Pointage

Olivier Chapuis
chapuis@lri.fr

Plan

1. Modélisation du Temps de Pointage
2. Techniques de pointage

Plan

1. Modélisation du Temps de Pointage
2. Techniques de pointage

Pointage: Loi de Fitts

$$TM = a + b.ID \quad \text{où} \quad ID = id(\frac{D}{W})$$

$$id(x) = \log_2(x+1), \quad id(x) = \log_2(2x), \quad id(x) = (x)^{1/c}, \text{ ...etc.}$$

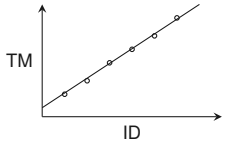
TM: Temps de Mouvement, *ID*: Indice de Difficulté (de la tâche) et a, b: constantes empiriques (qui dépendent des utilisateurs, du périphérique ...etc.)

Points importants: on regroupe les deux facteurs D et W en un seul $\frac{D}{W}$ et forme de la fonction id.

Forme la plus populaire en IHM (Mackenzie-Shannon):

$$TM = a + b.\log_2(\frac{D}{W} + 1) \text{ i.e. } ID = \log_2(\frac{D}{W} + 1)$$

Pointage: Loi de Fitts



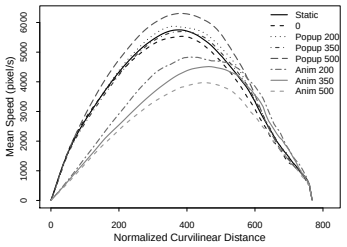
TM est donc linéaire en un indice *ID* qui représente la difficulté de la tâche.

ID est une fonction de $\frac{D}{W}$, on peut voir la loi de Fitts comme une balance entre vitesse (D) et précision (W). speed-accuracy tradeoff

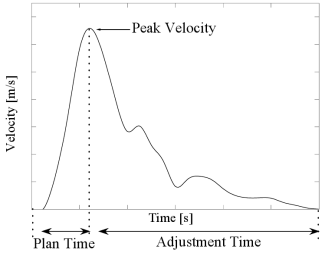
La loi de Fitts concerne l'environnement immédiat de l'Homme (et de l'animal). Limitations:

- Elle est fausse pour des grands D:
  - temps déplacement sur la terre linéaire en D
  - débrayage de la souris
- Elle est fausse pour de très petit W (e.g., $W < 4$ pixels) et pour des grands W (un W max $\sim$ 250 pixels)

Loi de Fitts: courbes distance/temps - vitesse



distance vs vitesse



temps vs vitesse

La courbe distance-vitesse (à gauche) est en forme de cloche.

La courbe temps-vitesse (à droite) possède un pic clair (mouvement d'approche rapide) puis des oscillations:

- on s'approche et on vérifie si on est sur la cible,
- on s'approche et on vérifie si on est sur la cible,
- ... etc.

Loi de Fitts: taux d'erreur

La loi de Fitts ne tient pas compte, à priori, du taux d'erreur. On demande, par exemple, aux participants d'une expérience d'être rapides et de ne pas faire d'erreurs. Typiquement on veut un taux d'erreur de l'ordre de 4%. Ceci n'est pas toujours possible ...

Pour résoudre ce problème on calcul une "effective width" W_e et une "effective distance" D_e à posteriori (pour chaque W et D):

$$W_e = 4.133 \times EcartType(click_x) \qquad D_e = mean(Dist)$$

- pas d'erreurs: $W_e < W$;
- balance erreur-vitesse: $W_e \sim W$;
- très rapide: $W_e > W$;

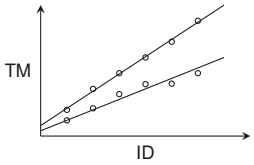
On espère alors avoir des a et b similaires dans les trois cas en considérant $\frac{D_e}{W_e}$

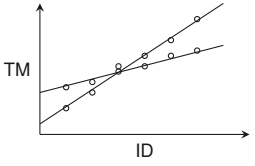
Loi de Fitts et IHM

Design d'interface (clavier logiciel, interface adaptée différent type d'utilisateurs, ...)

Technique de Pointage: réduire D ou agrandir W

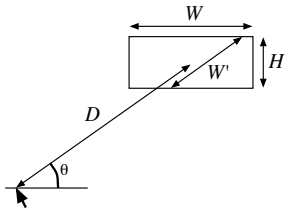
Comparer les performances entre périphériques d'entrée (souris, touchpad, tablette ...)





La pente de la droite comme un indice de performance

Pointage sur cible rectangulaire I



MacKenzie & Buxton 1992:

$$ID_{min} = \log_2\left(\frac{D}{\min(W,H)} + 1\right) \quad (1)$$

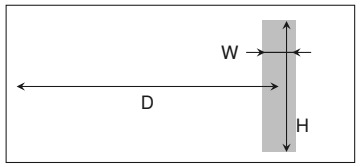
$$ID_{W'} = \log_2\left(\frac{D}{W'} + 1\right) \quad (2)$$

$$ID_p = \log_2\left(\frac{D}{(W^p + H^p)^{1/p}} + 1\right) \quad (3)$$

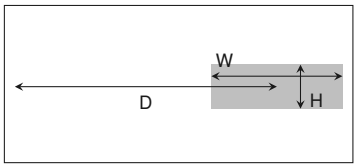
(1) meilleur que (2) ... on peut oublier (3).

Pointage sur cible rectangulaire II

La direction du mouvement d'acquisition est importante relativement à la forme de la cible. (Accot & Zhai 2003)



Contrainte en amplitude
(peu de contrainte angulaire)

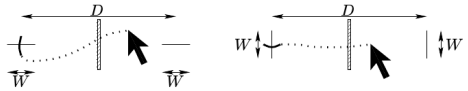


Contrainte angulaire
(peu de contrainte en amplitude)

Il est plus facile (à la souris) de contrôler une contrainte angulaire qu'une contrainte d'amplitude – au moins avec une souris. La tâche de gauche est plus difficile que la tâche de droite.

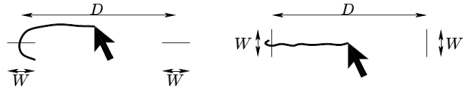
$$ID = \log_2\left(\sqrt{\left(\frac{D}{W}\right)^2 + \eta.\left(\frac{D}{H}\right)^2}\right) \quad \text{avec } \eta < 1 \text{ e.g. } \eta = 0.32$$

Franchissement (Crossing)



(c) ***D/CC*** — Discrete collinear goal-crossing

(d) ***D/OC*** — Discrete orthogonal goal-crossing task



(e) ***C/CC*** — Continuous collinear goal-crossing

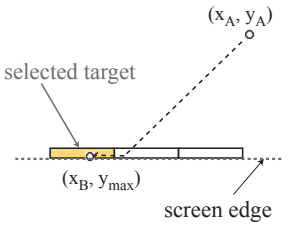
(f) ***C/OC*** — Continuous orthogonal goal-crossing

La loi de Fitts s'applique mais avec des paramètres *a* et *b* différents:

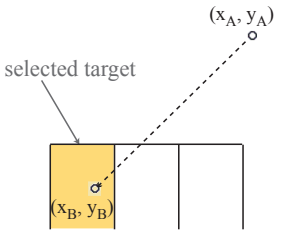
$$TM = a + b.ID \quad \text{où} \quad ID = \log_2(\frac{D}{W} + 1)$$

Pointage sur les bords I

Les bords de l'écran bloquent la souris dans la direction orthogonale aux bords.
Une cible sur le bord c'est comme une cible semi-infinie (Appert, Chapuis, Beaudouin, 2008)

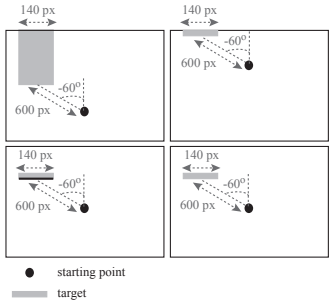


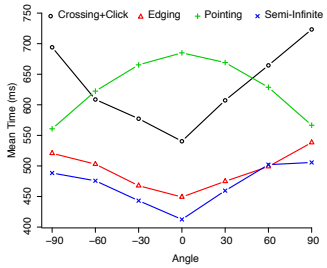
⇔



A priori les coins c'est le nirvana (cible de taille infini).

Pointage sur les bords





$$ID_{Angle} = \log_2\left(\frac{D}{W} + \frac{D}{H} + f(|Angle|).\frac{D}{min(W,H)} + 1\right)$$

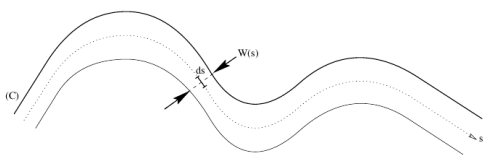
$f(|Angle|) = 0.6 \times sin(|Angle|)$ for *Edging* and *Semi-Infinite*

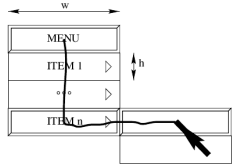
$f(|Angle|) = 0.6 \times cos(|Angle|)$ for *Pointing*

H infini $\rightarrow H = 250$ ...

Loi du mouvement canalisé (Steering law)

Déplacement dans un tunnel:





Temps de mouvement en fonction de C et W (Rashevsky, 1959; Drury, 1971; Accot and Zhai, 1997):

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

TM: temps de mouvement, C: chemin de navigation paramétrisé par s, $W(s)$: largeur du chemin en s, a et b des constantes empiriques. Ce qui peut donner (W(s) = W et $longueur(C) = D$):

$$TM = a + b.\frac{D}{W}$$

Attention: changement de direction et trajectoires courbes !

Loi de Hick-Hyman

Une loi qui modélise le temps qu'il faut à un utilisateur pour prendre une décision en fonction du nombre n de choix à sa disposition:

$$TR = a + b.\log_2(n+1)$$

Exemples:

- n cibles à l'écran, l'une "s'allume", il faut un temps linéaire en $\log_2(n+1)$ pour que l'utilisateur déclenche son mouvement vers cette cible.
- Trouver un item dans une liste de n items (e.g., menu):
  - Liste inconnue rangée de manière aléatoire: temps linaire en n
  - Liste rangée en ordre alphabétique: temps linaire en $\log_2(n+1)$
  - Après apprentissage: temps constant (?)

Menus hiérarchiques: Exercice

Modéliser le temps de pointage pour sélectionner un item dans un menu hiérarchique en utilisant la lois de Fitts, la loi du mouvement canalisé, la loi de Hick-Hyman avec un facteur d'apprentissage.

Plan

1 Modélisation du Temps de Pointage

2 Techniques de pointage

Techniques de Pointage

A priori il s'agit de "Battre la Loi de Fitts" en réduisant *D* et/ou agrandissant *W*.

Ces techniques peuvent être:

- ou bien "sémantique" (target-aware): besoin de connaitre les cibles,
- ou bien elles n'ont pas besoin de cette connaissance (target-agnostic)

Réduire *D*: Drag-and-Pop & Drag-and-Pick

(P. Baudisch et al. 2003)

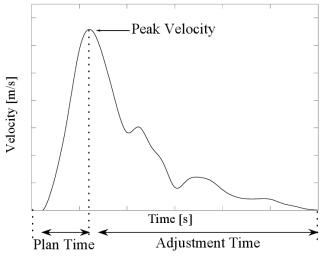
Drag-and-Pop: Lorsque l'utilisateur "drag" un icône, les icônes qui sont dans la direction du drag se rapprochent de cet icône pour faciliter le "drop".

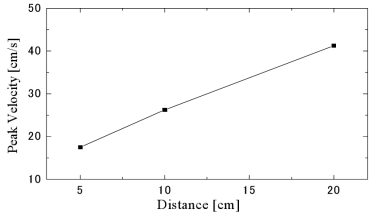
Drag-and-Pick: lors d'une sélection sur le fond les icônes qui sont dans la direction de cette sélection se rapprochent, l'utilisateur peut alors activer l'un de ces icônes en relâchant le drag sur un des icônes.

videos/pointage/DragAndPop.mpg

Réduire *D*: Delphian Desktop

(T. Assano et al. 2005)



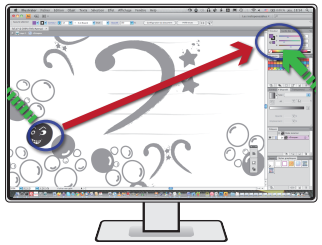


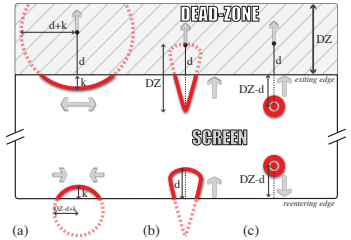
Le système utilise la vitesse au pic de vitesse pour prédire sur quelle cible l'utilisateur veut aller. Le système fait alors sauter le curseur sur la cible prédite ...

Difficile de croire qu'un telle technique puisse marcher ... Question: peut on prédire la position d'arrive lors d'un pointage ? (meilleur algo 20% d'erreur (Lank 2007) et ceci dans le cadre du laboratoire).

Réduire *D*: TorusDesktop

(Huot, Chapuis, Dragicevic 2011)



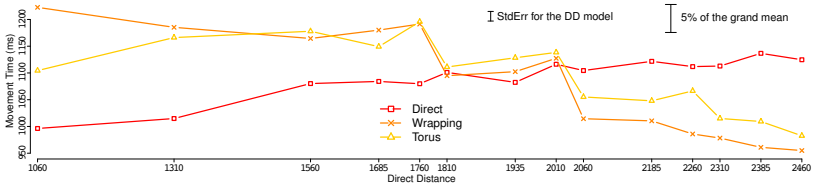


Idée naïve: lorsque le curseur arrive sur le bord le curseur saute au bord opposé. Problèmes: (i) activation involontaire et (ii) perte des bonnes propriétés des bords pour le pointage.

Solution: ajout d'une zone morte (e.g., 125 pixels) et d'un feedback pour montrer où est le curseur dans cette zone morte (3 feedbacks différents sur la Fig. de droite).

videos/pointage/torus-desktop.mov

Réduire *D*: TorusDesktop



Meilleur performance pour des distances $\leq$ 2010 pour un écran de 2560 de large (80% de la largeur de l'écran).

Il y a un coût pour le choix ($\sim 50ms$ utilisation ou non du wrapping) et pour le saut de curseur ($\sim 200ms$).

Mais la technique est "target agnostic" ... et implanté pour Mac OS X

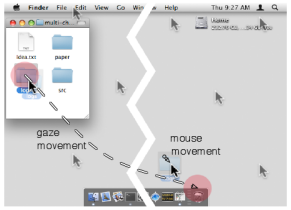
Réduire *D*: MAGIC & RakeCursor

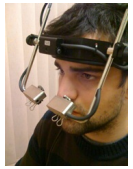
Utilisation d'un eye tracker pour pointer ... Problème on utilise les yeux pour faire bien d'autres choses ... aussi un bon eye tracker coûte cher ($\sim$ 20.000 euros).

Solution: Combiner la souris avec un eye tracker.

MAGIC (Zhai et al. 1999): déplacer le curseur à la position du regard lorsque l'utilisateur prend en main la souris.

RakeCursor (Blanch & Ortega 2009): utilisation d'une matrice de curseur et on sélectionne le curseur le plus proche de la position du regard lors d'un déplacement de souris. videos/pointage/rake-cursor.mov

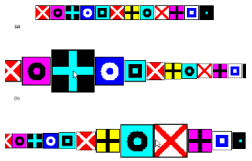




Agrandir *W*: Expanding Target

(McGuffin et al. 2002) Agrandir la cible lorsque:

- ou bien le curseur est sur la cible (agrandissement uniquement visuel)
- ou bien lorsque le curseur s'approche de la cible (pas toujours possible)



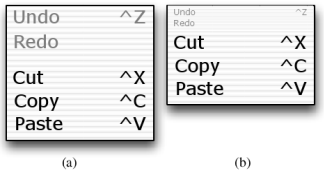
Démo: `http://profs.logti.etsmtl.ca/mmcguffin/research/expandingTargets/`

Agrandir W: Pointage Sémantique

(Blanch et al. 2004)

Agrandir la cible seulement dans l'espace moteur.

(a) design normal; (b) nouveau design espace visuel; (c) nouveau design espace moteur



(a) espace visuel ; (b) espace moteur

Problème: peut ralentir le pointage lorsque l'on passe sur des cibles agrandit dans l'espace moteur pour aller sur une autre cible (on agrandit D !).

Démo: `http://www.lri.fr/~chapuis/ihm-polytech/demos/semantic-pointing/`

Technique Extrême: Pointage Objet

(Guiard et al. 2004)

Une version extrême du pointage sémantique: au niveau moteur on supprime tout ce qu'il y a entre les cibles !

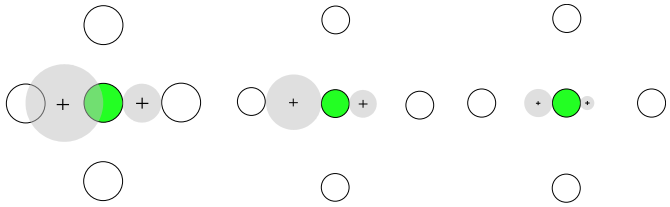
Lorsque l'on sort d'une cible on saute à la cible la plus proche dans la direction du mouvement.

Problèmes:

- Peut être très efficace mais les sauts du curseur sont problématiques
- On ne peut pas interagir avec le vide (e.g., sélection) il faut un mode

Agrandir la zone d'activation du curseur

(Worden et al. 1997)

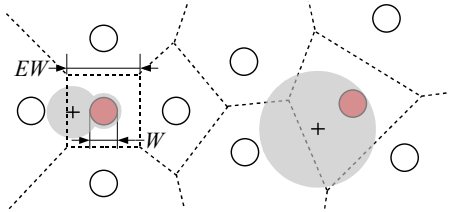


On agrandit la zone d'activation du curseur. Pour une cible isolée de taille W si le curseur est un rond de c pixels alors la cible à une taille $W + c$.

Lorsque le curseur intersecte plusieurs cibles on choisie ce qu'il y sous le curseur ou bien on choisit la cible la plus proche (mais dans ce 2ème cas problème pour interagir avec l'espace vide)

Technique Extrême: Bubble Cursor

(Grosmann et al. 2005)



Le curseur capture la cible la plus proche ! Peut on imaginer mieux ?

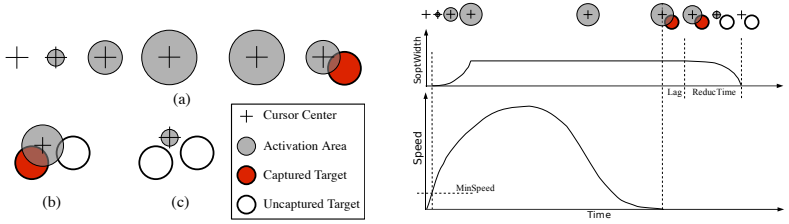
Il y a tout de même des problèmes

- Interaction avec l'espace vide ... il faut un mode ...
- Très efficace avec peu de cibles, mais dans ce cas le feedback peut être assez gênant.

videos/pointage/bubble-cursor.mov

DynaSpot: Speed dependant area cursor

(Chapuis, Labrune, Pietriga, 2009)



C'est un "area cursor" mais sa taille dépend de la vitesse du curseur (a). Permet d'interagir avec l'espace vide (c) et agrandit la taille des cibles lorsque l'on va vite (a,b). Une taille maximale pas trop grande, moins gênant que Bubble cursor.

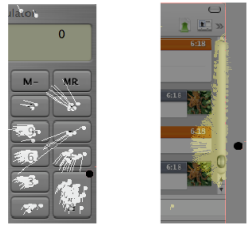
videos/pointage/dynaspot.mov

Dirty Desktop

(A. Hurst et al. 2007)

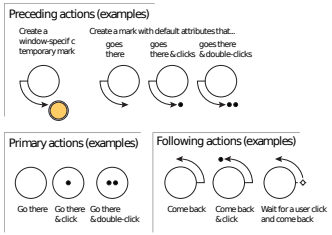
On enregistre les cliques sur les fenêtres et on applique le pointage sémantique aux endroits où l'utilisateur clique beaucoup.

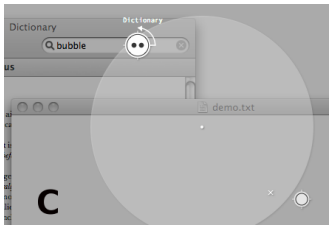
Technique adaptative ... le système essaye de faire quelque chose d'intelligent.



Être adaptable: UIMarks

(Chapuis & Roussel 2010) videos/pointage/uimarks.mov





L'utilisateur peut spécifier des cibles à l'écran avec des actions associées (langage graphique). Dans un mode particulier l'utilisateurs peut activer ces "marks" en utilisant Bubble cursor.

Les actions peuvent aller du simple pointage à des actions complexes qui nécessitent l'intervention de l'utilisateur: une marque sur une palette peut la rendre plus facilement accessible, mais la mark peut être configurée pour qu'une fois activé elle attende un click pour renvoyer le curseur à une ancienne position.