Thoughts on Circular Context Menus

Ian D. Eccles
ianpub [at] mathish.com

November 6, 2005

Abstract

A circular context menu aims at improving the overall usability of the context menus found in various applications. By using a circular layout, a user may perform desired actions more quickly than with a traditional list-style context menu. The goal of this document is to flush out the mathematics of constructing such a menu and compare access times for both types of context menus.

1 Introduction

Before beginning, it is important that certain concepts are familiar to the reader. The most important concept is the **context menu**. A context menu is a menu that offers actions in a specific context. Often times these menus are displayed using a secondary click of the mouse, or other pointer device, at the point on the screen where the click occurred. Due to this behavior, context menus are often referred to as "pop-up menus." The menu then offers actions to be performed that are specific to the selected object (be it a desktop icon, or a selection of text), thus providing them with a specific context. Nearly every computer user has encountered context menus, so this concept should already be familiar to the reader. An example of a context menu, taken from a snapshot of the open source text editor jEdit, can be found in Figure 1.

1.1 Background on Fitts' Law

Developed in 1954, Fitts' Law is a model of rapid, aimed movement in one dimension and is expressed by the formula:

$$\tau = a + b \log_2 \left(\frac{2A}{W} \right),$$

where τ is the time of movement, A is the distance from the starting point to the target's center, W is the width of the target, and a and b are constants [1]. For the purposes of

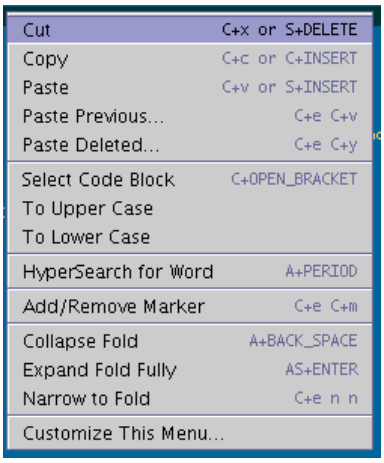

Figure 1: An Example Context Menu

this paper, we will rephrase Fitts' Law as follows:

$$\begin{aligned}\tau &= a + b\log_2\left(\frac{2A}{W}\right)\\ &= a + b\log_2\left(\frac{2\left(\alpha+\frac{1}{2}W\right)}{W}\right)\\ \tau &= a + b\log_2\left(\frac{2\alpha}{W}+1\right) \qquad (*)\end{aligned}$$

Here, the expression $\alpha+\frac{1}{2}W$ has been substituted for A, where α is the distance from the starting point to the edge of the target. Based upon how A was initially defined, this substitution and the resulting equation should be obvious.

1.2 Applying Fitts' Law

One of the primary results of Fitts' Law is that the further away a target is from the starting point, the larger the target needs to be in order to minimize τ. Minimizing τ should be the goal of every interactive interface. This ensures that the user spends less time to get the desired results from an application, thus improving the usability of said application.

There are a number of ways Fitts' Law can be applied to an application to improve its usability. A few examples are [2]:

- Associate actions with the edges of the screen. As mouse movement is typically bound within the screen, this produces targets with, effectively, infinite width. The result, by Equation (*), is that the time spent in motion to get to these points is constant, a. All one has to do is slide the mouse as fast as they can towards the edge without being concerned about "over shooting" the target.

- Make targets that are further away larger. This can be accomplished by either physically increasing the size of distant targets, or by decoupling the onscreen position of the mouse from the motion of the mouse, requiring more force to move the mouse further distances.

- Utilize circular context menus instead of linear ones. This, generally, results in less movement to access the desired action. It has the added benefit that all of the actions are equidistant from the current location of the mouse pointer (assuming the context menu is centered on the mouse's current location.) Also, as one moves further from the center, the size of the target increases due to the geometry of the circle.

This paper will focus on the latter method of applying Fitts' Law to application design.

There are some shortcomings to be had with Fitts' Law, and thus any user interface models stemming forth from it. This will be discussed in greater detail later.

2 The Trouble with Context Menus

A context menu is a useful way to group actions that apply to a specific context. Consider a file system viewer that, naturally, displays all of the files and sub-directories for a given directory. Commonly, such a viewer will have a common menu to perform various actions on a file (cut copy and paste, change ownership or permissions, and so forth.) However, different file types could have different actions available. For instance, text and image files may have a "Print" action, whereas a "Print" action for a binary library may be nonsensical. Therefore, if someone selected a text or image file, or a set of these files, and navigated about in the common menu, they would see the "Print" action is enabled; however, if a binary library were selected, the action would be disabled. Enter the context menu. I select a set of files, right click, and the "Print" action appears a short distance away from my cursor. Now, omitting actions found in the context menu from the common menu is generally considered bad form, but that is a topic for some other paper.

Context menus present some problems. The primary one being that they are conventionally displayed as a list of actions, with the upper left-hand corner of the list lying directly beneath the cursor. The further the desired action is from the top of the list, the more movement a user must make to get to the item. Each element in the list of actions is also, typically, the same size. Thus, the further down the list, the harder it becomes to accurately hit the target. The ultimate result is that a poor effort has been made to minimize τ. A conventional method of combating poor access times is to order the action list by frequency of use. In a file system viewer, the most common action is to open the file with its associated application. This is often the first item in such a context menu. In the GNOME environment, the second option is to open the file with an alternative viewer. This certainly does help matters. We could weight access times by their frequency to get a feel for τ_{avg}, the average time spent in motion, and I imagine such a layout would fair quite well. Another less conventional approach (in that, I have yet to see it implemented) is to present a context menu aligned to the current cursor by left edge and vertical center. If all items are equally likely to be accessed, this decreases

the maximum distance that must ever be traversed to get to an action. Both of these techniques work well to cope with the problems inherent in a list layout.

Another potential problem with the list-style context menu is that the user only gets one piece of spatial information to associate with each action: distance. In a traditional context menu, all motion is in the same direction (downward), so our spatial senses can only fixate on the single variable of distance. I conjecture that feeding the senses differing directional information for each action would improve usability. I have no evidence on which to base this claim, it's just been my experience when using various applications. This problem seems harder to rectify than the previous. One thought (though I admit I have not flushed this out) is that if the context menu's left edge were not directly beneath the mouse pointer, but rather some fixed distance away, we might get some directional feel out of it. However, as the number of menu items increased, that directional sense would likely be reduced.

3 A Circular Context Menu

To address the shortcomings previous listed, let us consider the use of a circular context menu. In such a menu, actions are distributed around the circle occupying equally sized "pie slices." Clicking anywhere within a given slice activates the associated action. While there are advantages to be gained by using distributions where the "pie slices" are not all the same size, the discussion in this paper will be limited to evenly distributed circular context menus. An example of this model can be found in Figure 2.

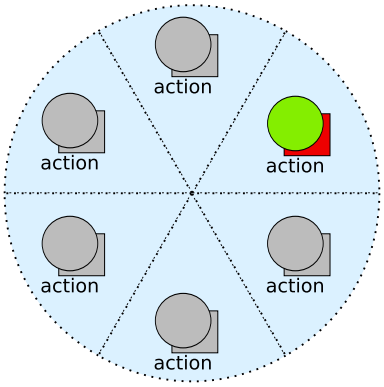

Figure 2: A Mock-up of a Circular Context Menu

Consider the shortcoming of list-style context menus related to distance. The further down the list a desired action is, the greater the time spent in motion towards the target. A circular context menu addresses this by all actions being equidistant from the starting point, thus the time to access any action is, generally, the same as accessing any other action. The exceptions to this rule are those actions which lie along an axis, as it is slightly faster to move the mouse mostly horizontally or mostly vertically than it is to move along an arbitrary trajectory. However, for sufficiently few actions, the

angular measure of a given "pie slice," which will be referred to as θ from this point forward, will be large enough to allow us to only have to move a few pixels along a trajectory before we are squarely within the desired action's area. Thus, even if it is slightly faster to move along an axis than an arbitrary trajectory, one does not have to move very far before they have acquired their target. The other strong benefit of circular context menus with regard to the distance problem is that the target area grows as one moves outward from the center. The size of the target increases with the distance from the center.

With regard to the second problem mentioned earlier, which is admittedly only a conjecture, a user can feed his or her spatial senses with directional information. This would mean that circular context menus are more usable than list-style ones, which only offer distances to differentiate actions, assuming of course that the conjecture is correct.

4 The Limitations of a Circular Context Menu

A circular context menu can offer definite advantages over the traditional list-style menu. To ensure that these advantages are being offered, there are a few problems one must be able to address. The problems listed below are the primary ones that must be planned for at the time of interface design; however, there are a few more that are discussed in Section 7.

Problem 1 *Noisy Motion*

As touched on briefly in Section 3, motion along an arbitrary trajectory tends to be more error prone than motion along a single axis. The general problem here is that all mouse motion is error prone. Movement along any trajectory, even those that lie on a single axis, results in error. The user moves a few pixels to high here, a few more pixels too low there. We would like to ensure that our circle is sufficiently large as to allow for quick movements toward the larger area of the targets, where "noisy motion" is no longer a problem.

Problem 2 *Small Angular Measures*

This problem appears when a sufficiently large number of elements is in a circular context menu. The distance between actions near the center of the circle is very small, so little errors could result in the wrong action being selected. This is rectified by increasing the radius of the circle, which results in greater spacing as we move outward. One then aims for the outer, larger, area of the "pie slice." This of course does increase access time, but as the targets are getting larger the further they are from center, the effects of over shooting can be minimized.

Both of these problems are solved in the same way, by making an appropriate choice for the radius of the context menu. We will seek out a lower bound for the radius, referred to as r_{min} from here on, then any choice of radius greater than or equal

to r_{min} will be sufficient for a given circular context menu. As the two problems outlined depend on distinct quantities, namely the number of elements in the menu, n, and "noisy motion," it should be no surprise that r_{min} will be a function of these two variables.

5 Into the Realm of Trigonometry

It was going to happen sooner or later. I knew it as did you, the reader. Eventually the written word of this discussion was going to break down into the crude form of conversation I like to call mathematics. If you have no interest in delving into the derivation of the end results of this work, simply note that the choice in radius for a circular context menu is very important. If you have no interest in the derivation, but would like to see the final results, skip onward to Section 6. For those of you in it for the long haul, I salute you.

We will begin by defining a few quantities. A circular context menu has $n \geq 1$ elements. Each of the n "pie slices" has an angular measure of $\theta = \frac{2\pi}{n}$. The quantity Δ is a quantity that ultimately arises from empirical measurement. It will be defined as twice the maximum distance from a trajectory that bisects a "pie slice" and the path traversed by the cursor. To illustrate this by example, consider Figure 3, where the green stroke represents a user's attempt to traverse the trajectory that bisects the "pie slice." In practice, multiple measurements would need to be taken from multiple users, and an average of the maximum distances would then need to be computed. This is not a difficult quantity to measure, I just feel that presenting error gathered only from myself would be a bit dishonest, and as I lack the resources to derive Δ through experimentation, I feel it is best to simply represent it only symbolically. The next quantity of interest is α which is defined as,

$$\alpha = \frac{\Delta}{2\tan\left(\frac{\theta}{2}\right)}.$$

Figure 3 provides the geometric significance of this value. It is important to note that the choice to use α here as well as in Equation (*) is no coincidence. The formula for α is a direct result of the fact that the triangle in the figure, with base Δ and height α is an isosceles triangle.

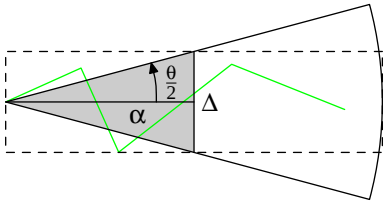

Figure 3: Determining Δ from "Noisy Motion"

The next, and last, two quantities that we will define are H and W, which represent the height and the width of the action's icon within the circular context menu, respec-

tively. In this model, W can be defined as any size the application developer sees fit, I will make a suggestion for W, but this suggestion should not be assumed to be the best choice. As far as the icon height is concerned, we will actually vanish that symbol away fairly quickly. Bear in mind we are seeking a lower bound for the radius of our context menu. We want to place the action's icon outside of the shaded region seen in Figure 3, because that area is more error prone, and I would be willing to wager that the icon placement will determine the user's target for movement, regardless of the fact that the entire "pie slice" activates the action. The presence of the icon will, at least initially, encourage the user to aim directly for it. If the icon's height is too small, it encourages a lot of mouse movement until the cursor is on top of the small icon. If the icon is too large, it becomes a factor in the size of r_{min}. We therefore will settle on the an icon height of $H = \Delta$. This is the perfect value to use so the icon can be placed at the very frontier of the error prone shaded region of Figure 3, without wasting space or having an impact on the radius of the context menu.

A quick look at Figure 4 reveals the geometric layout each "pie slice." The shaded region indicates where the icon will be drawn. The dashed triangle provides the necessary information to determine r_{min}, as the triangle has a hypotenuse of r_{min}, a base of $\frac{\Delta}{2}$ and a height of $\alpha + W$. From this, r_{min} is derived as:

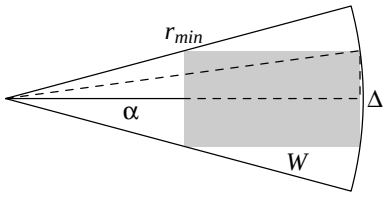

Figure 4: Calculating r_{min}

$$
\begin{aligned}
b &= \alpha + W \\
h &= \frac{\Delta}{2} \\
r_{min} &= \sqrt{b^2 + h^2} \\
&= \sqrt{(\alpha + W)^2 + \frac{\Delta^2}{4}} \\
&= \sqrt{\frac{\Delta^2}{4} + \alpha^2 + 2\alpha W + W^2}
\end{aligned}
\tag{1.a}
$$

Recalling that $\alpha = \frac{\Delta}{2\tan\left(\frac{\theta}{2}\right)}$, we can expand Equation (1.a) further:

$$\begin{aligned} r_{min} &= \sqrt{\frac{\Delta^2}{4} + \left(\frac{\Delta}{2\tan\left(\frac{\theta}{2}\right)}\right)^2 + \frac{\Delta}{\tan\left(\frac{\theta}{2}\right)}W + W^2} \\ &= \sqrt{\frac{\Delta^2}{4} + \frac{\Delta^2}{4\tan^2\left(\frac{\theta}{2}\right)} + \frac{\Delta}{\tan\left(\frac{\theta}{2}\right)}W + W^2} \\ &= \sqrt{\frac{\Delta^2}{4} + \frac{\Delta^2}{4\tan^2\left(\frac{\pi}{n}\right)} + \frac{\Delta}{\tan\left(\frac{\pi}{n}\right)}W + W^2} \end{aligned} \tag{1.b}$$

We now have r_{min} defined in terms of of the two free variables, n and W and the empirically measured quantity Δ. It is at this point that I offer my suggestion of $W = \Delta$. This results in a further expansion of Equation (1.b):

$$\begin{aligned} r_{min} &= \sqrt{\frac{\Delta^2}{4} + \frac{\Delta^2}{4\tan^2\left(\frac{\pi}{n}\right)} + \frac{\Delta^2}{\tan\left(\frac{\pi}{n}\right)} + \Delta^2} \\ &= \Delta\sqrt{\frac{1}{4} + \left(\frac{1}{2\tan\left(\frac{\pi}{n}\right)} + 1\right)^2} \end{aligned}$$

6 The Results

Through the lengthy derivation in the previous section, we have arrived at a means of determining r_{min} given n and Δ:

$$r_{min} = \Delta\sqrt{\frac{1}{4} + \left(\frac{1}{2\tan\left(\frac{\pi}{n}\right)} + 1\right)^2} \tag{1}$$

There are values of n for which one has to be weary of given this equation, namely when $\tan\left(\frac{\pi}{n}\right) = 0$ or when $\tan\left(\frac{\pi}{n}\right)$ is undefined. By definition, $\tan(\rho) = 0$ when $\rho = k\pi$, $k \in \mathbb{Z}$. In this case we only need concern ourselves with $n = 1$ and, theoretically, $n \to \infty$. In the first case, we only have one menu option, so the entire circle is our "pie slice," thus r_{min} need only be big enough to contain a single icon, with the icon placed in the center of the circle. In the case of $n \to \infty$, our problem becomes that we will require a larger radius, and thus more space for the menu. In theory, as n continues to increase, we would have infinitesimally narrow "pie slices" to work with, thus making the whole interface useless. Clearly, limiting the number of elements in this context menu is important. Next we consider when $\tan\left(\frac{\pi}{n}\right)$ is undefined. Again by definition, $\tan(\rho)$ is undefined when $\rho = \frac{\pi}{2} + k\pi$, $k \in \mathbb{Z}$. Obviously, we are not concerned with negative values of ρ as the quantity $\frac{\pi}{n}$ is never negative. Moreover, as n is a whole number, $\frac{\pi}{n}$ falls within the domain of ρ only when $n = 2$. In this case, $r_{min} = \Delta\frac{\sqrt{5}}{2}$, and you end up with the two icons back-to-back along the center of the circle, with the outside

corners of the icons intersecting with the circle. With a diagram of what has just been described, and a bit of trigonometry, or through a bit of analysis of $\lim_{n\to 2} r_{min}$, one will arrive at the same formula for r_{min}. Finally, the last value of n we need to be weary of is $n = 0$. However, this case is trivial to reconcile. If a context menu has no actions, no context menu is displayed.

Now, with all of that taken care of, we will consider a couple examples.

Example 1 *Suppose a good choice for Δ has been found to be 32 pixels. What is a suitable choice for r_{min} given 8 menu actions?*

By the formula for r_{min} given in (1), and the assistance of a calculator (or trig. tables, pick your poison), we can quickly conclude that $r_{min} \approx 72.41707$ pixels. For the sake of verification, I have produced Figure 5 to illustrate this. Obviously, r_{min} is not 72 pixels, nor is $\Delta = 32$ pixels as the image has been scaled. However, the proportions have been preserved.

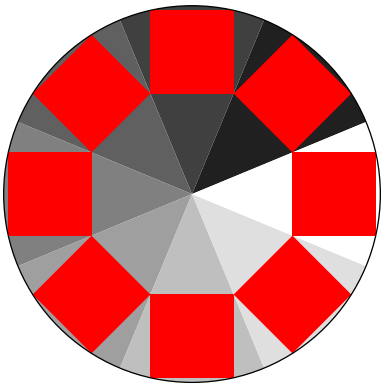

Figure 5: A Graphical Representation of Example 1

Example 2 *Suppose that we use the parameters from Example 1, and further suppose that the modified Fitts' Law formula, Equation* (*)*, applies to this model. Calculate τ, the time of motion, and compare it to the worst case time of motion for a traditional context menu with the same number of items, each item requiring 16 pixels of height.*

We will begin by using our new values of α and W to rephrase Equation (*) a bit more:

$$\begin{aligned}\tau &= a + b\log_2\left(\frac{2\Delta}{2\Delta\tan\left(\frac{\pi}{n}\right)} + 1\right)\\ &= a + b\log_2\left(\frac{1}{\tan\left(\frac{\pi}{n}\right)} + 1\right)\\ &\approx a + 1.77155b\end{aligned}$$

Next, we calculate the worst case access time of a traditional context menu, which happens to be if we're accessing the last menu item in a list of 8. In this case, $\alpha = 16 \times 7 = 112$, and $W = 8$

$$\begin{aligned}\tau &= a + b\log_2\left(\frac{2 \times 112}{8} + 1\right) \\ &= a + b\log_2(15) \\ &\approx a + 3.90689b\end{aligned}$$

Clearly, if our assumptions about Fitts' Law holding for this model, and the model of regular context menus, we have significantly reduced time spent in motion towards menu actions.

7 Final Thoughts

7.1 The Good

The circular context menu model does provide many benefits, even if it is considerably different from what most of us are accustomed to using. As Example 2 demonstrated, we could stand to gain a fair bit more performance from using such a model.

7.2 The Bad

There are still problems to be found in this model. The first is that Δ is never precisely defined, and likely will not be by me. I simply lack the resources to get a good empirical measure of this quantity. We also encounter the problem demonstrated in Figure 5, where the icon spaces have to be rotated to properly fit in their respective "pie slices." This does not mean the actual icons must also be rotated, just that additional calculations must be performed to ensure that a given icon with a given orientation will fit inside of the icon space. And finally, one must be selective in what items they wish to drop into a circular context menu, as too many options will yield a larger radius. The end result is more screen-estate must be utilized to display the menu, which could be awkward for the user, and the user will have to move further outward to be certain they are accessing the right option, thus reducing the effectiveness of this layout over the conventional list-style context menu. There is also the problem of screen resolution. We can hold Δ as a constant, and solve Equation (1) for n and thus determine the maximum number of elements a circular context menu can hold for a given screen resolution. Using the values from Example 1, we find that if the screen resolution is 640×480, thus $r_{min} = 240$, the menu can hold $n \approx 40$ elements before expanding beyond the dimensions of the screen (assuming of course, I have not made an error in my calculations.)

7.3 The Ugly

Fitts' Law is inherently one dimensional. There is no reason to believe it ties into two dimensions so nicely. I made attempts to reconcile this by assuming the user is mov-

ing along a straight trajectory (albeit not necessarily purely horizontal or vertical) with some error (the whole Δ factor.) However, it is not as if I have then magically converted two dimensions into one dimension, nor have I converted a formula of one dimension into a formula of two dimensions. So, be warned, the results of Example 2 may be the product of "garbage in, garbage out." The second ugly problem is how does one handle hierarchical circular context menus? The trivial solution is to not use them, but this is not always an option. Various solutions to this problem present unique problems all their own. Personally I feel this question is rich enough to deserve its own paper, to be written at a later date.

If you have any further questions or comments, feel free to submit them to `ianpub[at]mathish.com`. Bear in mind, I get a fair amount of spam sent to this address, so I make no guarantee that I'll reply, or even see, your comments.

References

[1] B. Amento, et al., *CS 5724: Models and Theories of Human-Computer Interactions*, `http://ei.cs.vt.edu/~cs5724/g1/`, Fall 1996.

[2] B. Tognazzini, *A Quiz Designed to Give You Fitts*, `http://www.asktog.com/columns/022DesignedToGiveFitts.html`, February 1999.