Interaction techniques in virtual environments

Sergio Moya, Sergi Grau, Dani Tost

February 7, 2012

1 Introduction

Virtual environments are used in very diverse applications: from leisure games to professional training and rehabilitation systems. In these applications, users interact with the elements of the environments through avatars in order to perform virtually actions such as picking objects, carrying and dropping them, and more complex manipulations tasks such as cutting and drilling in surgery training, shooting and punching in actions games, and cooking and cleaning in life simulation games and neuropsychological rehabilitation. To launch these actions, users must first select the objects with which they want to interact by pointing on them and confirming the selection by clicking a button or a key. This process is usually defined as *point-and-click* and more shortly *pointing* [**?**].

Pointing is an essential feature of interaction in virtual environments, and it is relevant to measure its usability. Besides, pointing in 3D environments has a lot of similarities with pointing in 2D graphical interfaces. Fitts's law [**?**] relates the efficacy of pointing with the distance to the target and the target's width. The efficacy is also affected by the density of the environment and its level of occlusion. The influence of these factors has been studied in various papers to propose different types of pointing mechanisms. In this report we classify and survey these techniques.

2 Background

The underlying principles of virtual pointing are inspired on those of physical pointing. Fitts's law [**?**] establishes that the time T required by a subject

instructed to work at maximum rate to touch an object with a hand or finger is a function of the minimum average amount of information per response demanded, which is the distance of the initial position of the hand and the object (D), also called *amplitude* and the *width* of the target seen along the axis of motion (W). Specifically:

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

where, the *index of difficulty* (I_d) of the task is the logaritmic term:

$I_d = \log_2 \left(1 + \frac{D}{W}\right)$, and a and b are empirically measured constants. Interpreting T as a linear regression of I_d, b is its slope and $1/b$ is called *Index of performance* and a is its intercept. The distance between the initial position of the hand and the target (D) is also called *Amplitude*

Fitts's law was experimentally tested with three manuals tasks: tapping and disk and pin transfer, but later experiments have shown that it works with a wide range of pointing actions. Moreover, it applies under a variety of subjects, physical environments and limbs. In addition, Card, English and Bur [**?**] and later MacKenzie[**?**] showed that Fitts's law can be extended to virtual pointing with a variety of input devices as mouses and stylus by only adjusting the values of parameters a and b. This is why it has become a basic reference in 2D and 3D graphical interfaces design.

Several authors have proposed optimized-submovement models to explain Fitts's law. The most popular model was proposed by Meyer et al. [**?**]. It divides the movement into two steps: an aimed large and fast movement toward the target called *open loop* and, eventually, a secondary set of slower and shorter corrective submovements *close loop*. The model is based on the observation that the standard deviation of the endpoint of a movement is proportional to the quotient between the distance and the time needed to reach it: $S = k(\frac{D}{T})$. Therefore, although the first long distance movement brings fastly the subject close to the target, it can miss it. Thus, a slower and short corrective movement can be necessary. This way, the total movement time is minimized.

3 Classification

Fitts's law has been applied to the design of 2D Graphical User Interfaces (GUI). For instance, since it states that, at constant distance, the larger the

width of the target, the faster is the movement, in a GUI icons and buttons must all have a reasonable size, and even, the size of the most used buttons can be larger than the others. Similarly, it is worth to put GUI elements at the edges and corners of a graphical area, because the mouse sticks on them and, thus, it is as if they had an infinite width [**?**].

In addition to these 2D interface design issues, pointing can be enhanced by indirectly reducing the distance D to the target and the target's width W without altering significantly the interfaces [**?**]. These optimized stategies can significantly improve pointing. Moreover, they can be extended to pointing in 3D environments, where the object sizes and positions are structural and cannot be customized as in 2D GUI. Table **??** shows a classification of the techniques surveyed in this document according to the dimension of the interaction space. Related with the dimensionality of space, a second aspect that differentiates pointing techniques is the type of application for which they are designed: 2D and 3D computer leisure games, educational ort therapeutical training, operating system desktop and other specific applications (see Table **??**). Moreover, some techniques have been designed for specific types of display such as touch screens [**?**] or input devices, such as stylus [**?**] (**??**).

Table 1: Classification of existing pointing techniques according to space dimension

Table 2: Classification of existing pointing techniques according to their application

Table 3: Classification of existing pointing techniques according to the hardware for which they have been designed

For 2D interfaces, Balakrishnan [**?**] classifies enhanced pointing methods into three categories, those aimed at indirectly:

- reducing D
- enlarging W

Table 4: Classification of existing pointing techniques according to the parameter that they optimize

- mixed techniques aimed at reducing D and enlarging W

In Table **??** we present classification of the papers surveyed in this document according to this criterion. The first category emcompasses methods aimed at reducing the distance between the cursor and the target by either moving the target close to the cursor o vice-versa. Since, usually, various possible targets are visible, these methods must predict which one is more likely the one that the user wants to select. This prediction can be based on different types of data: from a task-dependent knowledge of the user's intentions [**?**], to an analysis the previous movement of the cursor [**?**] and even an analysis on the location to which the user watches []. This type of predictions are also used in the design of aiming robots also called *aimbots*, third-person programs used by cheaters to gain unfair advantage over other players in multiplayer online games [**?**].

Width manipulation techniques expand artificially the target. They are often subdivided into two categories: those that expland the target [**?**] and that which enlarge the cursor [**?**] or the cursor's area of influence [**?**]. In the former category, some method expand the targets width visually only (*visual space target expansion*), while others expand them also in motor space *visual and motor-space target expansion* by altering the actual size of the targets and not only their appearance.

Finally, the third category of methods manipulates the distance and the width simultaneously by altering the mapping between the physical displacement of the pointing device and the corresponding cursor movement on the screen *(control-display ratio)*. Some of these techniques have been defined as *motor-space only target expansion* techniques.

The influence of other aspects than distance and width, such as colouring and highlighting targets has also been explored together with other sensory feedback such as auditory and tactile responses [**?**] [**?**] (see Table **??**).

Table 5: Classification of existing pointing techniques according to the factors that they consider to influence efficacy.

In addition to Balakrishnan's criterion, pointing enhancing techniques can be classified according to the type of users for which they have been

designed: regular experienced and non-experienced users [**?**] or persons with motor and/or cognitive impairements [**?**] (see Table **??**).

Table 6: Classification of existing pointing techniques according to the type of users

.

Finally, other important aspects of the reviewed techniques are the soundness of the experiments described to validate the proposed techniques (number of experiments, users and profiles) and the techniques with which they are compared. Table **??** summarizes these aspects.

Table 7: Classification of existing pointing techniques according to the type of the testing conditions

.

Finally, a closely related to pointing technique are contrained cursor motion across boundaries [**?**]. Since these techniques can be used alternatively to *point-and click*, we also describe them in a separate section **??**.

Table 8: Pointing and goal-crossing techniques

4 2D pointing

4.1 Minimizing distance techniques

Minimizing distance techniques aim to minimize the distance travelled by the cursor device in the physical world. There are two ways to achieve it: reducing the distance between cursor and active objects, and increasing the speed of the cursor in order to achieve target objects faster.

4.1.1 Reducing distance to objects

Baudisch et al. [**?**] propose two different techinques to reduce the distance between the cursor and the interaction icons: drag-and-pop and drag-and-pick. Drag-and-drop technique is an extension of the well-known drag-and-drop interaction technique for transferring or copying information. In this

case, when an object or icon is selected, linked objects or icons that can interact with it are shown in front of the user's cursor. The icons shown are a subset of the icons on the screen, and they are selected if their types are compatible. Drag-and-pick tecnique consists on showing, as a reaction of a drag interaction, all the objects or icons located in the direction of the drag motion, also called target sector. The sector sizes vary between 20 to 45 degrees, depending on the user's experience. These two techniques were tested by 7 participants with no experience using the techniques (2 female and 5 male between 18 and 35). All were right handed with normal or corrected-to-normal vision. The tests were run on the DynaWall (3 Smartboards units) and the task was to drag 10 document icons in different arrangements into a given target folder or application.

4.1.2 Increasing the cursor displacement speed

Guiard et al. [**?**] propose the Object Pointing technique, where they add an additional cursor plus the common pointer cursor. The additional cursor never visits empty regions of the space and always rests on an active object. The active object which is selected is determined by the direction of the common pointer movement. Depending on the direction, acceleration and velocity, the Object Pointing algorithm chooses an active object, puts the additional cursor on it and highlights it. When the additional cursor rests on the active object, it and common cursor suffer a parallel displacement, corresponding to the movement of the mouse, until the additional cursor reaches the boundary of the active object. When it occurs, the algorithm selects the next active object in the displacement direction and the additional cursor jumps to it. Object Pointing algorithm uses the last 5 frames of the input history to calculate the direction and the instant velocity. The algorithm selects an angular sector in this direction in order to find the next object quickly. The argument for the workability of the Object Pointing technique is that the gaze precedes the movement. The additional cursor simulates the gaze behaviour and intends to reduce the lag between the eye and the hand. The authors shows two experiments to test Object Pointing, 1D and 2D, by comparsion with the common pointer cursor. The test is realized for 12 unpaid participants and they conclude that Object pointing reaches higher speed than common pointer for high indices of difficulty, but not for low indices of difficulty, where the common cursor is faster.

Object Pointing selects all active objects along the direction of the move-

ment until the target object is reached by the additional cursor. In order to avoid these intermediate selections or jumps, Asano et al. [**?**] explore the possibility of predict directly the final target, in their Delphian desktop approach, by using the peak velocity of the movement. By experimental way, they prove that the peak velocity is directly proporcional to the distance between the current position of the cursor and the final target. Using this relation they are able to determine the direction and distance and, consequently, the target position. In their experiments with sixteen participants, they conclude that this prediction is effective when the users try to point faraway targets. For short distances, the time required for the movement is not sufficient to estimate the target position. In addition, this method requires a calibration per user in order to determine the user-dependent parameters of the distance prediction function.

4.2 Increasing width techniques

4.2.1 Expanding cursor techniques

Worden et al. [**?**] propose Area cursors method, which uses a squared hot spot instead of a single point. Each pixel in the spot behaves as a pointer cursor and can select any active object in the screen. This method is proposed together with Sticky icons **??** and was tested, doing a comparsion between both methods and the common pointer, by sixteen younger adults and sixteen older adults. They conclude that Area cursors improve the speed performance, specially, for older adults.

Grossman and Balakrishnan propose the Bubble cursor [**?**] technique. It consists of a round and semitransparent cursor that varies its size depending on the closest active object. The cursor arc always lies on the closest active object, calculated by Voronoi diagrams, which is activated. Therefore, this cursor cannot select empty space. In order to provide feedback to the user, the cursor is morphed to envelop the target object which is also highlighted. Grossman and Balakrishnan test this method for 1D and 2D, for ten and twelve experienced volunteers respectively. Test are realized in comparsion with the common pointer cursor and Bubble cursor shows better speed performance in all cases, specially, in dense scenarios.

Chapuis et al. [**?**] propose a Bubble cursor enhancement named DynaSpot. It uses the velocity of the pointer to dynamically adapt the cursor size without need of explicit switch. The method changes the size of activation area,

named spot, as a function of cursor speed until a certain threshold. The faster is the cursor movement, the greater is the spot size. When the cursor is totally stopped, after a certain lag, the spot size decreases during the reduction time empirically chosen. When more than one active object lies on the cursor area, the closest object to the cursor's pointer (center of the cursor) is selected. This approach allows empty space selection in contrast to Bubble cursor. The authors perform an experiment that compares the DynaSpot with Bubble cursor and simple pointer realized by twelve experienced users. The conclusion is that DynaSpot improves the pointer cursor by 18% speed-up and achieves better performance than Bubble cursor for low density scenarios and poorer performance for high density scenarios, where the selection tasks is more critical.

4.2.2 Expanding target techniques

Expanding the target is a way of increasing W. However, expanding all the objects a priori alters the user perception of the environment, and it can cause occlusions between nearby objects. Therefore, it is preferrable to expand the targets dynamically as users focus on them.

MacGuffin and Balakrishnan [**?**] have shown that Fitts's law is also valid for expanding targets, and that they improve targeting performance even when expansion occurs after a 90% of the movement has been done. Morevover, improvement depends on the final target size. In their experiments they applied only *visual target expansion*, without altering the actual size of the targets. Zhai et al. [**?**] revisited these experiments but applied expansion and shrinking randomly so that participants could not anticipate the final size of the target. They showed that the improvement of pointing using expansion is higher if it is the current user's target which is expanded, but that enlarging the target improves pointing in both cases. This result corroborates Gutwin's observations [**?**], who reported a negative impact on focus-targeting with fisheye visualizations [**?**]. He observed that as the focus changes, demagnified regions appear to shrink in the opposite direction of the movement. Thus, *visual-space expansion* does not bring motor-space advantage over static objects. He claimed that the performance improvement comes from motor-space enlarging. Therefore, he proposed a *visual and motor-space target expansion* technique called *speed-couple flattening* aimed at reducing the distortion when user is engaged in targeting, which is detected by an increased cursor acceleration and velocity. The underlaying idea is that users

move faster when they are targeting than when they are inspecting data. Cockburn tested their method on a reduced number (10) experienced users and concluded that it significantly improved targeting.

Cockburn and Fifth [**?**] experienced with expanding bubble targets and showed that they were efficient for small targets. In a later experiment, Cockburn and Brock [**?**] compared the benefits of *visual space target expansion* and *visual and motor-space target expansion*. They concluded that, for small objects, the differences between the two techniques are small.

Ramos et al. [**?**] proposed a *visual and motor-space target expansion* technique called *Pointing lenses* that temporarily enlarges the area under the cursor. Mandryk and Gutwin [**?**] found two problems in expanding target techniques: distracting side effects of the visual changes and eventual occlusions between targets. They argue that *motor-space-only expansion techniques* reviewed in Section **??** solve these problems.

4.3 Mixed techniques

4.3.1 Motor-space target expansion techniques

Findlater et al. [**?**] propose two Motor-space target expansion approaches for people with motor impairments. The first, Motor-Magnifier (MM), magnifies the motor space by a default factor of four in order to ease the pointing. On the other hand, Visual-Motor-Magnifier (VMM) magnifies the motor space by the same way that Motor-Magnifier but also magnifying the visual space by zooming the active area.

Worden et al. [**?**] explore the possibility of vary the C-D gain (motor space) when the cursor overlays an active object. The cursor reduces its velocity with respect to the device gain in this situation. This method is named Sticky icons and has an adaptive variant. The adaptive way only varies the gain when the cursor's velocity is lower than 30% of the peak velocity in order to enable sticky approach only for corrective sub-movements. This way intends to an icon becomes sticky only when it is the target icon. The authors tested, through 6 experiments, Sticky icons, Area cursors **??** and the common pointer cursor. Sticky icons improve the performance in all cases, but mostly in the older adult cases and, specially, when there are small targets on the contrary than large targets.

Semantic Pointing technique, proposed by Elmqvist and Fekete [**?**] consists of varying the C-D gain (motor space) when the cursor overlays an active

object reducing the cursor's velocity with respect to the device gain. This method is similar to Sticky icons, but additionally providing visual feedback by varying the cursor size. The farther is the target object, the bigger is the cursor size. When the cursor overlays the target object, it changes its color. The authors performed an informal user study with 8 experienced participants. They don't conclude anything in terms of selection speed. The selection accuracy, measured as the number of clicks per target, participants with semantic 3D pointing performed better than without.

5 Goal Crossing Techniques

Goal crossing techniques directly select the items onto which the cursor passes without need of explicit click-based selecting actions. Strictly, these techniques cannot be categorized as pointing. However, Accot and Zhai [**?**] showed that the time needed to cross a goal of with W at a distance D is an expression with the same form as Fitt's law but with the W measured in a direction orthogonal to the movement and not collinear to it as in the original law (see Figure **??**.

Cockburn and Fifth [**?**] experienced with goal-crossing expanding bubble

Findlater et al. [**?**] propose two goal-crossing approaches for people with motor impairments. In the first, Click-and-Cross (CLC), the user moves a circular area cursor and clicks to activate the crossing cursor. The crossing cursor consists in a circle around the clicked area. Target objects musts be selected by crossing target arcs, which are areas on the circle that overlays the object location. This method requires one click. On the other hand, in Cross-and-Cross (CRC), the area cursor is a circle around the pointer. The user moves this circle pushing it by moving the pointer near the circle edges. When the pointer is moved to the circle center, the circle remains still in its position. If the user waits 300 ms, the area becomes activated and the crossing cursor appears the same way as Click-and-Cross approach. This method eliminates the click requirement but it is necessary to wait 300 ms to activate the area.

6 3D pointing

Vanacken et al. [?] propose the Depth ray technique consisting of throwing a conic ray from the cursor that produces the volume of the active area. It has to disambiguate between activated objects. The object closest to a depth marker is the selected object. The selection marker lies along the central axis of the cone and can be moved by moving the 3D device in this axis direction. The authors tested it in comparsion with 3D Bubble cursor **??**. The results showed that Depth ray is the fastest in sparse environments.

Steinicke and Hinrichs [?] propose the Grab-and-throw metaphor for 3D Virtual Environments (VEs) that consists an extension of the drag-and-drop technique. It allows users to select a virtual object by grabbing it and to throw the object within VE. They show examples that follow object-action model and action-object model. The virtual throw is modified to ease the target acquisition. The trajectory is modified to reach the closest object (active object) to the original point of rest of the throw. The active object must be inside a defined sphere-shaped search region, otherwise the throw is cancelled.

7 todo

Bateman at al. [?] analyze the use of several pointing techniques as a mechanism of skill-accomodation between experienced and non-experienced competitive computer games players.

Potter et al.

8 Conclusions