ITE – Course Notes | ITE

Interaction Design and Evaluation

Pere-Pau Vázquez – Dep. Computer Science, UPC

Contents

1 Introduction

Successful interface design usually leads to positive feelings and enthusiasm among users. An enthusiastic user will likely recommend the application to another user. He or she will also be satisfied with his or her competence in achieving the tasks, the knowledge of the interface, and will enjoy its use.

Such satisfying interfaces are also referred to as *direct-manipulation interfaces*. They provide visible support for the actions of interest, that is, objects and actions are carried out with visual feedback. Moreover, they also provide rapid, reversible, and incremental actions. Typed commands are replaced by pointing actions on the object of interest.

In order to provide the best service to our users, we need to design the interaction properly. Some tasks will be easily accomplished by direct manipulation, such as when editing images, but other tasks, such as aligning objects in an image can be easier through a menu option than moving the objects directly. Therefore, we must properly select the adequate interaction for each task.

Pointing and selection are two related fundamental tasks in graphical user interfaces. Object selection techniques involving physical interaction are constrained by the human motor system as the speed and the accuracy of any gesture are limited by the nervous and muscular systems. In order to understand how we process information and interact with elements in the interfaces, it is very useful to know which laws seem to govern our behavior (reading capacity, clicking time and effort…). Getting a knowledge into these laws will make possible to analyze and to design better user interfaces.

The objective of this document is to present both the Hick-Hyman law and the three "Laws of Action", as defined by some researchers (e. g. Zhai), as well as to point some guidelines to use them in effective user interface design.

2 Background

Pointing and selection tasks have been studied thoroughly from a theoretical point of view. We will present here the theories that try to give an explanation on users' performance on pointing and selection tasks. Since all of them are based on a previous mathematically grounded theory: *Information Theory*, we will give first some insights on it.

2.1 Information Theory

Information Theory was born as theory that dealt with data communication. In 1948 Claude E. Shannon presented his seminal work *A Mathematical Theory of Communication* [Shannon48]. Based on previous works by Nyquist and Hartley [Nyquist24, Hartley28] on the transmission of electrical signals for telegraphic communication, he developed a measure named Shannon Entropy that measures the amount of information to be transmitted by a message. Shannon's approach to communication uses the following elements:

- **Information source:** The element that produces a message or sequences of message.
- **Transmitter:** Operates on the message to make it transmissible through a medium.
- **Channel:** The medium that transmits the message.
- **Receiver:** The element that reconstruct the message to the destination.

Shannon's system was a telegraph. In this context, the message is encoded into a signal and this signal transmitted through the channel. The receiver transforms back the signal into a message and delivers it.

The general scheme of this communication process appears in Figure 1.

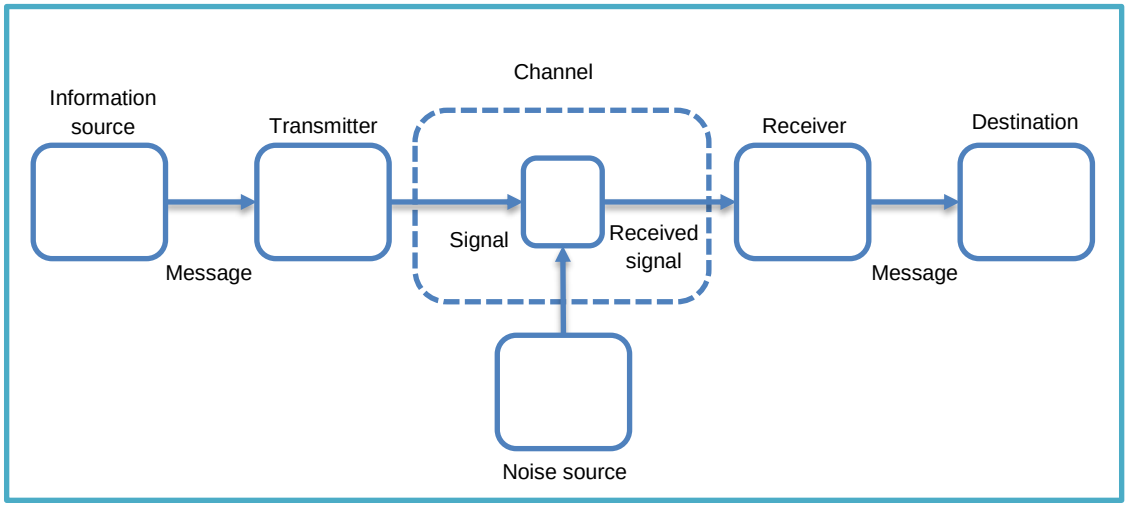

Figure 1: Diagram of a general model of communication system.

The amount of information that a communication channel transmits in a fixed amount of time is referred to as the channel capacity. Channels are bound by physical limitations and thus have different capacities. Other factors, such as noise, affect the communication channel. Thus, the effective capacity is always lower than are the theoretical specifications.

2.2 Information and Uncertainty

Information and Uncertainty are two terms that appear commonly together in Information Theory, so they need a deeper discussion. The first thing to take into account is that those two terms are used to describe the process of selecting one or more elements from a set of elements.

The following development is taken from Schneider's "Information Theory Primer" [Schneider2013].

Suppose we have a device that generates any of these 3 symbols: A, B, or C, every time we ask for one symbol. Before the symbol is produced, we do not know which is the one to come. We can say that we are *uncertain* on the next symbol. Once a symbol has appeared, since we know which of the symbols has appeared, our uncertainty decreases. It decreases because we *have received information*. That is, **information is a decrease in uncertainty**.

Then it comes the problem of measuring this uncertainty. Without many details on how they appear, we may say that a device that produces three different symbols has an uncertainty of log(3). We will discuss the units later. Then, if we have a second device that produces two different symbols (say 1 and 2), we will say that it has an uncertainty of log(2). We can make the following experiment: we iteratively produce elements from the first and the second device and combine both. If we do this, the number of different (combined) elements we will have will be six: A1, A2, B1, B2, C1, and C2. Note that this implies an uncertainty of log(6). Intuitively, when we combine the two devices, we would expect the uncertainty to increase. Actually, if we add the uncertainties of the both original devices we will have log(3)+log(2) = log(6). So if we use logarithms to encode uncertainty, we can add it as we would intuitively expect.

This can be applied with any logarithm, but the typically used here are logarithms with base 2. If we use such logarithms, the value measured by the uncertainty will be *bits* [Shannon54].

Thus if a device produces one single symbol, we are uncertain by $\log_2 1=0$ bits. This means that we have **no uncertainty** about what the device will do next, since it will produce the symbol we expect. If, on the contrary, we have a device that produces two symbols, the uncertainty would be $\log_2 2 = 1$ bit.

So far, we have assumed that our devices produce symbols with equal probability, and in that case, the uncertainty is $\log_2$ (M), where M is the number of different symbols a device may produce. If we have M symbols and the probability of producing any symbol is the same, we can say that the probability we have is 1/M. Then, we can change the logarithm in the following way:

$$\log_2 (M) = \log_2 \left(\left(\frac{1}{M} \right)^{-1} \right) = \log_2 (P)^{-1} = -\log_2 (P)$$

That is, the uncertainty of getting a certain value is the –log of the probability of this symbol. This value, also called the *surprise* (or *surprisal*) indicates whether we may or not expect a certain value. If we are *certain* on a certain symbol, there will be no *surprise* when it emerges. If we recall the previous example, we have total certainty when the set of symbols is composed by a single element. In that case, the surprise ($\log_2 (P)$) will be zero, as can be seen when we calculate it: $\log_2 (1) = 0$.

For the different symbols that can be produced, we have that their probability sums up to 1:

$$\sum_{i=1}^{M} P_i = 1$$

This can be used to analyze the common case where the probability of producing different symbols varies. That is, some symbols appear more often than others. In this case, we may have that probability values are not all $1/M$, but this value can change (provided that the total sum adds up to 1). In that case, we will be more *surprised* to see a low probability symbol than seeing a higher probability symbol.

2.3 Shannon Entropy

As said, information is defined in Information Theory as a reduction in uncertainty. And the purpose here is to measure the information.

The classical measure of information, as presented by Shannon is measured over a distribution of probabilities. It is defined as the average *surprisal* from a set of symbols produced by a device. If we measure the surprise for the infinite set of symbols that can be produced by a device, the frequency of each symbol transforms to the probability. Then, if we sum over all the symbols set, we will get:

$$H = \sum_{i=1}^{n} p_i \log_2 \left(\frac{1}{p_i}\right) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

where n is the number of alternatives, and p_i is the probability of the ith alternative (here I changed P by p since it is the common notation found in literature). H is the entropy of the message that is to be transmitted. It is actually the amount of information expected to be received at the destination. It is also designated as $H(x)$. Since there is usually an interference during transmission, the information actually received in destination is reduced, and it is designated $H(y)$. The average information that is faithfully transmitted is calculated as:

$$R = H(x) - H_y(x)$$

where $H_y(x)$ is the *equivocation*, or the conditional entropy of x when y is known. As we will see later, in Hick's and Fitts's laws, when a participant commits no errors, he or she is said to be extracting all the expected information of the stimuli.

3 Choice-reaction time

3.1 Hick-Hyman Law

William E. Hick was one of the first to apply Information Theory to psychological problems. His theory had as objective to measure the relationship between choice reaction time and stimulus information content (entropy). It has been previously discovered, as early as 1868 [Donders68] and Merkel [Merkel85], that it takes longer to respond to a stimulus when it belongs to a large set as opposed to a smaller set of stimuli. This regularity caught the attention of psychologists, who saw its analogy to the classic Information Theory.

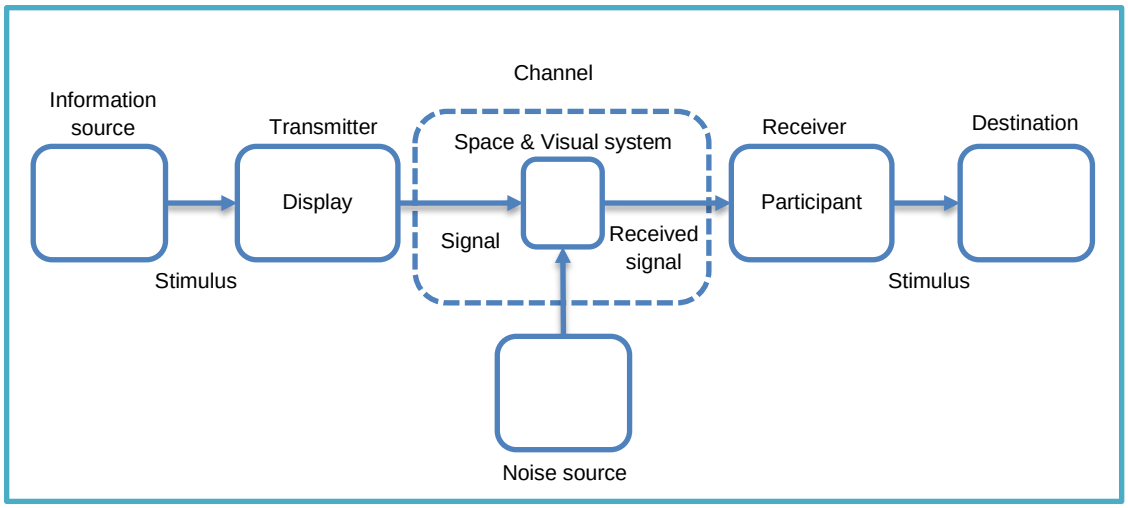

Figure 2: Diagram for the choice-reaction time experiment as a model of communication system.

Hick's Law describes human decision time (or *Reaction Time*) as a function of the information content conveyed by a visual stimulus. After his first experiments [Hick51, Hick52], Ray Hyman [Hyman53] extended his work, and nowadays this law is also called Hick-Hyman Law [Seow2005]. It is defined, in its most general way, as:

$$RT = a + b\, H_T$$

where a and b are empirically determined constants, and H_T is the transmitted information. The transmitted information is commonly fitted as

$$H_T = \log_2(n + 1)$$

where n is the number of equally probable alternatives and the *+1* indicates the uncertainty about whether to respond or not, as well as about which response to make [Card83].

Intuitively, one can reason that Hick's Law has a logarithmic form because people subdivide the total collection of choices into categories, eliminating about half of the remaining choices at each step, rather than considering each and every choice one-by-one, requiring linear time.

3.2 Evidences of Hick-Hyman law

Hick-Hyman's Law has been demonstrated in many experiments, and has been used to justify menu design. For instance, Cockburn *et al.* [Cockburn2007] and Cockburn and Gutwin [Cockburn2008] show that novice users tend to select with a visual search that takes linear time on the number of elements of a menu, while experts decide upon item location, and their decisions times are adequately predicted by Hick-Hyman's logarithmic formula. Allison *et al.* [Allison2004] also showed that Hick-Hyman's Law predicts accurately the decision time in card sorting tasks.

Landauer and Nachbar [Landauer85] observed that expert performance in hierarchical full-screen menu selections is well described by Hick-Hyman and Fitts components applied in series.

Sears and Shneiderman [Sears94] observed that selection times decay logarithmically with menu length for frequently selected items, but linearly with infrequent ones. It seems that participants move from linear visual search with unfamiliar items to Hick-Hyman decision times as the locations are learnt.

4 Fitts' Law – Law of Pointing

4.1 Formulation

Published in 1954, Fitts' Law states a linear relationship between task difficulty and movement time (MT). His formulation is also based on Information Theory. In this case, the human motor system is the communication *channel*, the amplitude of movement is the *signal*, and the target width is the *noise*. The task difficulty, in the first writings by Fitts ([Fitts54, Fitts54b, Fitts54c]), is expressed as:

$$ID = log_2\left(\frac{2A}{W}\right)$$

where *ID* is the *Index of Difficulty*, *A* is the amplitude of the movement, and *W* is the width of the target. Movement Time is then defined as a function of the Index of Difficulty:

$$MT = a + b\ ID$$

where *a* approximates the start/stop time in seconds for a given device, and *b* measures the inherent speed of the device. Both must be empirically determined for each device. Note the similarity of this formulation with the Hick-Hyman's Law. Posterior experiments noted that there is a lower adjustment of the regression curve when testing with low IDs. Actually, the relationship originally established by Fitts was using an approximation of Shannon's theorem that is valid only if the signal-to-noise ratio is large ([Fitts54, McKenzie89]). Therefore, other two approximations were developed, e. g. by Welford [Welford68]:

$$MT = a + b\ log_2\left(\frac{D + 0.5W}{W}\right)$$

Note that we substituted here the amplitude of movement (A) by the Distance (D). Or MacKenzie ([MacKenzie92]):

$$MT = a + b\ log_2\left(\frac{D}{W} + 1\right)$$

Several versions of Fitts' Law are used, and you can find them consistently in literature, but this formula has been demonstrated to provide good predictions in a wide range of situations.

Commonly, the Fitts' Law is applied to the task of pointing or clicking a button. In this case, the parameters involved are the distance *D* the pointer (e. g. mouse) has to cover, and the width *W* of the button in the direction of the movement. In Figure 3 all these elements are represented.

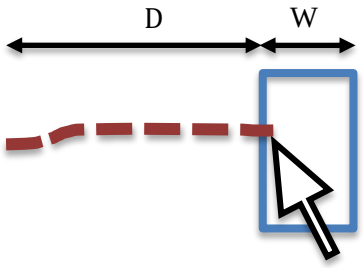

Figure 3: The pointing task according to Fitts' Law.

Several of the other formulae may be better indicated to address some concrete tasks. Note that non trivial variations may be due to differences such as changes in the direction of motion (vertically vs horizontally), device weight (heavier devices are harder to move), shape or size of targets (the original formulation, for instance, does not hold for low ID values), or arm position. So when experimenting with such a priori simple tasks, we must have all the variables under control, because trivially extending the Fitts' law may be invalid. We will see some extensions later.

Fitts' Law describes the Movement Time in terms of one dimensional displacement, as we have already seen. This is valid for either purely horizontal or purely vertical movements. The only thing that has to be taken into account is that we need to use the dimension of the pointing target in question that goes in the direction of the movement. Apart from that, the formula can be applied indistinctively (see Figure 4).

Figure 4: Fitts' Law can be applied to one dimensional movements, either in horizontal or vertical.

Another related measure is the Index of Performance, which is analogous to the Capacity in Shannon's theorem. Some authors call it throughput (TP). IP is measured as bits per unit, and calculated as:

$$TP = ID/MT$$

However, the use of *TP*, commonly applied to measure performance characteristics of input devices, is quite problematic. The main reason is that TP is only a constant (hence, something that can be representative of the device) when a (also called the intercept) is zero. If a was zero, then the throughput would be $TP = 1/b$ (b is the slope).

4.2 Example

We can show an example here. Let's imagine that we have empirically determined the constants a and b for a certain device. For instance, let *a=300 ms* and *b=200 msec/bit*. If we want to reach a target of 2 cm (*W=2*) at a distance of *14 cm* (*D=14*), then, according to Fitts' law, the Movement Time would be

$$MT = 300 + 200\ log_2\left(\frac{14}{2} + 1\right)$$

that is, *MT = 900 ms*.

4.3 Fitts' Law extensions

4.3.1 Fitts law for 2D

The original formulation of Fitts' Law is in one dimension. In all the very initial experiments, the movements demanded to the users were in one direction. With the advent of graphical user interfaces, the navigation region increases, and therefore it is interesting to evaluate the difficulty to reach objectives that require two-dimensional displacements.

Note that, when having non one-dimensional moves, we may point to a target that has different dimensions in height and width (see Figure 5). As a result, we do not have a proper dimension of the target, and some modifications must be applied, since the original formula cannot be used directly.

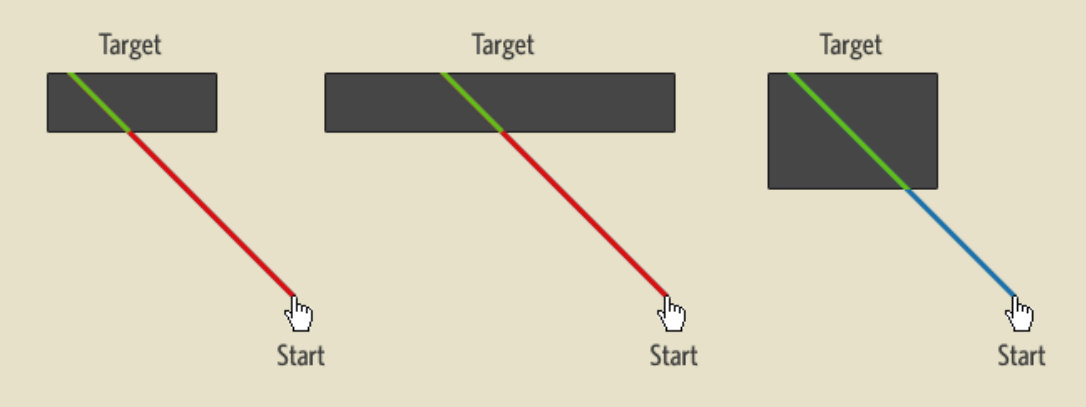

Figure 5: Pointing to a target with a 2D movement.

There have been several developments with this goal in mind. For instance, Crossman and Goodeve [Crossman93] suggested that a second term has to be added to take into account for the height of the target, developing the following formulation:

$$MT = a + b\ log_2\left(\frac{2A}{W}\right) + clog_2\left(\frac{2A}{H}\right)$$

Like in the previous cases, the constant c must be determined empirically. W is the amplitude constraint size and H is the height. Other researchers later refined this formulation. A more recent approach is due to Accot and Zhai [Accot97]. They suggest that a better formulation is this:

$$MT = a + b\ log_2\left(\sqrt{\left(\frac{D}{W}\right)^2 + \eta\left(\frac{D}{H}\right)^2} + 1\right)$$

Where η is a constant commonly in the range $1/3$ to $1/7$. This indicates somewhat that the directional constraint is less difficult to handle than the amplitude constraint of the same magnitude. Like in the previous case, W is the width and H the height constraints of the target. More formulations have been proposed, but the details are beyond the scope of this course.

4.3.2 Fitts for finger touch

This predictive model is a great help to decide location and size of buttons and other elements in the user interface.

For the case of very small targets, we cannot expect the user to be as efficient as with regularly sized elements. Some studies have found that, for very small elements, there is an extra time devoted to fine adjustment. Moreover, the use of touchscreens also modifies the timing we require to point targets. Sears and Shneiderman ([Sears91]) derived an extension of the Fitts' Law by analyzing the behavior both in tactile screens and small targets. This led to what some authors call the Precision Pointing Movement Time (the original authors call it modified version of Fitt's Law for touchscreens, or FFits):

$$FFits = a + b \log\left(\frac{cD}{W}\right) + d\ \left(\frac{e}{W}\right)$$

where the first logarithmic factor measures the time to place the finger on the screen initially, while the second factor measures the time to position the cursor once the finger has been placed on the screen. D is the distance, measured in three dimensions, from the original hand location to the location of first contact. W is some measurement of target size. The rest of the constants, a, b, c, d, and e must be determined for each specific case.

If a succession of rapid tasks are performed (e. g. clicking one button and then clicking another), then D will be the distance between both buttons. Since in this formulation we have more than the original two freedom degrees, it might turn that it is easier to fit in a regression curve.

As noted previously, the validity of the formulation is constrained to the datasets on which it was proven. This means, that we cannot simply extrapolate. This is especially important since the experiments in this case were performed in quite early hardware, with much lower precision than current capacitive screens commonly available for most hand held devices. In the case of Sears and Shneiderman, the authors had to implement a stabilization software that was able to reliably yield a single touch position (though this did not mean that the position resolution detection was even closer to the most accurate mouse interaction, in part due to the resolution of the screen, and in part to the recognition software).

4.4 Fitts' Law in keyboards

The primary method for textual data entry is the keyboard. Keyboards have large history and have even been criticized devices. The average ratio of an office worker is roughly 50 words per minute, although higher rates (up to 150 words per minute) can be achieved. Although regular computer keyboards only allow a key press at a time, some specialized devices, such as the chord keyboards, allow multiple key presses at a time. These may achieve 300 words per minute, but require lengthy training to retain the complex pattern of chord presses.

4.4.1 Keyboard layouts

The typewriter was built in different ways by the middle of the nineteenth century. Several ways to put the paper and several designs for the keys failed. The initial successful proposal was due to Christopher Latham Shole. His design was successful in part because the intelligent placement of the keys facilitated the reduction in key jams. The original layout was called QWERTY due to the keys arrangement in the top left part of the keyboard. This has been the dominant keyboard layout since then. In Europe, some countries use small variations over these layouts, such as the AZERTY layout used in French keyboards or the QWERTZ versions used in Germany (see Figure 6).

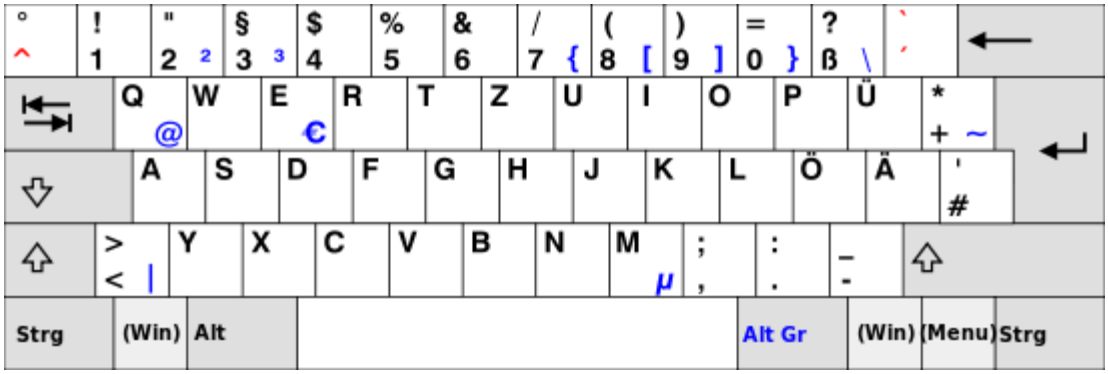

Figure 6: QWERTZ layout used in German keyboards.

By the 20s the Dvorak layout was developed. It was a new arrangement that supposedly reduces finger travel distances by at least one order of magnitude (shown in Figure 7), thereby increasing the number of words per minute approximately a 30% (some other researchers talk of 5-10% of improvement [Norman82]), and reducing errors. Although with a number of devotees, the acceptance of this layout has been low.

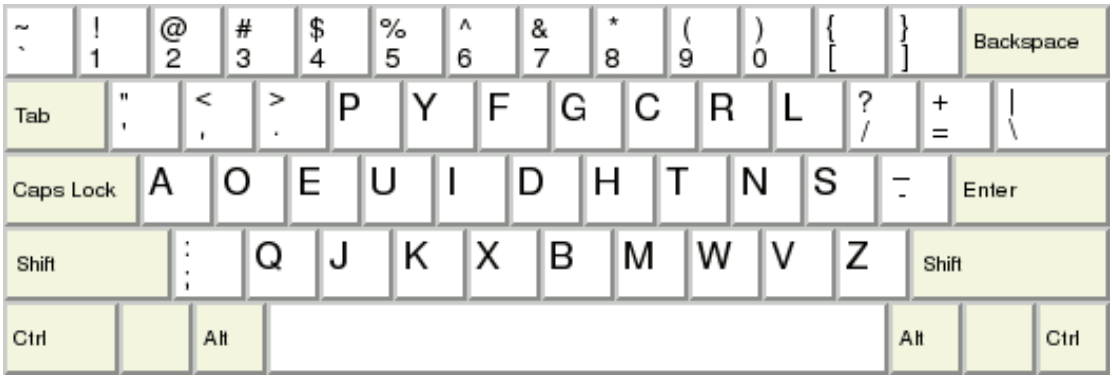

Figure 7: Dvorak layout.

Ideally, as shown by different experiments, for optimum typing speed, keyboard arrangements should be designed so that:

- The loads on the right and left hands should be equalized: both hands do the same amount of work.
- The load on the home (middle) row should be maximized: thus reducing the displacements.
- The frequency of alternating hand sequences should be maximized: this way one may maximize the frequency of typing because the fingers alternate and therefore one does not need to wait for the end of the movement of the first finger before starting the second movement.
- The frequency of same finger typing should be minimized: this may avoid fatigue.

If we analyze the Dvorak layout under these variables, it does a good job, especially on the first two elements: 67% of the typing is done on the home row, and the left-right hand balance is 47-53%. Although the QWERTY keyboard fails at these two conditions, (most of the typing is done on the top row and the balance between the two hands is 57-43%), the policy to put successively typed keys as far apart as possible favors the third condition, thus leading to relatively rapid typing. As a consequence QWERTY layout is superior to other arrangements such as the alphabetical ones. And it is roughly only 5 to 10% (though some reports raise this up to 30%) slower than DVORAK layout.

Note that this is only valid for 10-finger typing. Using this technique, the fingers are placed at fixed positions on the keys (A, O, E, U – for left hand, and H, T, N, S – for the right one), and these makes that very common letters such as vowels, require less effort than other, more uncommon letters. However, different layouts must be evaluated differently if we assume one or two finger typing. As a consequence, the comparison is not quite simple, and the distances the finger must cover may increase in the DVORAK layout in the case of one-finger typing.

4.4.2 Why experimenting with keyboard layouts is difficult

As we have commented many times, the experimentation is a difficult task. You can only assert conclusions on the exact parameters of the experiment. Testing with keyboard layouts adds a second difficulty here: users get their proficiency from practice. Therefore, in order to be able to perform a proper testing, users would require months of training in

any layout we can imagine is the best for typing before one can conduct the user study [Norman82]. Moreover, since the layouts are artificial in some sense, the same people would require to be trained back to the original (e. g. QWERTY) arrangement they use such as to be able to use normal computers.

As a consequence, it is commonly accepted that formal results are yield by using a predictive human performance model rather than user testing for evaluation [Lewis99].

4.4.3 Touch-Screen Keyboards

Modern multi-touch-screens enable text entry that is adequate for mobile interaction, thus bringing touch-screen keyboards to the mainstream. There are many problems when using touchscreens as input method, and several decisions must be taken, such as the keys sizes.

In practice, key size is usually determined by the screen size and its orientation, so the designers try to take advantage of all the possible space, while keeping some room for the display of the entered text. This has a major impact, since the available room is usually not enough: An iPhone 4 character in portrait mode measures 4 × 5.9 mm, while the Apple Guidelines for user interface design recommend that the minimum clickable size to be 6.85 × 6.85 mm.

Virtual keyboards also require significant visual attention because the user must look at the screen to press the correct key. Since there is no rest position such as in regular keyboards (which also increases the effort and results in a more fatiguing experience), the user does not know where his or her hands are with respect to the keys. Moreover, there is no physical feedback on the user's actions. Therefore, we can only ensure the key typing by looking at the keyboard and/or the rest of the screen, where the letter is inserted. For larger form-factors, this is especially problematic, due to the relatively large distance from the insertion point to the place where the virtual keys appear.

A second problem is the insertion point: since the screen is touchable, if the user accidentally taps on some part of the screen, the cursor may be changed and the following text ends up being inserted mistakenly at an unintended location.

Some attempts to improve the feedback go from audible "clicks", to a tactile screen that actually requires pressing on it to effectively enter a key. Its benefits are not clear. For example, the pressing screen, present in mobile devices such as the BlackBerry Storm requires relatively large amount of pressure, and its original layout also lost some precision on the corners (which was improved on the Storm 2 version).

Another big disadvantage of virtual keyboards is that they occlude an important portion of the screen, resulting in less space for the document we are editing or the form we are filling. This is a clear problem when talking of smartphones, and only partially alleviated in tablets.

Some more innovative designs combine the power of touch and stylus-based typing with stroke gestures. They have been shown to produce high rates of text entry, once the user masters them, although it remains unclear if such approaches will achieve widespread adoption.

Finger touch in tactile screens also follows Fitts' Law. Once the users have reached expertise, the time to move the tapping device with a single finger from one key (i) to another (j) depends basically on the distance and key width of the keys, following this equation [Bi2013]:

$$MT_{ij} = a + b \log\left(\frac{D_{ij}}{W_{ij}} + 1\right)$$

Where D_{ij} is the distance between keys i and j, and W_{ij} is the width of each key. They further refine using the effective width of the key, instead of the nominal value, but a deeper analysis goes beyond the needs for our course.

4.4.4 Digram-based keyboards for single-finger typing

As already noted before, the design of an efficient typing keyboard for single-finger typing should minimize the overall displacement of the finger for the most common words. This is what DVORAK actually does for 10-finger typing, but this is not a current typing method in smartphones or tablets. Commonly, one hand is devoted to hold the device and the user types with the other.

Several attempts have shown their proficiency, but usually as a matter of good prediction + excellent error correction together with some tricks to accelerate input (see for example Research In Motion's new keyboard for the Blackberry 10 mobile OS [RIM2012]). However, most of the keyboards still rely on the QWERTY layout, with a notable exception that has attracted a lot of attention: the Minuum keyboard by Whirlscape ([Whirlscape2014]), which uses the QWERTY layout but in a single row (plus a strong word prediction and correction).

For hand-held tablet and portable computers (including pen-based systems), it is important to evaluate the best arrangement of keys for typing layouts when users type with one finger or a stylus. A nonstandard typing-key layout (in a roughly 5 x 5 key matrix) based on digram Predictive human performance models for 10-finger keyboards use the frequency matrix of English-language digrams (pairs of letters that commonly appear together) and a matrix of empirically derived interkey typing times [Lewis99].

To improve the layout for single-fingered typing, it is reasonable to analyze the language digrams to determine the relative associative strengths among the letters so the layout can minimize the distance between strongly associated letters.

This may lead to an arrangement such as the one in Figure 20, where those distances are minimized and thus the predicted improvement raises to approximately 25 words per minute (over the typical 20 words per minute on a complete QWERTY layout and one single finger typing). The authors claim that, after the corresponding training, such a layout should be about 27% better than the QWERTY layout. And even, an alphabetic typing-key arrangement, again in a roughly 5 x 5 key matrix, should be about 13% better than the QWERTY layout for single-finger entry.

```
  Q R W X Y
  L U A O F
Z T H E N G
  V D I S P
  B C M J K
```

Figure 8: Digram-based layout for single-finger typing.

4.4.5 Swipe-based keyboards

Entering text has always been painful in the mobile space. There are many issues that affect the text entry, especially on mobile. To name a few:

- Size of the keyboards: As already noted, the keys' size is smaller than most UX guidelines recommend, especially in smartphones. As a consequence, the size of the keyboard is prone to errors in typing.
- One hand vs two hand entry: When having both thumbs free, we may probably write better, but it is quite common to have to type with one single finger, which, again, is more prone to errors because the hand location is more variable (than two fingers with the rest of the hand in a fixed position). But even in those cases, the number of errors is quite larger than with a regular keyboard.

There is a tendency in mobile space to provide different keyboard interaction techniques for accelerating text entry. Sometimes this acceleration is real, while sometimes it is only felt by the users [Nguyen2012].

Gesture typing basically consists on using a single trace that goes from the first letter of the word until the last one without lifting off the finger, and by moving it on the screen over all the letters of the word. This is illustrated in Figure 9.

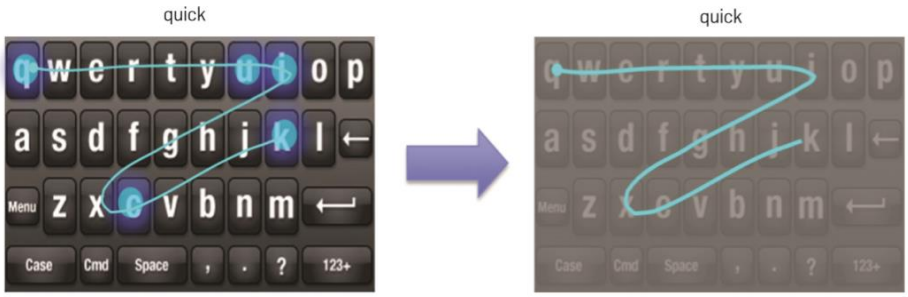

Figure 9: Illustration of the gesture keyboarding for entering the word "quick" in an application for gesture-based typing in mobile devices.

One of such tendencies is the use of gestures (swipes) to write a whole word [Kristensson2012, Zhai2012]. Nguyen *et al.* [Nguyen2012] have studied the effect in tablets, where they found that regular typing and gesture typing perform at similar speeds, however, there is a higher user satisfaction ratio when using one of the concrete applications that provide gesture typing (Swype) than regular keying.

Note that all those approaches are commonly executed single handed and in a QWERTY keyboard, which, as we saw before, do not especially reduce the distances the finger has to travel to edit words.

Among the other different improvements, a notable one is a two finger gesture keyboard [Bi2012] which showed that the finger travelling was shortened about the 50% but the speed does not increase over single finger entry, most probably due to the high demand in attention to coordinate both hands. In Figure 10 we can see two proposals for writing "interaction" in such keyboard: one using continuous tracing (right), and the other by interrupting the trace between hands' change (left).

Figure 10: Two different approaches for a bimanual gesture-based input of the word "interaction".

4.5. Evidences of Fitts law

The Fitts law has been used for successfully modeling many UI access tasks. However, when assessing the validity of such law, we must be very careful on the analysis of the experiment. Experiments cannot cover all the variety of users and tasks, and therefore are constrained to a set of configurations that may be not very large (e. g. 4 different target sizes and 4 different distances). As a consequence, although Fitts' law has proven its validity in many experiments, we may not freely extrapolate (e. g. assuming it works for elder people or children…) for all sizes and distances. More concretely, as we have already seen, there are limits of the validity of the Fitts' Law: for very small targets, for instance, the MT function regression curve starts to be less aligned with the resulting values.

Apart from the concrete limitations given by the experimental setup, Fitts' Law has shown its validity in multiple setups and devices, which go from mouse, joystick, finger, stylus… and different screen types of varying sizes ([Chapuis2011]).

An interesting observation that has been analyzed in some experiments is the previous knowledge by the users of the targets' sizes and positions. It is an important issue, because users who select objects in point-and-click interfaces quite often know these features of the targets. Studies show that for precued targets the preparations lead to more efficient and precise pointing movements than for non-precued targets [Hertzum2013]. Target precuing also interacts with pointing device, distance to target, and target size, but not with user age. In particular, the benefit of precuing is larger for the mouse than the touchpad, suggesting that the movement preparations users are able to make on the basis of precues depend on how demanding the pointing device is to use.

It is curious to note that, as Balakrishnan and MacKenzie found [Balakrishnan97], the index finger alone does not perform as well as the wrist or forearm in pointing tasks, but the thumb and index fingers in coordination outperform all above cases.

4.6 Fitts variants and limitations

Note that Fitts' law models properly adult users, but children behave differently. Some results show that for point and click tasks, Fitts' law adequately predicts the first time the children enter a target, but not the time of the final selection.

Some researchers suggest measuring the ID using the *effective target width* (or W_e), instead of the *nominal width*. This arises from the observation that users do not adjust their performance as much as might be expected when target width is changed. More concretely, when changing target width, the changes in endpoint variability are typically much smaller would be expected ([Fitts64, Sourkoreff2004]).

Therefore, this formula, commonly called ID_e is written as:

$$ID_e = log_2\left(\frac{W}{W_e} + 1\right)$$

is the one that is suggested also in the ISO 9241-9 ([ISO2000]). However, there is quite a large controversy on its use. There are two main reasons for that:

1. The data does not always fit better than when using *ID*.
2. The effective width is the result of the actions of the users. As a result, it cannot be measured previously, and therefore is not useful for design without experimentation.

The first problem is still in debate. Wright and Lee [Wright2013] performed a set of studies and found that "it has been shown to compensate for the differences in effective target width that are present across conditions and participants". However, they do not recommend its use without much care since the obtained gain may not compensate for the extra effort to generate such data. Moreover, they believe that when those values are not available, it is safe enough to proceed using *ID*.

Zhai *et al.* [Zhai2004] believe that both methods (regressions from uncorrected widths and from effective widths) are valid, but the appropriate method depends on the goal [Chapuis2011]:

- Using uncorrected widths seems useful to reliably assess actual user pointing times in user interfaces, accounting for natural biases in target utilization, and to assess errors separately.
- Using effective widths instead usually deteriorates the fit but is strongly recommended when one needs to interpret the constants *a* and *b*, for example, in order to compare the performance of different input devices.

5 Applications of Fitts' Law in interface design

Fitts' Law describes the Movement Time in terms of one dimensional displacement, as we have already seen.

5.1 Use of screen edges

There are several interfaces of Operative Systems that put some of the graphical elements of the UI next to the edges of the screen. This is true, for instance in the Mac OS system (see Figure 11), where the top menu is attached to the top edge of the screen. Since there is no distance from the top edge to the menu, the mouse movement required to select any of the options of the menu does not require precision in the Y direction (while it requires in the X axis). As a consequence, the selection process may benefit because the movement can be done fast (using mouse acceleration techniques that will be visited later).

Figure 11: The Mac OS X operative system has a menu bar on top of the screen.

From the point of view of Fitts' Law, this kind of behavior can be interpreted as having targets with virtual infinite length in one dimension. This, at its time, reduces the ID of the target acquisition and therefore intuitively makes the movement simpler/faster to achieve (see Figure 12).

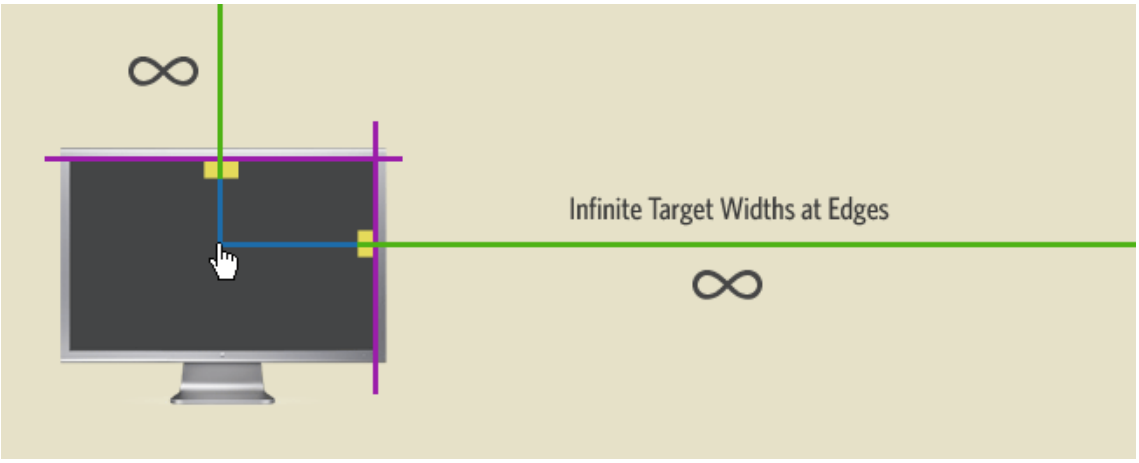

Figure 12: Interpretation of edge elements according to Fitts' Law: They can be considered as having infinite width.

With this in mind, it turns out that corners are the most accessible places in such an UI (see Figure 13), because they have virtual infinite dimensions.

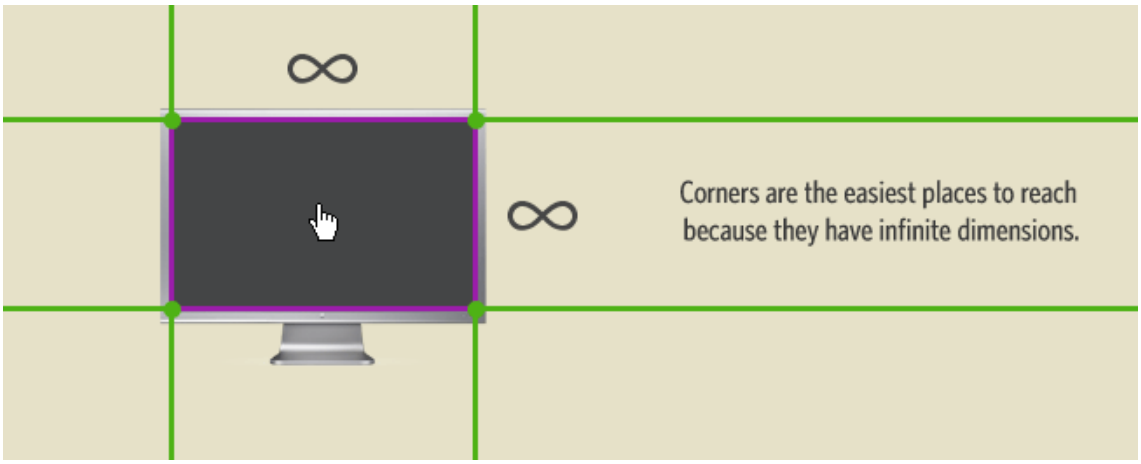

Figure 13 Elements at the corners are the most accessible because they do not require neither horizontal nor vertical precision in one direction.

We may compare this to the typical interface that some programs have presented, such as most Windows applications (in Figure 14 we can see a snapshot of the old Microsoft Word 2003), where the top items were slightly displaced from the top of the screen. We can intuitively see that an extra effort will be necessary for properly placing the pointer in any of the menu titles.

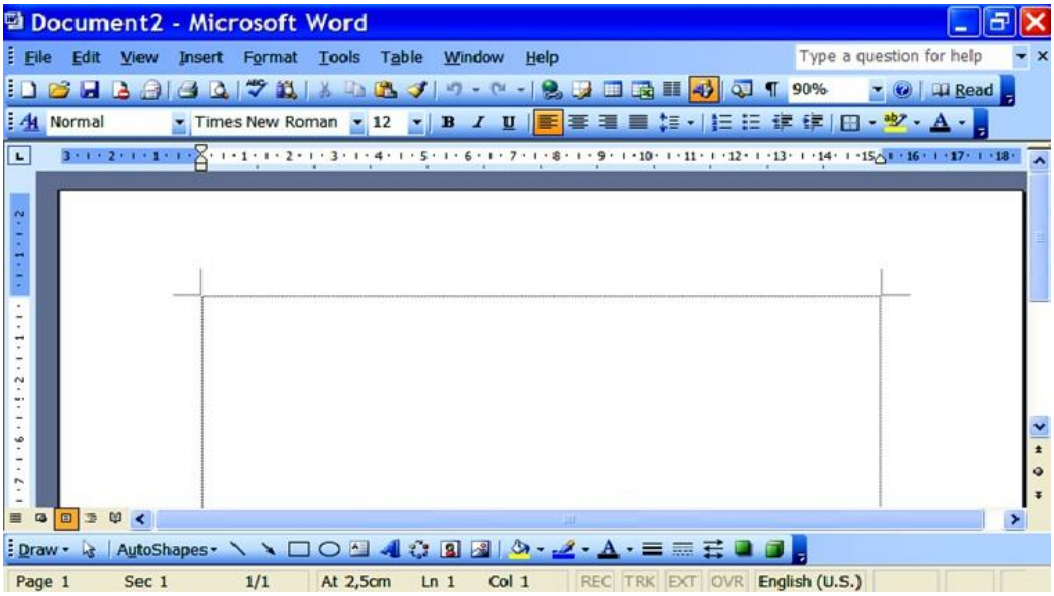

Figure 14: Elements of the menu in Microsoft Word 2003 are not placed at the edge, and therefore require more effort (since their ID is larger) than if they were placed right next to the edge.

5.2 Placing related things close

The use of Fitts' Law can go even further than that. By placing elements that are commonly used together, we are going to facilitate the work of the user. We can see an example in Figure 15, where we compare the Mac OSX scrolling bar with that of Windows.

By analyzing ID we may deduce that the Mac OSX scrolling bar may be faster to use than the Windows one. There are two reasons for this:

- The slider is larger and thus easier to reach.
- The direction buttons are together, so if we need to change the direction, we will need to perform a smaller movement of the mouse.

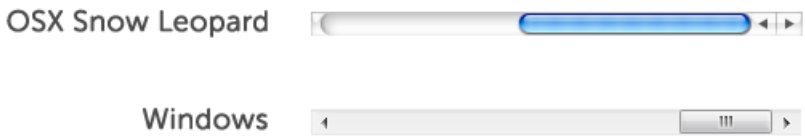

Figure 15: Mac OSX vs Windows scrolling bars. The Mac OSX is theoretically faster because it is easier to reach (its slider is larger) and changing the scrolling direction only requires to move the mouse slightly. In the case of the Windows scroll bar, if we want to change the scrolling direction, we need to traverse the whole bar.

5.3 Complicate the access to elements

In the case of elements that are not supposed to be clicked together, or the elements that may be dangerous to confound, it may be useful to do the contrary. That is, separating those elements may avoid mistakes.

Figure 16 shows an image where two buttons that perform very different tasks are placed next to each other, and therefore the user might accidentally chose the wrong one.

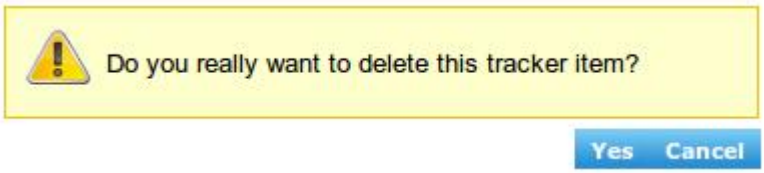

Figure 16: Buttons doing very different tasks and that are placed together may induce errors. In this case, buttons *Yes* and *Cancel* have opposite meanings. Moreover, there is not a clear separation on both, which is a second mistake. They should be placed at separate positions.

5.4 Pie menus

Another not so popular types of menus are circular and semi-circular (or pie) menus (see Figure 17). These menus appear around the contact point of the finger or the clicking point of the mouse. The distance from each menu item to the cursor is constant and very small.

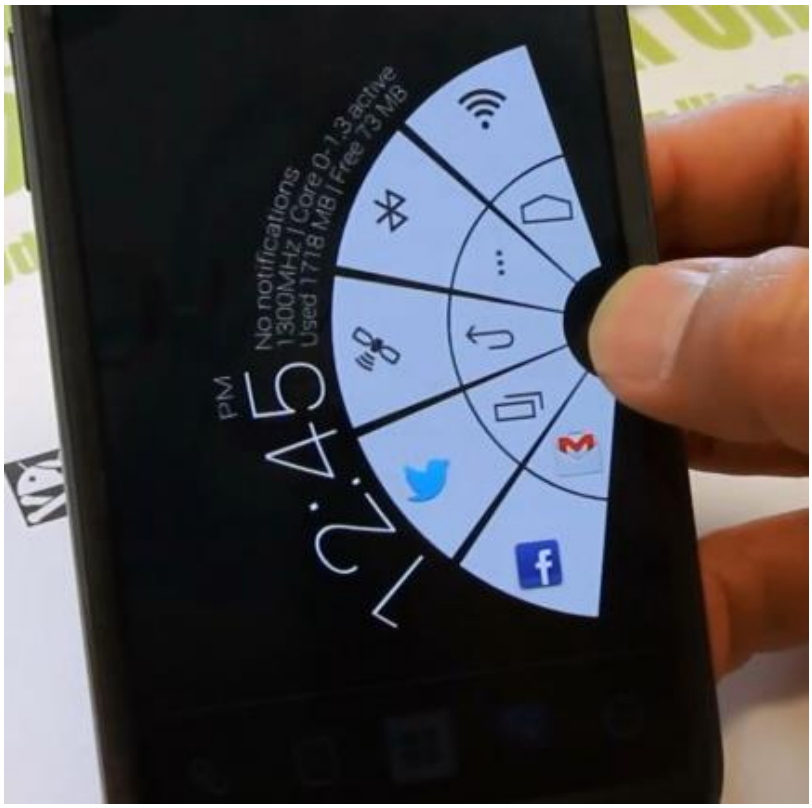

Figure 17: Pie menus rely on the fact that the items can be placed at the same distance from the pointer. When the number of items is high, several layers of menus are required.

The idea is quite interesting. However it requires some elements to make it usable:

- First, the menu has to be created on demand, on a certain position, that leaves room for the whole menu to appear. This turns this kind of menu into a contextual one, with its advantages (it may vary its contents in function of the position) and shortcomings (it will be used only by expert users).
- Second, for the menu to be practical, it may have not occlusions (the original pie menus are circular, which is something that will necessarily be partially occluded in a tactile device), and the elements must be sorted in such a way that the access is simple.

The original first pie menus where circular (see Figure 18) since they were developed for desktop systems. In such systems, the pointer (mouse) does not occlude the menu when it appears (since the menu is all around the pointer representation).

Figure 18: Circular menu with all the elements at the same distance to the pointer.

6 Law of crossing

6.1 Introduction

Apart from simply pointing or clicking, the use of stylus or analogous to stylus devices (e. g. finger) in tactile surfaces naturally leads to other quite intuitive tasks such as crossing elements (see Figure 19). Accot and Zhai ([Accot97, Accot99]) and Zhai *et al.* [Zhai2002] demonstrated that there exists the so-called *Law of crossing*. It models the time to move a cursor or a stylus across two goals. Moreover, the Law of crossing follows the same characterization than the Fitts' Law:

$$T = a + b\ log_2\left(\frac{D}{W} + 1\right)$$

Where T is the average moving time between passing the two goals. D is the distance between the two goals and W is the width of each goal. Like in the previous case, a and b are constants to be determined.

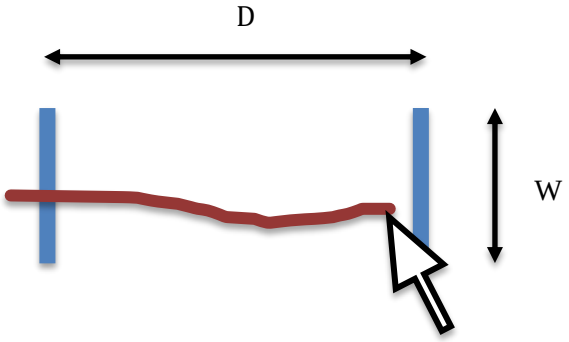

Figure 19: Crossing movement with the pointer.

The crossing action is well related to the pointing action described by Fitts' Law as we commonly interpret it in a graphical user interface with buttons. The different elements are compared in Figure 20.

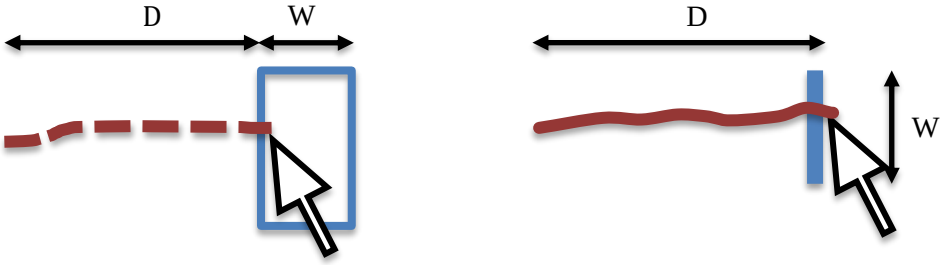

Figure 20: Pointing vs crossing: the interaction techniques differ in where the precision is required: in the direction of the movement (such as in the crossing interface – right – where the width determines how far we may move from the initial direction) or in the termination point (when pointing – left – the limit is bound by the width of the target).

The way to analyze this kind of interaction is not simple, since it can be performed in very different ways. Hence, many variables come here into play:

- **Discreteness vs continuity of the movement:** In some cases it is necessary to land the stylus before crossing the first target and lifted off after crossing it, and the same for the second objective, while the interaction can also be done without lifting off the stylus (see Figure 21).

- **Direction of the targets vs direction of the movement:** Targets may be aligned in the same direction of the movement or orthogonal to it. In the first case, a line may cross many targets while in the second case, a different trace will be necessary (shown in Figure 22).

Note that, since the so-called *Law of Crossing* takes the same form as the Fitts' Law, it is important to try to isolate such constants in order to be able to determine whether crossing is better than pointing.

6.2 Crossing configurations

In contrast to pointing, the crossing technique and targets can be configured in different ways (e. g. is the target to cross orthogonal to the direction of movement?). Thus, for the sake of completeness, the performance of crossing must be evaluated in each of the configurations. The different varying variables include the direction of the movement vs the orientation of the target, and whether it is necessary or not to lift the pointer before crossing the target. We start discussing this last element.

6.2.1 Continuous vs Discrete

The implementation of the technique may be so that the pointer (stylus, finger…) can move in contact to the screen or it may be required to lift it up before crossing. These two configurations are called continuous and discrete, respectively, and are depicted in Figure 21.

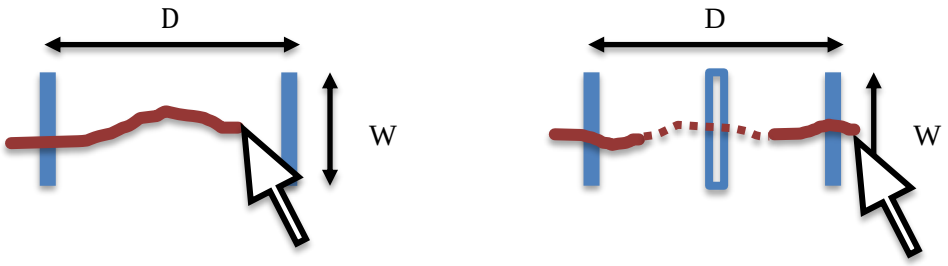

Figure 21: Continuous crossing (left) vs Discrete Crossing (right) interaction.

Intuitively, discrete is more costly than continuous, but the studies show that this is only more time consuming when the distance between the targets (and thus the obstacle) is small.

6.2.2 Collinear vs Orthogonal

The targets, both in pointing and in crossing, can have their width aligned or not with the direction of the movement. When the width *W* is orthogonal to the direction of movement, we have the configuration on the left in Figure 22, where the movement seems more simple and continuous, while when the objectives are not aligned with the direction of the movement, the pointer must trace a slightly more complicated path. This is also applicable to crossing targets with obstacles (discrete crossing).

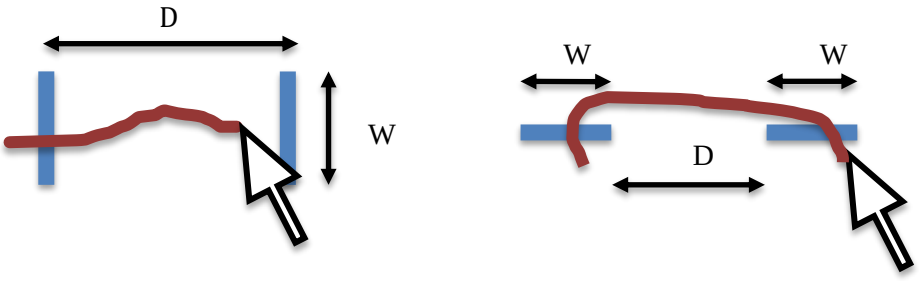

Figure 22: Collinear crossing (left) vs Orthogonal Crossing (right) interaction.

In order to analyze the performance of crossing, it is necessary to perform a factorial combination of the different configurations, that is, one should test continuous traces with orthogonal crossing, continuous tracing with targets in the same direction of the movement plus discrete tracing with these two configurations of the targets.

The studies by Apitz *et al.* [Apitz2010] compare the different configurations of the more classical pointing technique: collinear and orthogonal, with all the different configurations of the crossing method: the collinear discrete, collinear continuous, orthogonal discrete and orthogonal continuous. And the experiments are carried out for several different target sizes and distances. The different configuration scenarios are analyzed in relation to *ID* and error rate. The conclusions the researchers reach are, very briefly:

- Crossing-based interfaces achieve similar (or faster) times than pointing.
- The error rate in crossing is smaller than in pointing.
- Discrete crossing becomes more difficult if the distance between the targets is small.
- Crossing (especially continuous) seems superior than pointing for *ID* values > 5.

7 Law of Steering

7.1 Introduction

Another commonly necessary task for the interaction with modern user interfaces is the creation of trajectories. These tasks are useful for navigating through nested menus, 3D navigation, dragging elements, and other drawing tasks.
Very often, such tasks are directional movements with a lateral constraint (see Figure 23).

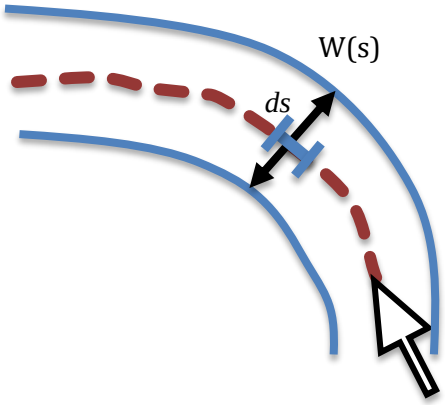

Figure 23: Generalized path steering with amplitude constraint.

Accot and Zhai found that the performance of manual steering tasks (T_s) for a generalized path C (see Figure 23) can be expressed in the following integral form:

$$T_s = a + b \int_C \frac{ds}{W(s)}$$

Where T is the time to successfully steer through the path *C*, *W(s)* is the path width at *s*. With this formulation, the Index of Difficulty of the steering task is

$$ID_s = \int_C \frac{ds}{W(s)}$$

Accot and Zhai arrived to this formulation by transforming the steering task into an accumulation of an infinite number of goal-crossing tasks ([Accot97]). The time to cross a goal of width *W* at distance *D* follows the Fitts' Law equations. This way, by transforming the goal crossing task into a succession of goal crossings along a trajectory, they can calculate the summed up index of difficulty of N consecutive goals as:

$$ID_N = N\ log_2\left(\frac{D}{NW} + 1\right)$$

Taking *N* to infinity yields the equation of ID_s.

7.2 Steering through straight paths

Using these results, we can calculate the time to steer through a straight path. The straight path can be determined using the *D* and *W* variables for the distance to trace and the width of the path, respectively (see Figure 24).

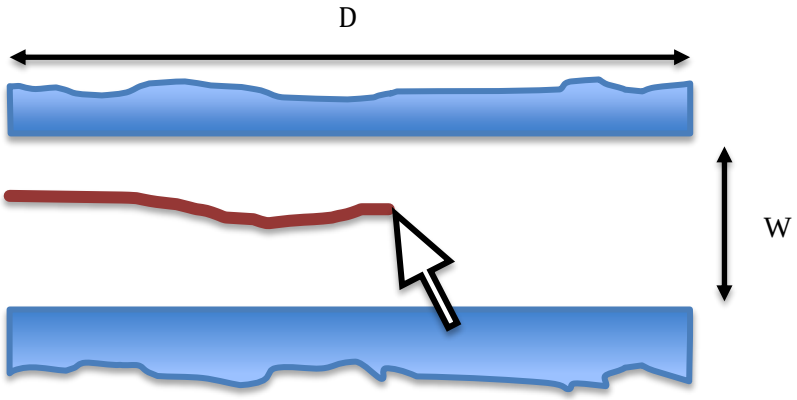

Figure 24: Path movement in a straight path with amplitude constraint.

For a straight path, the time required to successfully steer through it would be:

$$T_p = a + b\ \frac{D}{W}$$

Where *W* is the path width, and *D* the path length. The index of difficulty of the task will be then:

$$ID_p = \frac{D}{W}$$

The previous equation also applies to circular paths with constant width.

7.3 Evaluations of Steering Law

Accot and Zhai [Accot97] showed steering law works for paths of different shapes: a cone, a spiral shape, or a circle, and that it also works for different input control devices. The upper limit on the application of this law is the human body limitations: This means that we can change configuration parameters so that the theoretical speed can be increased, but if this exceeds human body capacities, the results saturate.

One of the direct applications of steering law is the design of nested menus. Since accessing an item of a submenu is addressed by performing a path that follows an 'L' shape, this path can be subdivided in two different steering paths, and therefore the access time can be calculated by directly applying the steering law. As a result, different menu configurations (e. g. classical nested menu vs pie or hierarchical pie menu) can be evaluated using this law.

A more complex interaction experiment was also developed by Zhai *et al*. [Zhai2004] to evaluate locomotion in a VR setup of 160 degrees of field of view. The users had to navigate through two types of path, one straight and the other circular. Graphics were rendered in real time in response to users' actions. The path was 1885 meters long. The users were instructed to drive as quickly as possible without going out of the path boundary.

Figure 25: The Virtual Reality simulator setup used in the steering experiment.

The user's behavior was measured every tenth of a second throughout the experiment. The recorded variables were: vehicle's X and Y coordinates, heading, gas pedal amplitude, brake pedal amplitude, steering wheel amplitude, speed, deviation from middle line, and collisions.

The users had a representation of the car also projected on the screen, to help them have references of their relative position with respect to the road. Some examples of the images seen by the users throughout the driving experiment are shown in Figure 26.

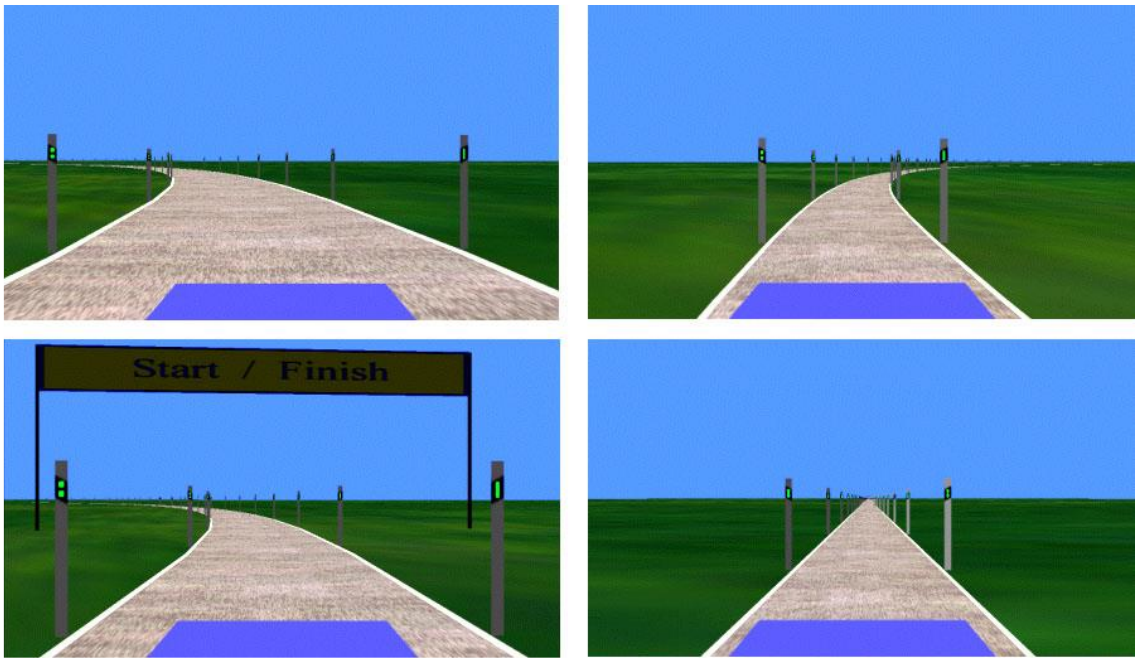

Figure 26: The images projected in the simulator, as seen by the users.

The results show that there is higher variance between users than in other experiments where no so complex tasks are performed. For example, some users used the total power of the gas pedal, unlike in the pilot experiment. The results of those users were eliminated from the analysis. Otherwise the results would not be comparable.

Results show that the steering law behaves well for locomotion, although with a lower dependency on the path width, since driving involves stronger dynamics than simple gestures such as hand drawing. In any case, the authors found that there is a linear relationship between the trial completion times versus the index of difficulty of the path, with a high fitness value.

8 Accelerating target acquisition

8.1 Introduction

Besides the uses of Fitts' Law in the design of User Interfaces, commonly the screen space is restricted enough to avoid decisions that imply the use of larger targets or cluttered screens to accelerate target acquisition. This, together with an improvement in the computation power, has led to the use of techniques that modify such parameters dynamically.

Commonly, the modifications that are performed are the same than we have seen for static UIs: amplitude reduction and increase of target size. However, we may change these parameters a non-constant value. Moreover, we may add to this the modification of the speed of the pointer, the so-called dynamic control-display ratio change.

8.2 Modification of Target Width

In order to reach and manipulate virtual objects, applications provide a pointer together with the virtual representations of the elements to manage.

8.2.1 Target and screen size

As stated by the Fitts' Law, the size of the target we are trying to reach has a great influence on the time required to select it. Actually, the cost or effort to reach an object depends on the amplitude of the movement and the size of the target. The larger the target, the easier it is to select, but the longer the distance to cover, the more difficult it is to reach.

There is a tight relation between screen size and target size, the larger the screen, the larger the targets can be. However, since the amount of information we want to include in our windows is also relatively large, we have to balance between information and controls.

Although nowadays screen resolutions often leave enough room for the design of large target icons, the fact that screens are large also make that the distance the user has to cover can also be large. Therefore, according to Fitts' Law, the cost of reaching the element also increases with the distance.

On the other hand, the resolutions of portable devices, although being rather high (such as 900x600 pixels), the size is small (around 4'' on a modern smartphone), and therefore, the number of pixels that has to be devoted to a single icon is relatively high, thus making the screen *effectively smaller*. There is a second issue: The fact that the pointer in most modern screens is a finger. Hence, its size relative to the size of the screen makes the display also *effectively smaller*.

With respect to Fitts' law, there are two simple ways to reduce the difficulty of a pointing task: enlarging the target or moving it closer to the cursor. Both have been explored in several ways. These are also coupled with the velocity of the cursor. We will deal with this issue, named Control-Display ratio later. Now, we introduce some techniques that try to reduce interaction times by manipulating the target sizes dynamically.

8.2.2 Expanding targets

One approach to easing acquisition of small targets consists in enlarging the target size when the cursor approaches the element to be selected. This technique is called expanding targets [McGuffin2002] and has been implemented in different ways. Mac OSX operative system uses this technique on the icon panel that serves as program launcher, also known as Dock, shown in Figure 27.

Figure 27: Mac OSX Dock with its size-enlargement and position-changing icons.

Note that this is a mix on two techniques: target size increase and moving target. The system changes the size and moves the position of the elements in the Dock in relation to the position of the cursor. Therefore, the *selectable* area is smaller than it seems to be, because when approaching the center of the icon (in a horizontal move), the icon itself moves towards the opposite direction of the cursor movement. This has been reported to cause frustration on the users because the expansion induces these movements that may be not totally predictable if the cursor approaches the Dock in a perpendicular direction. This can be alleviated if the system allows the enlarged icons to overlap over the neighbors. A second method that reduces frustration is expanding the icons only when the cursor velocity decreases on final target approach.

8.2.3 Bubble targets

Other not so successful methods for target expansion have been tested, such as the *bubble targets* [Cockburn2003]. These add a selection region around the target with a bubble shape (Figure 28).

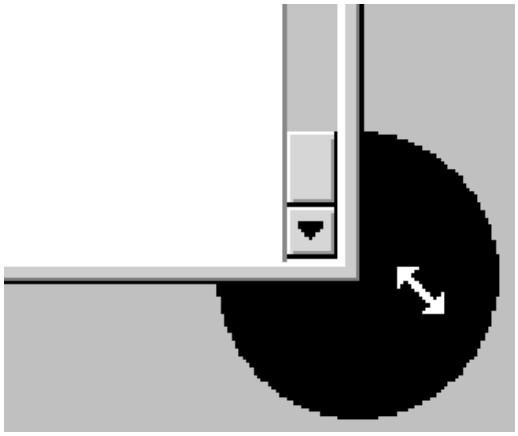

Figure 28: Bubble-based targets for easier selection.

The bubbles only appear when the mouse is pretty close to the selectable point. This technique improves the speed of selection of small targets, but some users have reported distraction due to the appearing of the bubbles.

8.3. Increasing cursor size

Some authors address the same problem by changing the area that is affected by the cursor, instead of modifying the target size. This leads to an effective reduction of amplitude movement. But it can also be seen as a magnification of the target size.

Grossman and Balakrishnan [Grossman2005] increase the size of the cursor in a circular region that embraces the closer target for a more comfortable and easy selection. They call this technique the *Bubble Cursor*. The cursor's size is dynamically adjusted so that only a single target is enclosed by it. Moreover, its shape also is modified to envelop the closer target when it is not completely inside the main cursor bubble. In Figure 29 we can see a sketch of such technique.

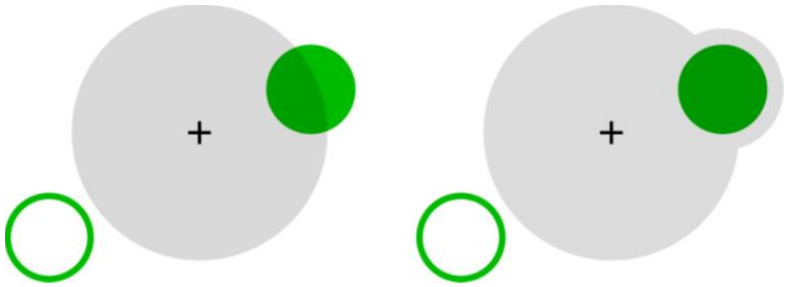

Figure 29: Bubble cursor adapting its size to cover the closer target.

Their implementation is quite straightforward, since they divide the space of targets using a Voronoi diagram where each region contains a single target and the effective with of the cursor is modified to envelope the target contained in that region (see Figure 30).

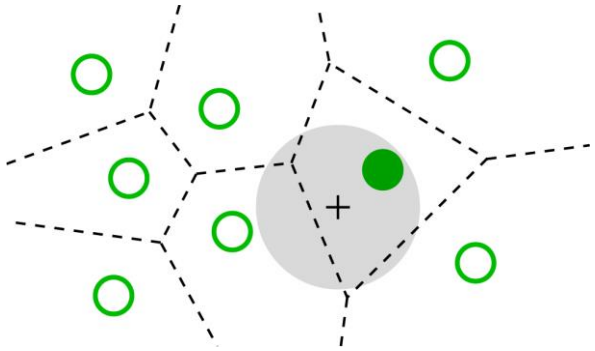

Figure 30: Voronoi subdivision of the target space. When the cursor enters in one of the regions of the target Voronoi map, it may adapt its size to envelop only the target of interest.

This technique has been proven very useful and fast. However, the experiments were performed using a zero mouse acceleration value. Chapuis *et al*. [Chapuis2009] improve on this technique by taking into account also the speed of the mouse. The technique they develop is called *DynaSpot*. In this case, since the area of the cursor grows with the speed, it is likely that the cursor area will overlap several targets at the same time. However, only the closer to the cursor center is the one candidate to selection. We can see in Figure 31 how the area is changed according to the speed of the cursor.

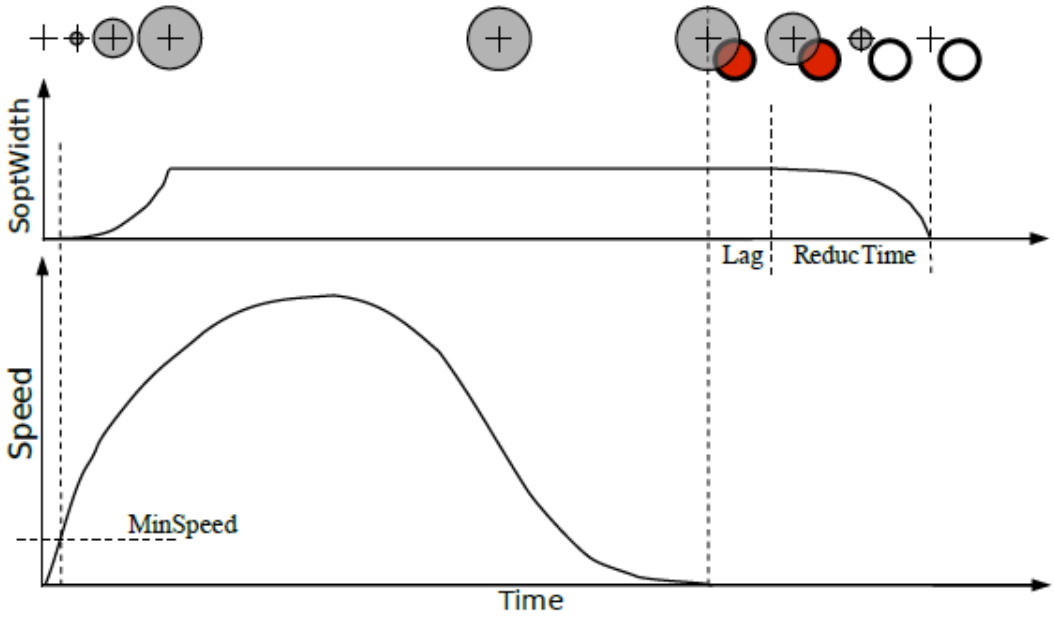

Figure 31: Area change of the cursor as a function of the mouse speed.

Like in the previous case, visual cues are provided to inform the user which are the targets that are going to be selected as well as the current size of the area cursor. This is shown in Figure 32.

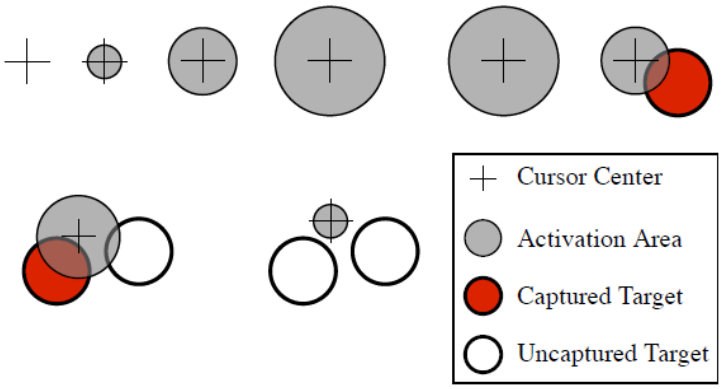

Figure 32: Indication of the target to be selected (the closer to the center of the cursor).

Chapuis *et al.* [Chapuis2009] also compare this method with the Bubble Cursor and find that this new method obtains better results both in acquisition time and reduction of number of errors.

8.4 Target moving and other techniques

8.4.1 Target moving

Besides enlarging targets, there is another popular possibility that enhances selection and consists in moving the targets to the cursor.

The method used by the Mac OSX Dock combines target enlargement and target movement, however the movement is relatively small compared to other techniques.

Probably the most widely used direct application of target movement is contextual pop-up menus. Contextual menus are the ones that appear at the cursor location after some user action (such as selection and/or right click). This makes the distances to the items to be minimal, as compared to visiting the classical top menus. Moreover, as compared to the classical pop-down menus, the contextual menus can adapt their contents to the previous action: For instance, if the user selected a word, several options related to word formatting can appear, which may differ on the possible actions if no region or word is selected.

Another not so popular types of menus are circular and semi-circular (or pie) menus (see Figure 33). These menus appear around the contact point of the finger or the clicking point of the mouse. The distance from each menu item to the cursor is constant and very small.

Figure 33: Semi-circular menu on Android's Honeycomb version of the browser.

8.4.2 Other techniques

Other experiments have shown that the selection can also be helped by the *sticky targets* technique. The main idea is to convert the selectable points as *attractors* of the cursor. Therefore, when the cursor is close to the selectable point, it moves to it. This method improves efficiency and has been implemented both in 2D and 3D user interfaces. Experiments have shown that users adapted easily to this technique and were preferred to other methods such as the bubble targets.

This technique has also been implemented in 3D virtual environments by Andújar and Argelaguet [Andujar2007], by modifying the selection ray direction to point to the target it is getting close (see Figure 34).

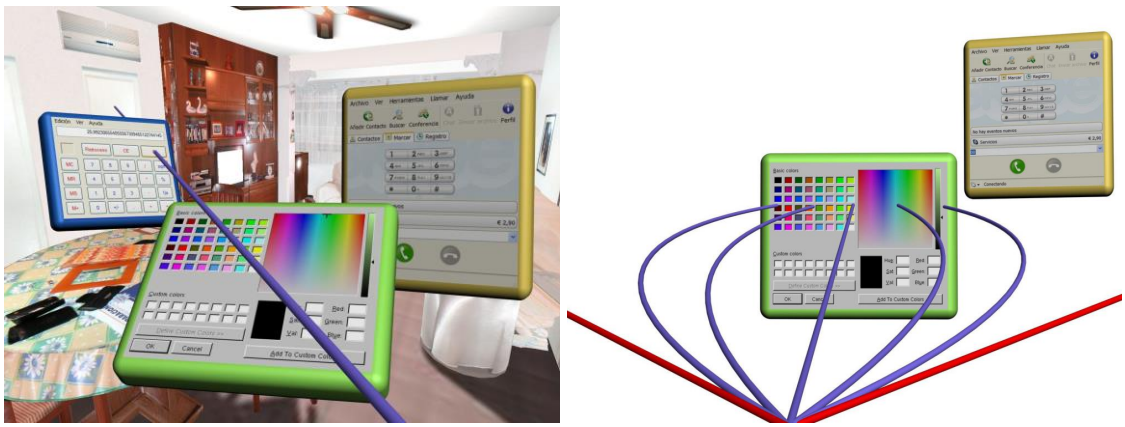

Figure 34: Ray direction modification in a 3D Virtual Reality environment to point to the closer target.

8.5 Discussion

Comparisons with other techniques show that target resizing facilitates pointing even if the expansion is late and unpredictable. The main problem of such techniques is that in order to expand a target if other targets surround it, those must be shrunk. As a consequence, some experiments showed that no performance improvement is obtained for systems like the Mac OSX Dock. More generally, techniques that dynamically change the screen layout cause a spatial disorganization that may limit their expected benefits.

In other environments, where the scene may be cluttered, and a lot of selectable targets may be placed in a small region of space, such as in many 3D virtual reality scenarios, a combination of techniques may be useful. Argelaguet and Andújar [Argelaguet2008] combine expanding targets with clipping away other closer candidate targets with a technique they call *forced discocclusion*.

8.6 Control Display Ratio

8.6.1 Concept

Control Display ratio is the relation between the amplitude of movements of the user's real hand and the amplitude of movements of the virtual cursor. More specifically, the Control Display Ratio usually refers to the distance the mouse has to cover in the physical world to move the pointer on the screen by a given distance. These are usually measured in meters (physical move) and pixels (cursor movement).

Obviously, any control system must translate physical displacements of the interaction devices to virtual movements of the cursor. The way this translation is implemented may affect the performance of the user.

8.6.2 Control-Display Ratio adaptation techniques

There is no clear idea on how the definition of C-D ratios affects our perception of the virtual world and therefore improves productivity by reducing selection times. There are three typical C-D ratio configurations:

- Constant
- Dependent on mouse speed
- Dependent on the cursor position

The idea behind the *mouse acceleration* is to provide easier ways to access further targets. The cursor moves by a larger distance when the mouse covers a given amplitude more quickly, capturing an intention: when users move the physical device quickly, they typically wish to go further, so the cursor can be displaced even faster to cover the distance more quickly.

An opposite effect can be mouse *slowing* when the cursor is near a target: covering the same number of pixels requires moving the mouse by a longer displacement. The idea here is to improve pointing precision.

Although some studies conclude that nonlinear mappings are not intuitive enough to provide positive improvements for pointer movements, other studies have shown increase

in performance for 3D navigation, 3D rotations and other pointing problems. Therefore, there are no definitive results.

Compared to the previous approaches that address target magnification, C-D ratio adaptation can also be interpreted as dynamic magnification of the physical motor space where the mouse movements take place. In this sense, these techniques are related to zoomable interfaces that use a local or global magnification of the visual space.

Pointer acceleration is the default behavior on the Microsoft Windows, Linux, and Apple Mac OS X operating systems. The implementation in those systems dynamically changes the C-D ratio the following way: When the speed of the control device is high, CD gain is high (typically well above 1) and when the control device moves slowly, the CD gain is low (in some cases less than 1). In some Operative Systems, this behavior can be configured, as it is also related to other accessibility options (e. g. large fonts and icons for people with vision problems…).

9 Pointing tasks

Complex information displays generate the necessity of pointing and selecting items. It is a fundamental and frequent task in graphical user interfaces, so even a marginal improvement in pointing performance can have a large effect on a user's productivity. This direct-manipulation approach is attractive because it prevents the users from learning commands. It has other advantages such as reducing the chance of typos and keeps the attention of users on the display. As a result, pointing and selecting often results in faster interaction, fewer errors, easier learning, and higher satisfaction.

Pointing, however, does not come without particularities. Depending on the display we want to use, different devices or techniques may be more convenient. One of the first elements we must consider when analyzing the properly pointing technique and device is the goal. We may find six types of interaction tasks in literature:

- **Selection:** Users must choose one or more than one element from a set of items. This technique is used for the simple task of selecting an item in a menu, to more elaborate tasks such as the selection of a part in a CAD design.
- **Position:** Positioning is another fundamental task in sketching, drawing, or CAD/CAM applications. Users must choose a point in one to three dimensions.
- **Orientation:** Like in the previous case, designing software often requires determining orientations for the new elements to be added or to modify certain aspects of the scene.
- **Path creation:** The path must be built from a set of positions and orientations or in a continuous way.
- **Quantify:** The selection of a quantitative value is often implemented as a one-dimension selection task.
- **Text:** Adding, deleting, or modifying text is also a task accessible through selection and pointing processes. Text must be added simply, but other operations such as text font and size, or page layout also fall into this category.

Although many of these tasks can also be implemented with a keyboard by typing numbers or letters to determine position, orientation, and so on, it is often far less efficient than with a direct manipulation tool. Expert users, however, may use a combination of direct pointing and keyboard shortcuts (such as *Ctrl+C*) to further accelerate the processing.

9.1 Pointing devices

There is a great amount of pointing devices and its number has increased constantly for several years. We may classify the pointing devices in two families:

- Direct-control devices
- Indirect-control devices

Direct-control devices are those that work directly on the surface of the display, such as the stylus or our fingers on a capacitive screen.

Indirect-control devices work away from the surface, such as the mouse, joystick, graphics tablet, and so on.

9.1.1 Direct-control devices

Direct-control devices have existed for a long time. One of the earliest devices is the *lightpen*, which enabled users to point to a spot on a screen and press a button to perform a selection operation.

The *lightpen* dates from back the middle seventies (you can see an example in Figure 35), as it was a device that already worked on UNIVAC 1652 in 1976. Compared to today's touch screens, the *lightpen* was heavy and fragile. Its operation caused fatigue and the hands over the screen obscured part of the information.

Although today's designs still produce higher fatigue than the mouse, a proper design, such as with a surface on which to rest the arm, greatly alleviates the problems of the *lightpen* and older screens.

Figure 35: Univac's interaction with a lightpen (around 1976).

Direct-control devices have other problems, still present in mobile phones, such as the imprecise pointer (in this case it is mainly related to the quality of the screen), and the ability of the software to accept the touch immediately (also called *land-on* strategy). This has some advantages, such as the fast response, but also some problems, such as the inability to correct the mistakes from the user. This may be somewhat painful for iPhone users. Since the unlock screen does not require a confirmation of the entry code, as can be seen in Figure 36, the user may be wrong, let's say, when keying the last digit, and then, the iPhone tries to unlock with a wrong code. Since the user may be aware on his or her mistake, it is frustrating not to be able to correct the input, and this may lead to completely lock the device if the same mistake is committed three times.

For some tasks, touch screens use the *lift-off* strategy. It has three steps: When the user taps on the screen, a cursor appears that can be dragged (without removing the finger from the screen) to finely adjust the correct position. When the user is satisfied with the cursor position, they lift the fingers off the display to activate. This is the technique used in the virtual keyboard of iOS system (see Figure 37) for accentuated or other special characters. In this concrete case, the new characters appear right on top of the finger's position, and the user moves horizontally over them to activate the desired character.

Figure 36: Unlock screen of the iOS devices. The lack of confirmation button is error-prone, since the user has no time to correct the input even if he or she detects that the last key was wrongly pressed.

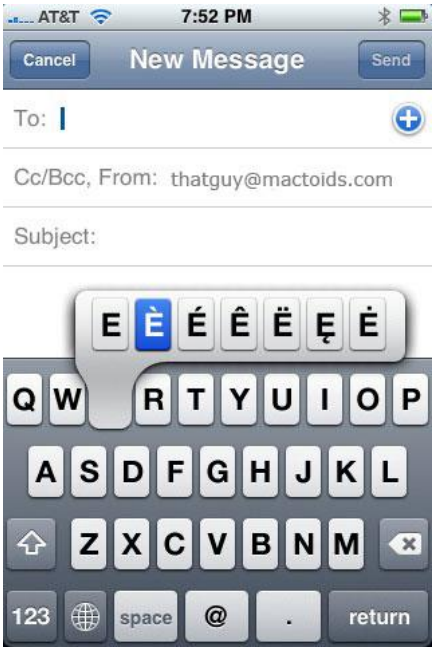

Figure 37: Lift-off strategy for introducing accented letters in the iOS operating system. The user selects the letter, and leaves the finger on, until the submenu appears with the differently accented versions of the selected letter.

Touch screens are often integrated into applications directed at novice users in which the keyboard can be removed completely. They have some advantages in public-access systems. Because they can be implemented with no moving parts, their durability in high-use

environments is good (some people say that the touchscreens are the only input device that has survived at Walt Disney World theme parks).

Multi-touch screens allow a single user to perform multiple finger entry, such as pinch-to-zoom gestures, or multiple users to work on the same application together.

Although some people may prefer finger input on touch screens, some tasks are better addressed with a pen. Pens are familiar and comfortable for users, and facilitate the movement of the cursor while leaving almost the whole context in view. The use of a pencil has also some shortcomings. The most clear is probably the additional work to pick up and put down the stylus.

9.1.2 Indirect-control devices

Indirect pointing devices alleviate hand-fatigue and eliminate the problems of hand obscuring the screen. On the other hand, they require the hand to locate the device and demand more cognitive processing and hand/eye coordination to bring the cursor to the desired target.

One of the most popular devices is the mouse: It is cost-effective, allows for a comfortable position of the hand, its buttons are easy to press, and positioning can be done quickly and with precision. However, for long movements, users must pick up and replace the mouse position.

Another popular device is the trackball, which resembles an upside-down mouse. The ball is used to move the cursor in the screen as it is moved. Joysticks have been long used in games. They are appealing for tracking purposes because they only need relatively small movements to move a cursor.

Touchpads have become very popular lately due to their presence in portable computers (and its decrease in price). They offer the convenience and precision of a tactile screen while keeping the user's hand off the line of sight. Like in the case of touch screens, their lack of moving parts is usually an advantage.

The graphics tablet is a touch-sensitive surface that is separated from the screen. It is often used lying flat on the table and can be operated with a finger, pencil, stylus, and so on, and usually provides input possibilities not only by position, but also by pressure. This last feature, almost unique in touchscreens, is highly appreciated by artists. Like the mouse, when resting in a flat surface, may be used for long periods without producing fatigue.

9.2 Comparison of pointing devices

When analyzing pointing devices, there are two major variables to take into account: speed and accuracy. Some studies have found that direct pointing devices are often faster but more prone to errors than indirect control devices.

For decades, the mouse has shown its superiority to other devices in speed and accuracy. For instance, the mouse is faster than the *trackpoint* due to tremors in finger motion during fine finger movements.

When analyzing two devices we must take into account the task. For scrolling large lists or large documents or webpages, mice equipped with a scroll wheel may be convenient, although some studies showed that they are not superior to regular mice.

The usual belief is that pointing devices are faster than keyboard cursor keys, but depending on the task, these may be superior due to the smaller movement required to use them, and the fact that some shortcuts, if mastered, are very convenient for document edition. Usually, short tasks that mix typing and pointing are faster addressed with cursor keys.

Users with motor disabilities may prefer devices that are fixed, such as joysticks or trackballs.

10 3D Selection

10.1 Introduction

Several designers believe that 3D interfaces can make several tasks easier than classical 2D systems. They believe that this would allow a behavior that is similar to real life, and therefore, tasks can be accomplished more efficiently. In Figure 38 we see an example of 3D environment being used by two people at the same time.

Figure 38: An inspection session in a 3D virtual environment. Both the users can see in 3D but the view is calculated accurately for the user whose googles are tracked (left).

Besides that, some people are also studying whether enhanced interfaces may even be better than reality. Of course, the term *better* may have different meanings, but the idea is that some scenarios can be performed using a virtual reality environment in a more effective way than in the 3D world. There is some controversy, because the tasks a 3D

environment may be more suitable or even superior to *reality* need to be computer tasks, such as computer-assisted design, molecular modelling, and so on. In these tasks, we can provide the user with super-natural powers such as travelling back in time to undo some of the actions.

10.2 Definitions

Over the last decade, the field of 3D user interfaces has grown out of its infancy, forming the basis for many game and industry applications. There are some terms that are important to get familiar with:

- **3D interaction:** An interaction between a person and a computer in which the user's tasks are carried out in a 3D spatial context using directly 3D input devices or 2D input devices with direct mappings to 3D.
- **3D user interface:** A User Interface that involves 3D interaction.
- **3D interaction technique:** The technique designed for solving a certain task. This may involve both the use of hardware (a device) and software (a module that has as input the device signals and that implements the behavior in the virtual space).

Note that none of those definitions implies the use of a 3D environment in the sense of Virtual Reality (i. e. stereo viewing as a minimum). Although many 3D interfaces are commonly linked to 3D environments, we may implement those in a desktop system without stereoscopy.

There are many devices tailored to produce 3D stereo and many devices used to interact with 3D interfaces. We will not visit them, since this is beyond our scope. We will only address one of the problems that arises in 3D environments and whose solution is complex: 3D selection.

10.3 3D selection

We refer to 3D selection as the selection task when it is carried out in a 3D immersive environment. Compared to other selection methods, new problems appear, such as the occlusion of the target.

7.3.1 3D Selection techniques

Commonly, 3D object selection techniques are implemented with a selection tool such as a data glove or a Wanda device. Since we need to select in a 3D environment, we need to define a 3D position, which is often given by a ray, a 3D cursor or a simple 3D shape such as a sphere. The computer must perform a 3D intersection or proximity test with the environment. In contrast to the *selection tool*, the method used for effectively selecting the elements is usually called *selection technique*.

The selection technique defines how the user will control the selection element. In typical VR environments tracking devices will usually perform the monitoring of the users' actions.

Selection in 3D Virtual Reality environments can be accomplished using different techniques. However, the two main paradigms are:

- **Hand extension techniques:** These techniques are also called 3D point cursors, since they represent a 3D point in space as a mapping of the user's hand position (see Figure 39).
- **Ray-based techniques:** Instead of using the hand position, these techniques use the position and some other element to indicate an orientation to generate a ray in space that is used as a pointer. These techniques are also called aperture-based selection techniques or ray cursors. The different configurations can be seen in Figure 40.

Figure 39: The representation of a hand in a VR driving simulator as a mapping of its real position and orientation.

Previous research in VR has demonstrated that ray-based techniques are often superior to hand extension techniques due to the faster selection times.

10.3.2 Ray-based selection and pointing

In order to define a ray, we need an initial position, and an orientation. Usually, the user controls the 3D cursor by moving its dominant hand: the control is performed using the hand's position as the initial point, and the direction of the ray is calculated using the users' wrist orientation. Sometimes, the virtual ray starts from the head of the observer and that goes along the hand position.

For selecting single objects or points in scenes, ray cursor techniques have an inherent problem. Since the ray is a 2D geometric object, the ray often intersects multiple objects, and so the actual target of interest is ambiguous. Moreover, selecting occluded elements, if needed, may become a problem, because the occluding objects must be erased, moved, or visually removed in an easy way. A second possibility is to define a selection point on the pointing ray. Then, some mechanism has to be provided to move the cursor along the ray.

This is also problematic because the hand can move (especially if we also use it to operate a wheel or a joystick that performs the 3D cursor movement on the ray) and the desired direction can change. One possibility is to lock the ray after the direction has been chosen appropriately (a technique is called *lock ray*) and then allow for the movement of the 3D cursor on the ray. The three methods are illustrated in Figure 40.

There is a second problem that arises when using rays defined by the hand as their initial position: the ray direction and the viewing direction is not the same. As a consequence, some object that is visible from the user's viewpoint can be unreachable from the user's hand position. This is not easy to address, since many computations have to be done on real time if we want the application to detect this problem. Some elements that may alleviate this issue consist of using elements such as the sticky targets in 3D [Andujar2007] (Figure 41) and enlarging the pointed elements in a similar way to the bubble controls [Argelaguet2008] (see Figure 42), or to locally flatten the elements to facilitate pointing.

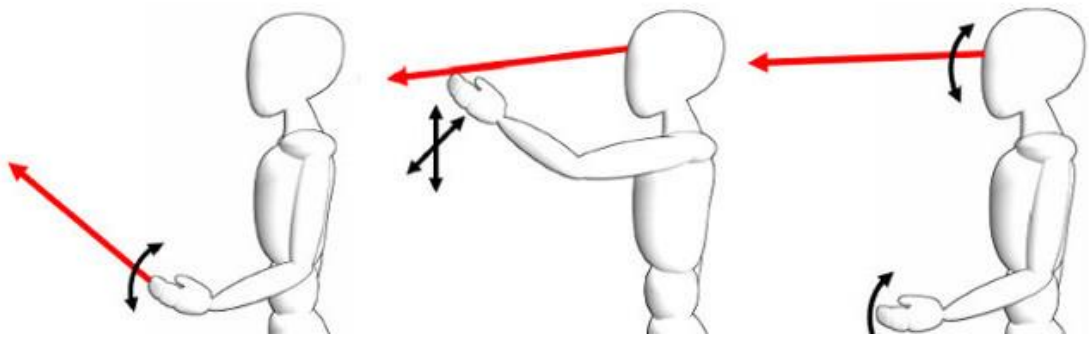

Figure 40: The three typical paradigms of ray pointing in VR: Left shows the interaction being controlled by the hand position and wrist orientation. Center shows the ray being started at the eye, and the intersection with the hand creates the ray direction. Right shows the ray starting in the eye, while the direction is controlled by the wrist orientation.

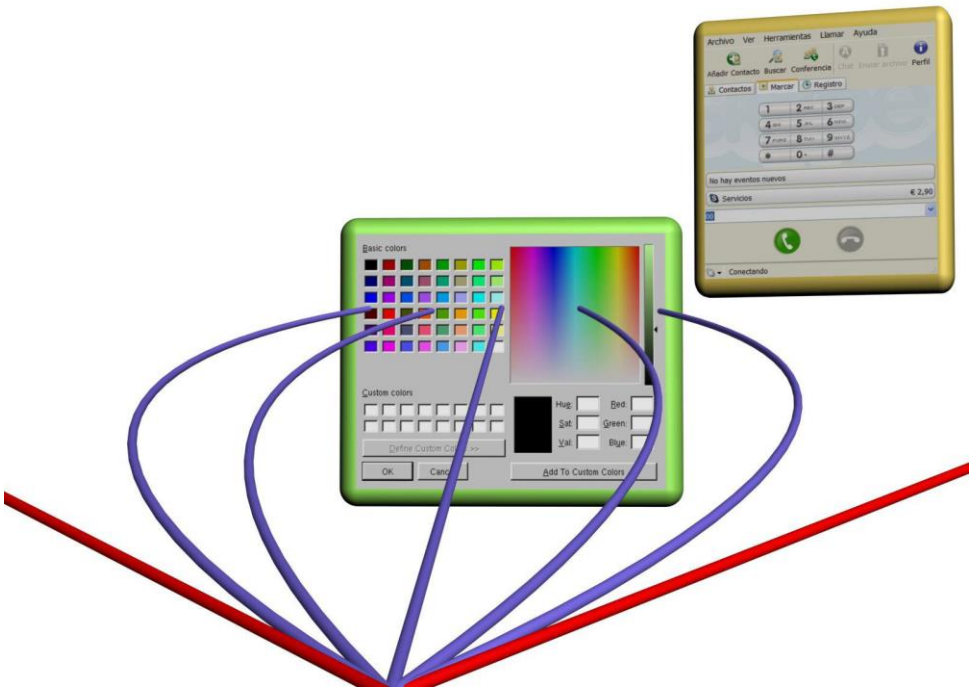

Figure 41: Implementation of sticky targets for ray attraction in a 3D environment for the help in menu selection.

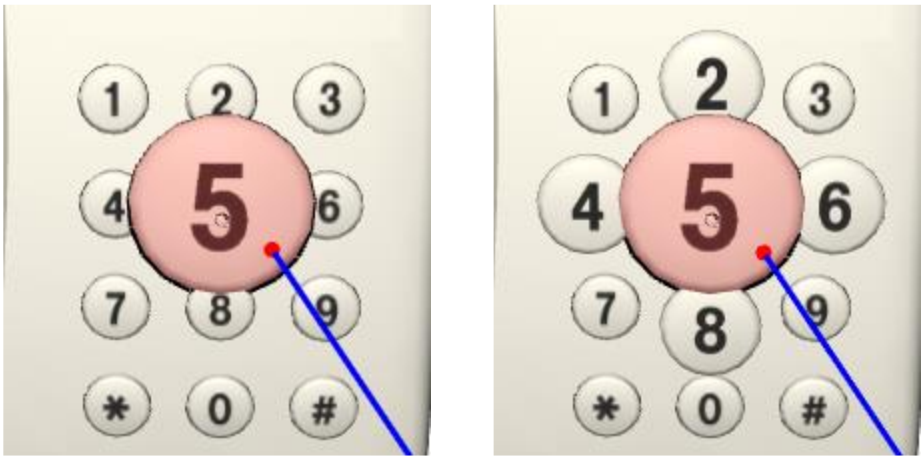

Figure 42: Expanding targets in a 3D environment [Argelaguet2008]. Different configurations are possible, such as only expanding a single target, the one closer to the user (left), or an expansion of all closer selectable targets (right).

11 Bibliography

[Accot97] Accot, J. & Zhai S. (1997) *Beyond Fitts' law: models for trajectory-based HCI tasks*. In Proceedings of ACM CHI'97 Conference on Human Factors in Computing Systems, pages 295–302.

[Accot99] Accot, J. & Zhai. (1999), Performance evaluation of input devices in trajectory-based tasks: an application of the steering law. *CHI '99 Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 466-472.

[Accot2002] Accot, J., & Zhai, S. (2002). *More than dotting the i's - foundations for crossing-based interfaces.* Proc. CHI: ACM Conference on Human Factors in Computing Systems, Minneapolis, Minnesota, 73 - 80.

[Allison2004] Allison, B., Desai, A., Murphy, and Sarwary, R.M., (2004) Choice reaction time in card sorting tasks: validation of the Hick-Hyman Law, San Jose State University ISE 212, Summer 2004.

[Argelaguet2008] Argelaguet, F., Andújar, C. (2008). Improving 3D Selection in Immersive Environments through Expanding Targets. In *Proceedings of the 8th International Symposium on Smart Graphics (SG '08).* pp:45–57.

[Andujar2007] Andújar, C., Argelaguet, F. (2007). Anisomorphic Ray-Casting Manipulation for Interacting with 2D GUIs. *Computers & Graphics*. 31:15–25.

[Apitz2010] Apitz, G., Guimbretière, F., Zhai, S. (2010). Foundations for designing and evaluating user interfaces based on the crossing paradigm. ACM Transactions Computer-Human Interaction, 17(2).

[Balakrishnan97] Balakrishnan, R. and MacKenzie, I.S. (1997). Performance differences in the fingers, wrist, and forearm in computer input control. *CHI ' 97: ACM Conference on Human Factors in Computing Systems*. p. 303-310.

[Bi2013] Bi, X., Li, Y., & Zhai, S. (2013). FFitts law: Modeling finger touch with Fitts' law. In *Proceedings of the 2013 ACM annual conference on Human factors in computing systems*, pp. 1363-1372.

[Card83] Card, S.K., Moran, T.P., & Newell, A. (1983). The Psychology of Human–Computer Interaction. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

[Chapuis2009] Chapuis, O., Labrune, J.-B., and Pietriga, E. (2009). DynaSpot: Speed-dependent area cursor. *Proc. CHI '09: ACM Conference on Human Factors in Computing Systems*. New York: ACM Press, 1391–1400.

[Chapuis2011] O. Chapuis, and P. Dragicevic, (2011) Effects of motor scale, visual scale, and quantization on small target acquisition difficulty. *ACM Transactions on Computer-Human Interaction (TOCHI)*, Volume 18 Issue 3, July 2011, pp. 13-32, ACM New York.

[Cockburn2003] Cockburn, A. and Firth, A. (2003). Improving the acquisition of small targets. *British Human Computer Interaction Conference*. p. 181-196.

[Cockburn2007] Cockburn, A., Gutwin, C. and Greenberg, S. (2007). A Predictive Model of Menu Performance. Proceedings of *CHI'07: ACM Conference on Human Factors in Computing Systems*, San Jose, CA, 627-636. ACM Press.

[Cockburn2008] Cockburn, A., Gutwin, C. (2008). A Predictive Model of Human Performance with Scrolling and Hierarchical Lists. In *Human Computer Interaction*, vol. 24 no. 3, 273-314.

[Crossman83] Crossman E., Goodeve P. (1983). Feedback control of hand movement and Fitts' law. *Quarterly Journal Experimental Psychology*, 35A:251–278.

[Donders68] Donders, F. C. (1868). Die Schnelligkeit psychischer Processe. (On the speed of mental processes). *Archiv für Anatomie und Physiologie und wissenschaftliche Medizin*, 657–681.

[Fitts54] Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47, 381–391.

[Fitts54b] Fitts, P. M., & Deininger, R. L. (1954). S-R compatibility: Correspondence among paired elements within stimulus and response codes. *Journal of Experimental Psychology*, 48, 483–492.

[Fitts54c] Fitts, P. M., & Seeger, C. M. (1954). S-R compatibility: Spatial characteristics of stimulus and response codes. *Journal of Experimental Psychology*, *46*, 199–210.

[Fitts64] Fitts, P. M., & Peterson, J. R. (1964). Information capacity of discrete motor responses. *Journal of Experimental Psychology*, 67, 103–113.

[Grossman2005] Grossman, T. and Balakrishnan, R. (2005). The Bubble Cursor: Enhancing target acquisition by dynamic resizing of the cursor's activation area. *Proc. CHI '05: ACM Conference on Human Factors in Computing Systems*. New York: ACM Press, 281–290.

[Hartley28] Hartley, R.V. L. (1928). Transmission of information. *Bell System Technical Journal, 535*, 1928.

[Hertzum2013] Hertzum, M., and Hornbæk, K., (2013). The Effect of Target Precuing on Pointing with Mouse and Touchpad, *International Journal of Human-Computer Interaction*, vol. 29, no. 5 (2013), pp. 338-350.

[Hick51] Hick, W. E. (1951). A simple stimulus generator. *Quarterly Journal of Experimental Psychology*, *3*, 94–95.

[Hick52] Hick, W. E. (1952). On the rate of gain of information. *Quarterly Journal of Experimental Psychology*, *4*, 11–26.

[Hick53] Hick, W. E. (1953). Information theory in psychology. *IEEE transactions on Information Theory*, *1*, 130–133.

[Hyman53] Hyman, R. (1953). Stimulus information as a determinant of reaction time. *Journal of Experimental Psychology*, *45*, 188–196.

[ISO2000] ISO. (2000). *ISO9241-9 International standard: Ergonomic requirements for office work with visual display terminals (VDTs)—Part 9: Requirements for non-keyboard input devices:* International Organization for Standardization.

[Kristensson2014] Kristensson, P.O. (2014). From wax tablets to touchscreens: an introduction to text entry research. *ACM XRDS 21(1): 28-33. Invited. Special Issue on Computers and Language*.

[Landauer85] Landauer, T. K., & Nachbar, D. W. (1985). Selection from alphabetic and numeric menu trees using a touch screen: Breadth, depth, and width. *Proceedings of the CHI'85 Human Factors in Computing Systems*, 73–78. New York: ACM.

[Lewis99] Lewis, J. R., Kennedy, P. J., & LaLomia, M. J. (1999). *Development of a Digram-Based Typing Key Layout for Single-Finger/Stylus Input.* Proc. of The Human Factors and Ergonomics Society 43rd Annual Meeting.

[MacKenzie89] MacKenzie, I. S. (1989) A note on information-theoretic basis for Fitts' law. *Journal of Motor Behavior*, 21, 323-330.

[MacKenzie92] MacKenzie, I. S. & Buxton, W. (1992). Extending Fitts' law to two-dimensional tasks. Proceedings of the CHI'92 Conference on Human Factors in Computer Systems. New York: ACM.

[McGuffin2002] McGuffin, M. and Balakrishnan, R. (2002). Acquisition of expanding targets. *ACM CHI Conference on Human Factors in Computing Systems*. p. 57-64.

[Merkel85] Merkel, J. (1885). Die zeitlichen Verhältnisse derWillensthätigkeit (The Temporal Relations of the Actions of Will, or The Timing of Voluntary Action). *Philosophische Studien (Philosophical Studies)*, *2*, 73–127.

[Nguyen2012] Nguyen, H., Bartha, M. C. (2012). Shape Writing on Tablets: Better Performance or Better Experience? *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, San Diego, CA, HFES, pp. 1591-1593

[Norman82] Norman, D. A. and Fisher, D. (1982). *Why alphabetic keyboards are not easy to use: Keyboard layout doesn't much matter*. Human Factors, 24, 509-519.

[Nyquist24] Nyquist, H. (1924). Certain factors affecting telegraph speed. *Bell System Technical Journal*, *47*, p. 617.

[RIM2012] Research in Motion (2012). *The BlackBerry 10 keyboard demoed*. http://www.youtube.com/watch?v=lOwcIuq_TUU, checked Oct. 2014.

[Schenider2013] Schneider, T. D. (2013). Information Theory Primer, With an Appendix on Logarithms, http://alum.mit.edu/www/toms/paper/primer/ (checked October 2014).

[Sears91] Sears, A., and Shneiderman, B. (1991). High precision touchscreens: Design strategies and comparison with a mouse. *International Journal of Man-Machine Studies*, 34, 4, pp. 593-613.

[Seow2005] Steven C. Seow (2005). Information Theoretic Models of, HCI: A Comparison of the, Hick-Hyman Law and Fitts' Law, *Human-Computer Interaction*, 2005, Volume 20, pp. 315–352.

[Sears94] Sears, A., and Shneiderman, B. (1994). Split menus: Effectively using selection frequency to organize menus. *ACM Transactions On Computer-Human Interaction* 1, 1, 27 - 51.

[Shannon48] Shannon, C. E. (1948) A mathematical theory of communication. *Bell System Technical Journal*, 27, 379–423, 623–656, 1948.

[Sourkoreff2004] Soukoreff, R.W., & MacKenzie, I.S. (2004). *Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts's law research in HCI*. International Journal of Human- Computer Studies, 61, 751–789.

[Welford68] Welford, A. T. (1968). Fundamentals of skill. London: Methuen.

[Whirlscape2014] Whirlscape (2014). *The Minuum Keyboard*. http://minuum.com/

[Wright2013] Wright, C. E., & Lee, F., (2013) Issues Related to HCI Application of Fitts's Law, *Human-Computer Interaction*, Volume 28, Issue 6, pp. 548-578, 2013.

[Zhai2002] Zhai, S., Accot, J., Woltjer, (2002) R. Human Action Laws in Electronic Virtual Worlds — An Empirical Study of Path Steering Performance in VR, *Presence: Teleoperators and Virtual Environments*, Vol.13(2), 113-127.

[Zhai2004] Zhai, S., Kong, J., & Ren, X. (2004). Speed-accuracy tradeoff in Fitts' Law tasks: On the equivalency of actual and nominal pointing precision. *International Journal of Human-Computer Studies*. 61, 6, 823–856.

[Zhai2012] Zhai, S. and Kristensson, P.O. (2012). The word-gesture keyboard: reimagining keyboard interaction. *Communications of the ACM 55(9): 91-101. Invited. (Research Highlights)*.