 1999.

[21] G. J. Lepinski, T. Grossman, and G. Fitzmaurice. The design and evaluation of multitouch marking menus. In *Proc. ACM CHI*, pages 2233–2242, 2010.

[22] Y. Luo and D. Vogel. Pin-and-cross: A unimanual multitouch technique combining static touches with crossing selection. In *Proc. ACM UIST*, 2015.

[23] M. McGuffin, N. Burtnyk, and G. Kurtenbach. FaST sliders: Integrating Marking Menus and the adjustment of continuous values. In *Proc. GI*, 2002.

[24] M. J. McGuffin and I. Jurisica. Interaction techniques for selecting and manipulating subgraphs in network visualizations. *IEEE TVCG*, 15(6):937–944, 2009.

[25] S. Pook, E. Lecolinet, G. Vaysseix, and E. Barillot. Control menus: Excecution and control in a single interactor. In *Extended abstracts of CHI*, 2000.

[26] J. Raskin. *The Humane Interface: New Directions for Designing Interactive Systems*. Addison-Wesley, 2000.

[27] A. J. Sellen, G. P. Kurtenbach, and W. A. S. Buxton. The prevention of mode errors through sensory feedback. *Human Computer Interaction*, 7(2), 1992.

[28] J. O. Wobbrock, L. Findlater, D. Gergle, and J. J. Higgins. The aligned rank transform for nonparametric factorial analyses using only ANOVA procedures. In *Proc. ACM CHI*, pages 143–146, 2011.

[29] T. Yoshikawa, B. Shizuki, and J. Tanaka. HandyWidgets: local widgets pulled-out from hands. In *Proc. ACM ITS*, pages 197–200, 2012.