Leibniz Universität Hannover

Interactive Systems (ISY)

Auditorium Exercise 05

Human-Computer Interaction Group

Jan Feuchter
jan.feuchter@hci.uni-hannover.de

Lectures

Session	Date	Topic	Details
1	2.4.	Introduction	human performance, empirical research, modeling
2	9.4.	Interaction elements	input devices, interaction elements, states, layouts
3	16.4.	Event handling	events, bindings, reactive programming, scene graph
4	23.4.	Scene graphs	event delivery, coordinate systems, nodes, animation, concurrency
5	30.4.	Interaction techniques	alignment and pointing techniques
6	7.5.	Interaction techniques	
7	14.5.	Web-based user interfaces	document object model, client-server issues
	21.5.	Pfingstwoche	
8	28.5.	Web-based user interfaces	reactive Programming for the Web
9	4.6.	Experiments and data analysis	designing experiments, hypothesis testing
10	11.6.	Modeling interaction	descriptive and predictive models, keystroke-level model, regression
11	18.6.	Visualization	visual encodings, perceptual accuracy, treemaps, dynamic queries
12	25.6.	Human-Centered AI	introduction to human-centered AI, human control and automation, examples
13	2.7.	Deep learning in HCI	guidelines for human-AI interaction, neural networks
14	9.7.	Deep learning in HCI	convolutional and recurrent NNs, face recognition, gesture recognition

What is an interaction technique?

- "An interaction technique is a way of using a physical input/output device to perform a generic task in a human-computer dialogue."

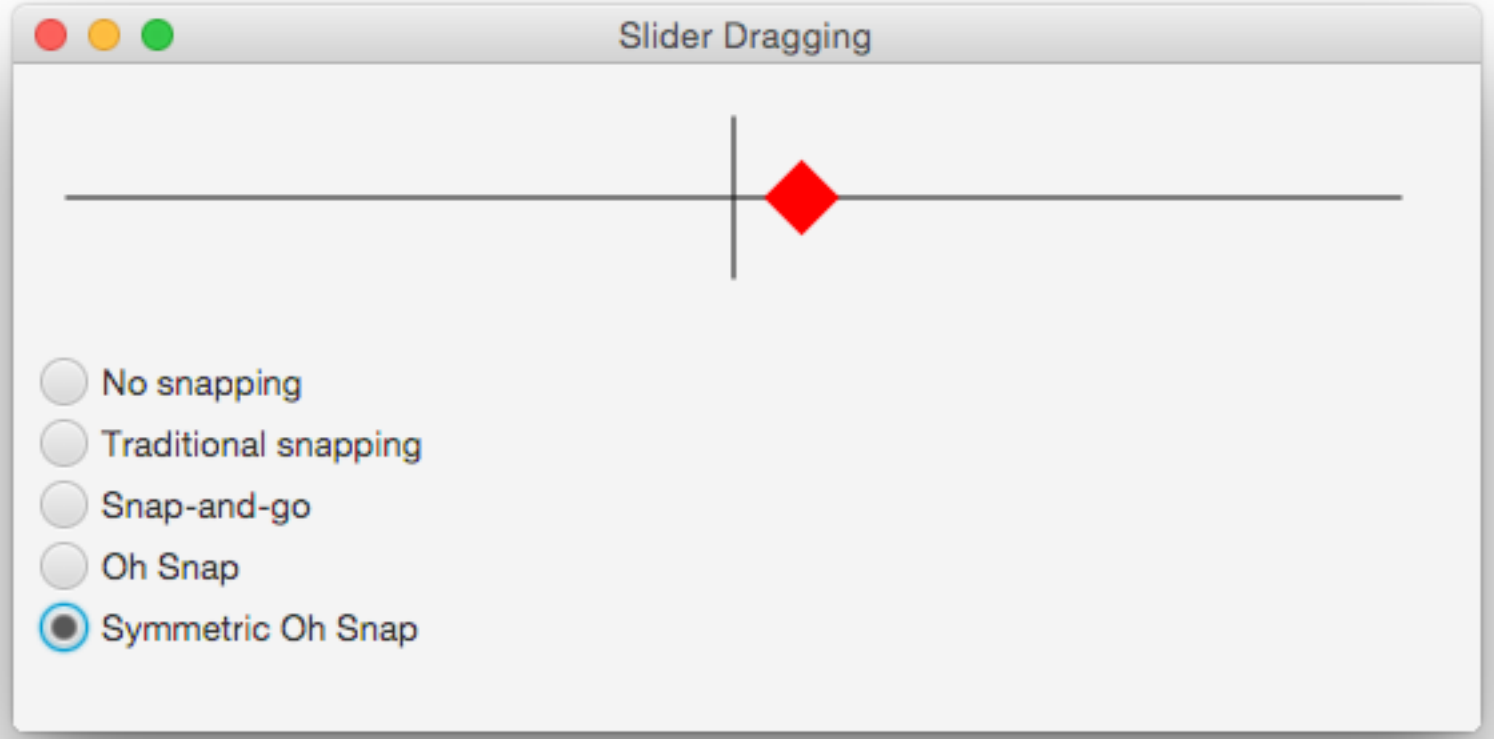
  Foley, van Dam, Feiner, Hughes. Computer Graphics: Principles and Practice, 1990.

- "An interaction technique is the fusion of input and output, consisting of all software and hardware elements, that provides a way for the user to accomplish a task."

  Tucker. Computer Science Handbook, 2004.

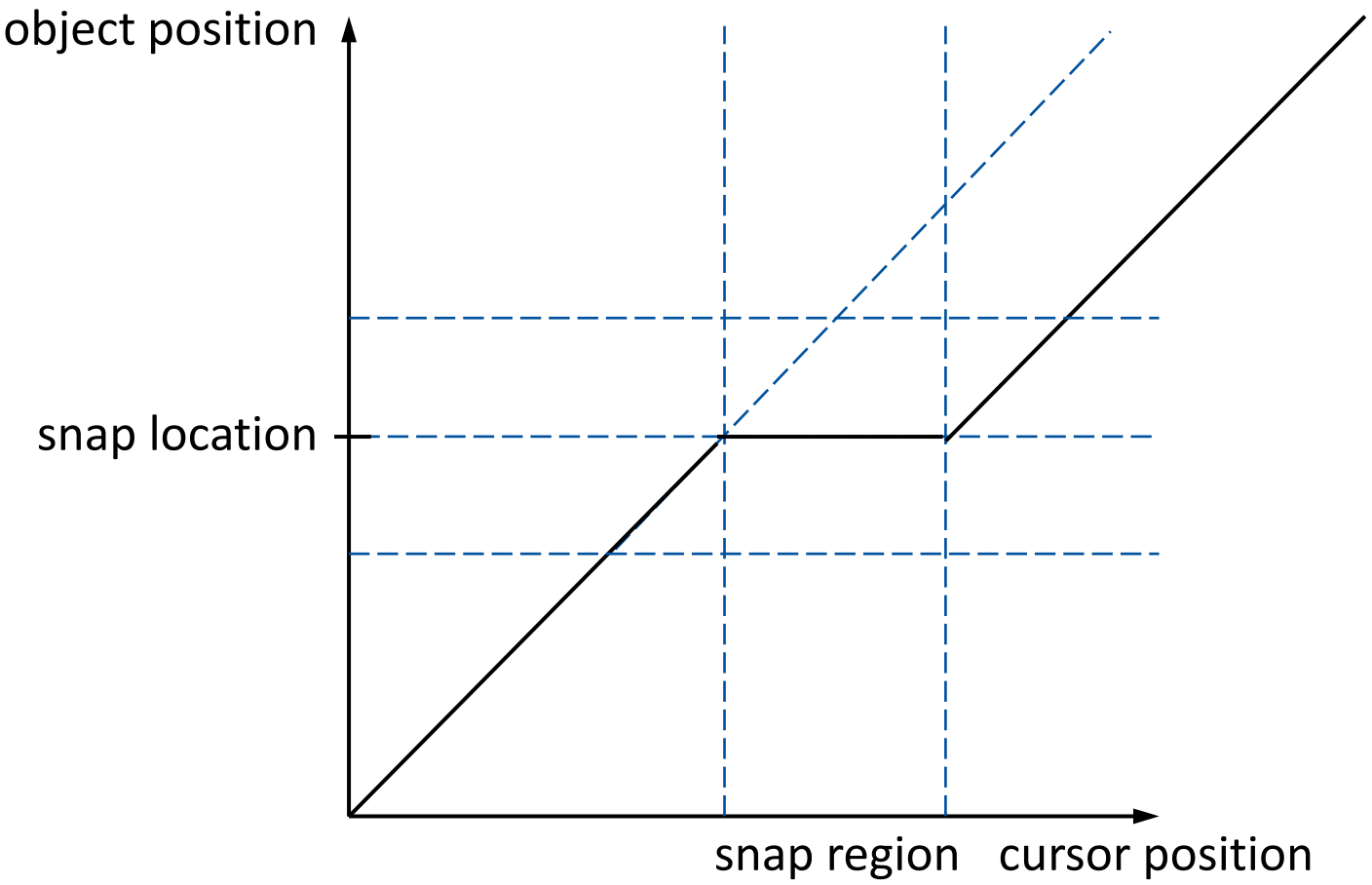
- An interaction technique is a solution to a specific user interface design problem. For example, snapping is a solution to the problem of precisely aligning objects.

Snapping Technique Demo: SliderDragging



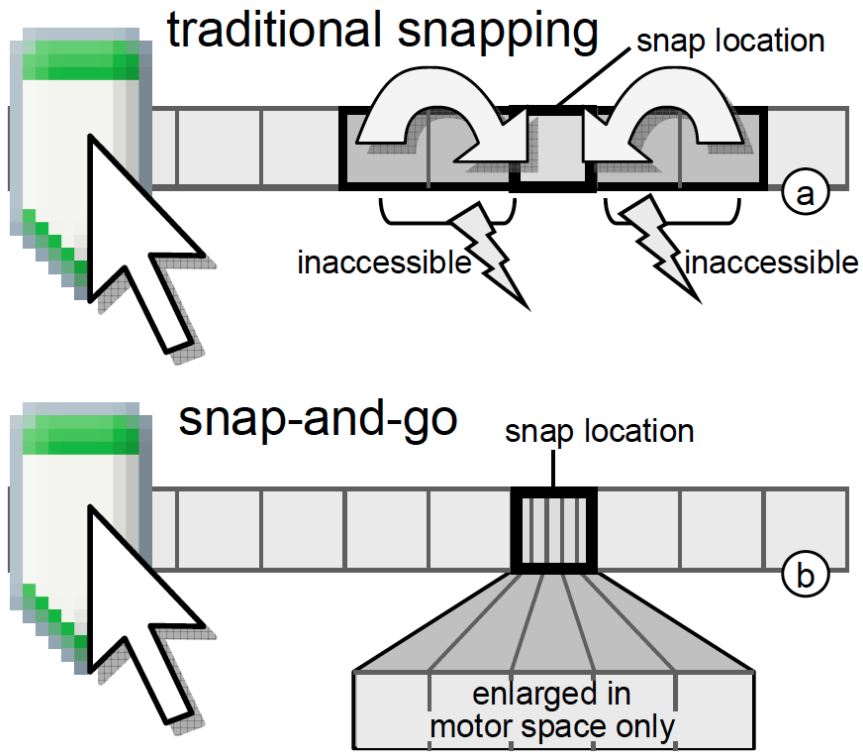
Stud.IP: SliderDragging.zip

Was zeigt das Diagramm?



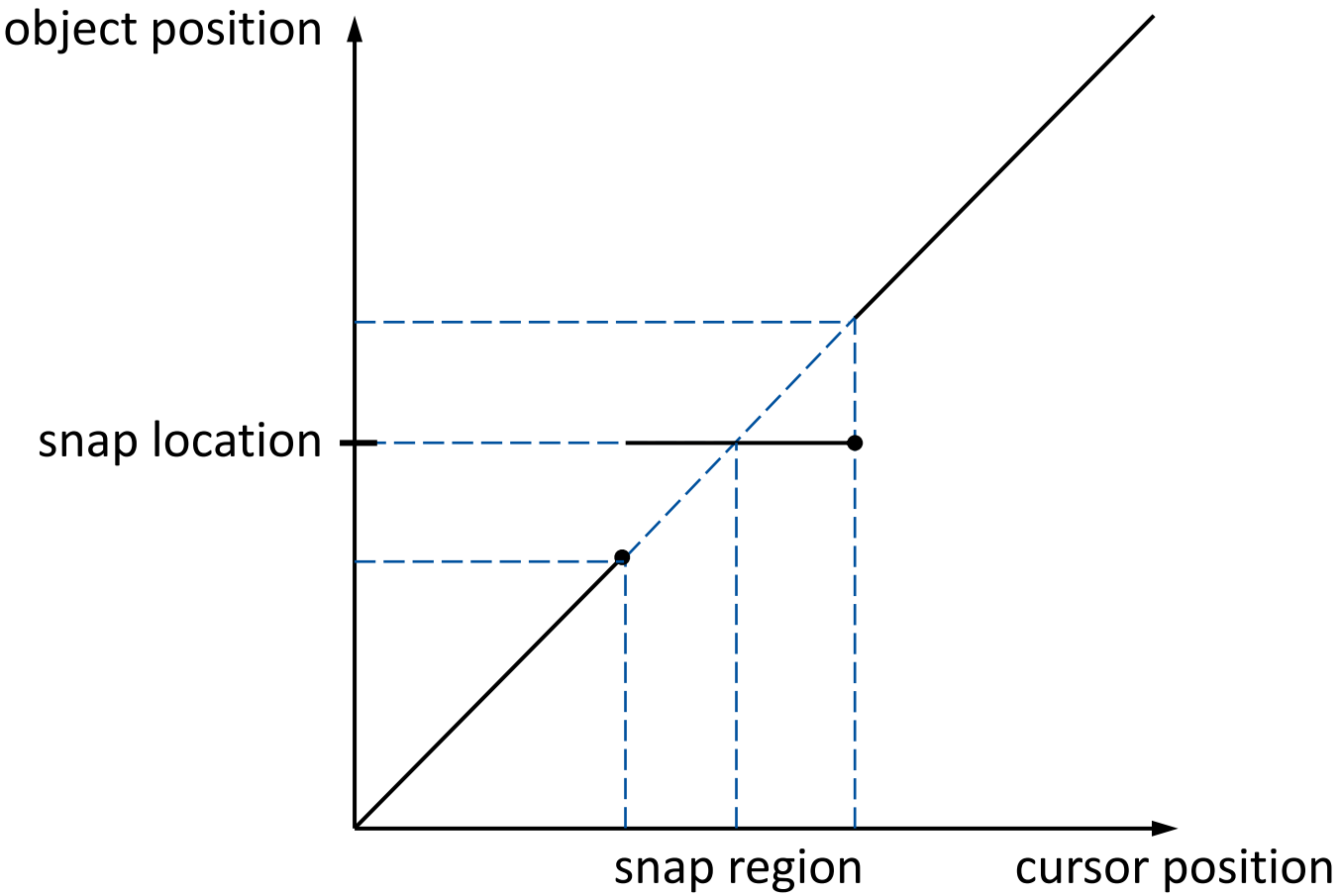
Snap-and-Go

- Traditional snapping warps objects to snap location
  - Area in close proximity of snap location is inaccessible
- Snap-and-go stops object at snap location
  - Inserts additional motor space at the snap location
  - Shifts the cursor position relative to the object



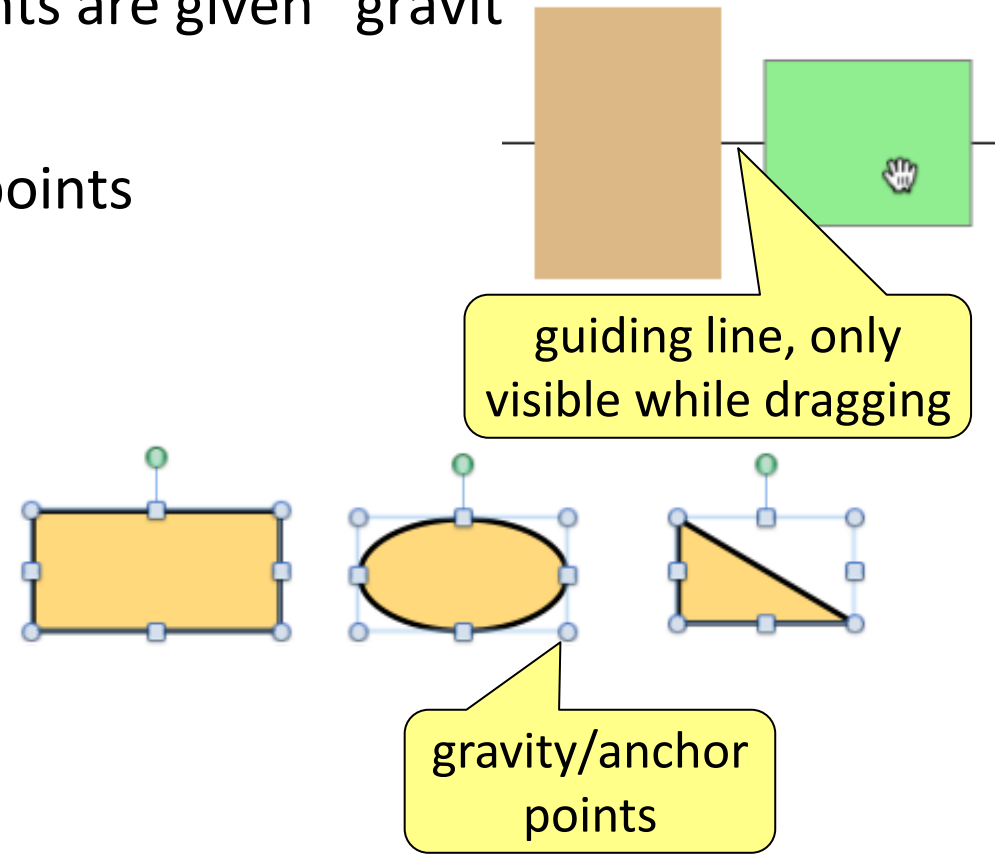
Baudisch, Cutrell, Hinckley, Eversole: Snap-and-go: Helping users to align objects without the modality of traditional snapping. CHI 2005.

Was zeigt das Diagramm?



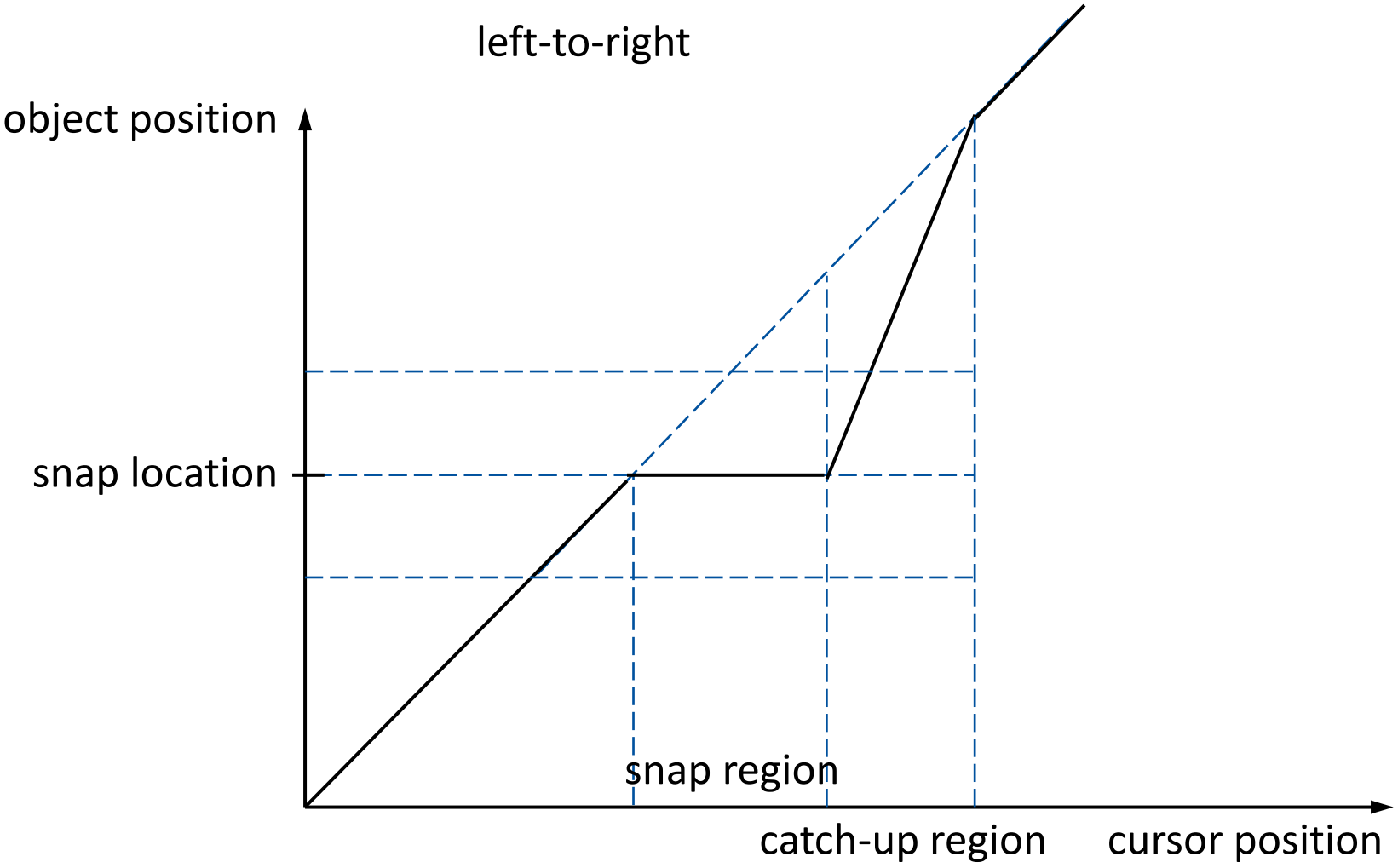
Snap Dragging: Precisely Placing Objects on a Plane

- Snap Dragging: When objects are dragged or stretched, positions that correspond to preferred ending points are given “gravit ”
- Dynamic guiding lines at specific object points
  - Only visible when object is dragging
- Warp (“snap”) objects to gravity points, when close enough
  - Makes common alignment operations simple
  - Requires suitable choice of anchor points



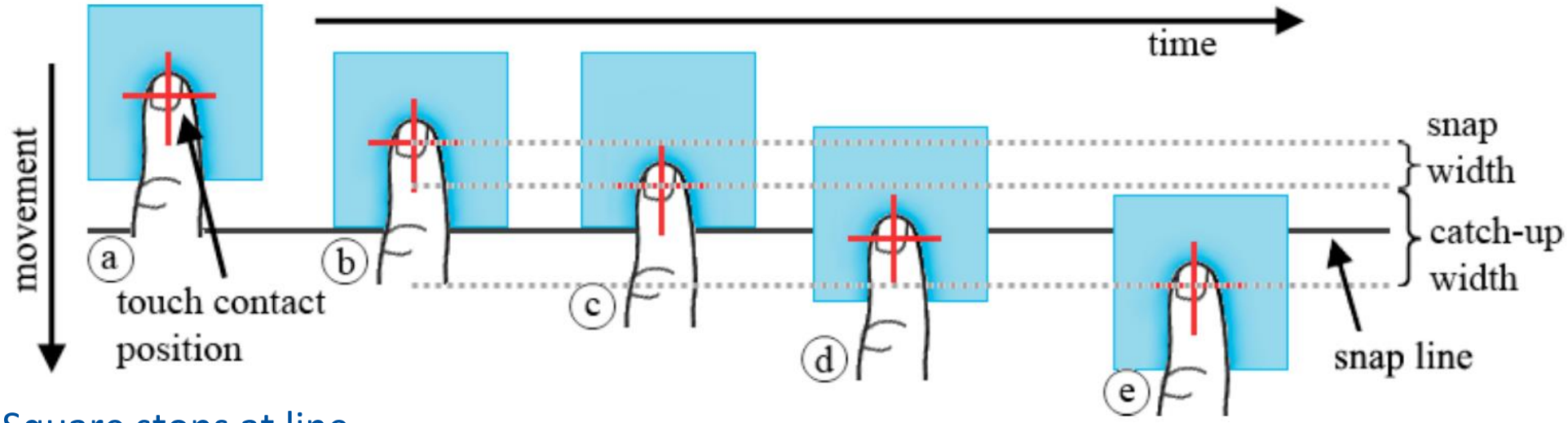
Bier, Stone: Snap-Dragging. SIGGRAPH 1986.

Was zeigt das Diagramm?



“Oh Snap”: Catching up After Snapping

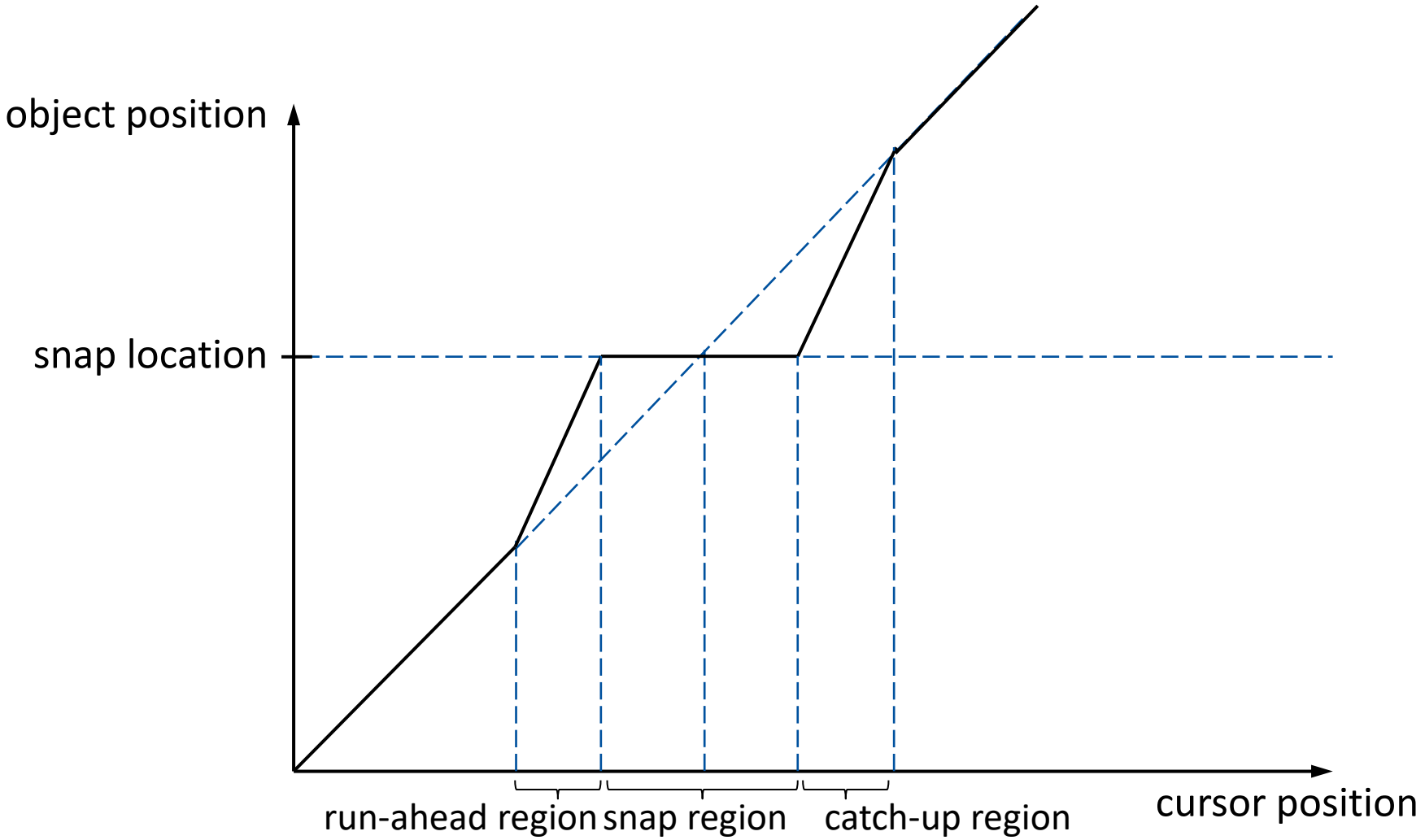
- Avoids object jumps towards snap location
- Avoids displacement by catching up after snap location



- Square stops at line
- Stays at line while finger movement continues through snap width
- Catches up while finger movement continues through catch-up width

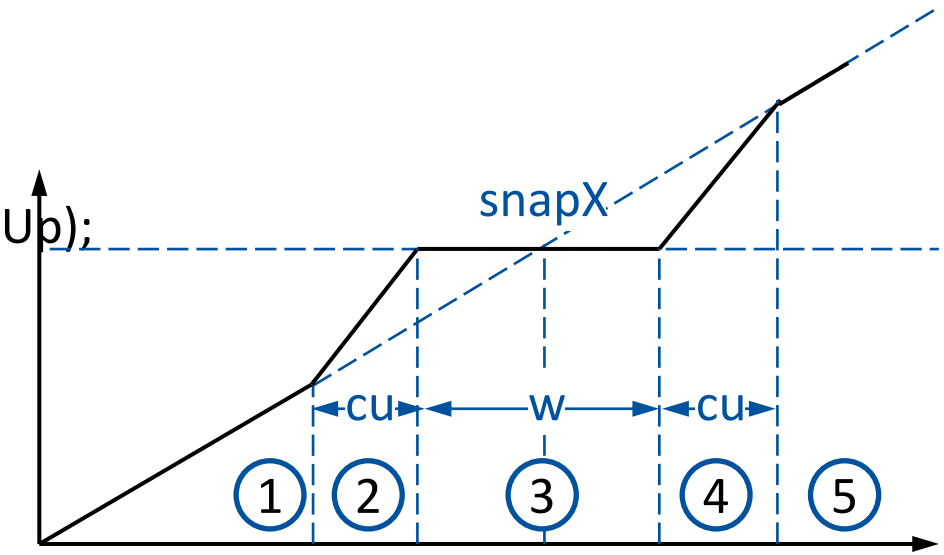
Fernquist, Shoemaker, Booth: ”Oh Snap” - Helping users align digital objects on touch interfaces. Interact 2011.

Was zeigt das Diagramm?



Symmetrical Snapping Implementation

```
double snap(double x, double w, double snapX, double catchUp) {
    if (x < snapX - w/2 - catchUp) {
        return x;                                          (1)
    } else if (x < snapX - w/2) {
        return snapX + (x - (snapX - w/2)) / catchUp * (w/2 + catchUp);   (2)
    } else if (x < snapX + w/2) {
        return snapX;                                      (3)
    } else if (x < snapX + w/2 + catchUp) {                (4)
        return snapX + (x - (snapX + w/2)) / catchUp * (w/2 + catchUp);
    }
    return x;                                              (5)
}
```



POINTING TECHNIQUES

Improvements over Point Cursors

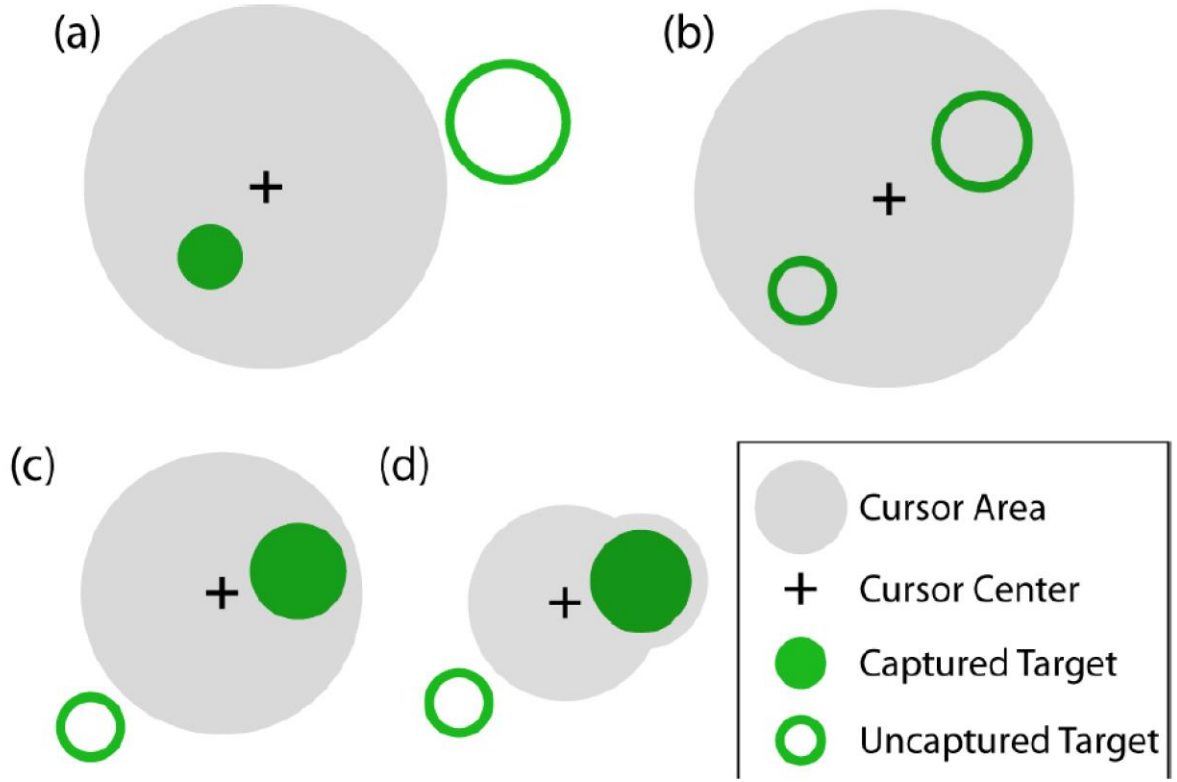
- Fitts' Law
  - Movement time MT depends on target width W and distance D
- Movement time (MT)
  - MT = a + b * ID [sec]
- Index of difficulty (ID)
  - ID = $\log_2(D / W + 1)$ [bits]

- In order to reduce MT
  - Increase target size (dynamically)
    - Expand active area of object or of cursor, increase motor space
  - Reduce distance (dynamically)
    - Bring cursor closer to target, or vice versa

Area Cursor

- Was sind die Unterschiede zwischen point und area cursor
- Wann ist ein area cursor vorteilhaft?

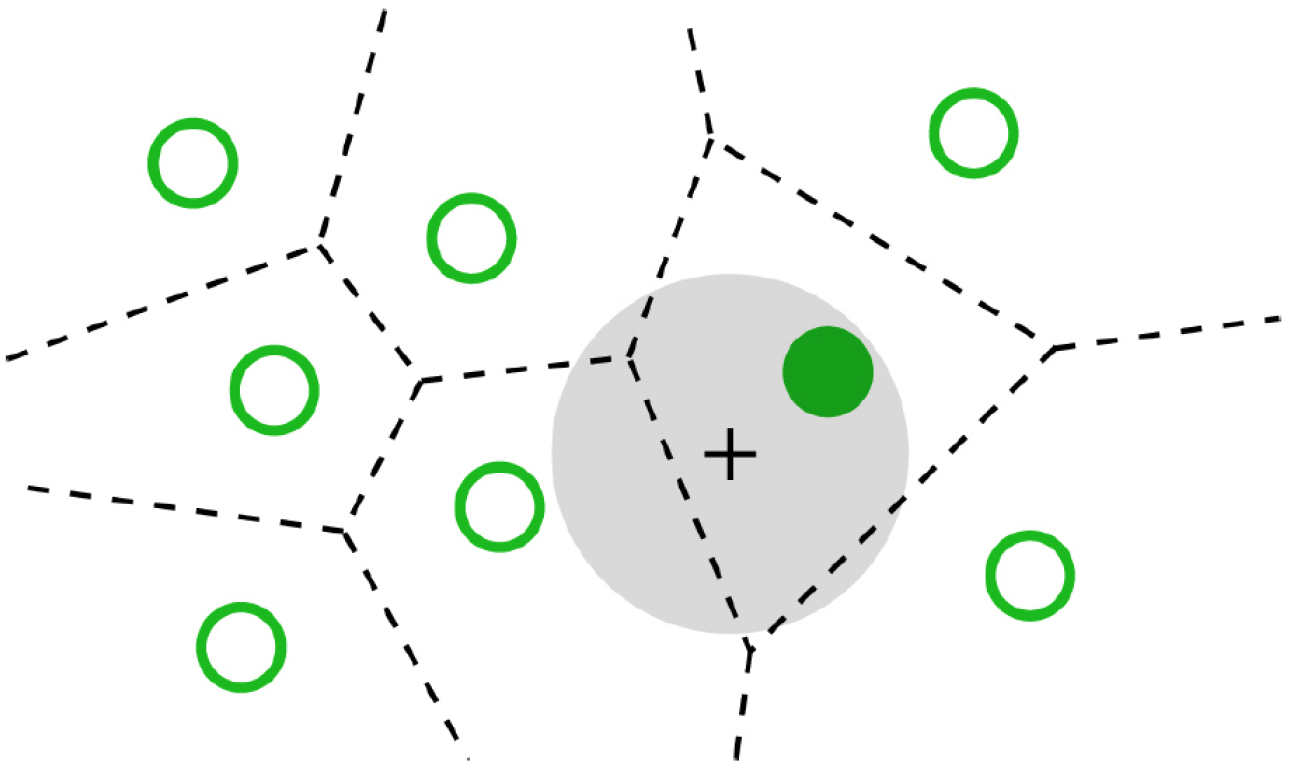
Bubble Cursor: Warum die Bubble?



(a) Area cursors ease selection with larger hotspots than point cursors.
(b) Isolating the intended target is difficult when the area cursor encompasses multiple possible targets.
(c) The bubble cursor solves the problem in (b) by changing its size dynamically such that only the target closest to the cursor centre is selected.
(d) The bubble cursor morphs to encompass a target when the basic circular cursor cannot completely do so without intersecting a neighboring target.

Grossman and Balakrishnan: The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. CHI 2005.

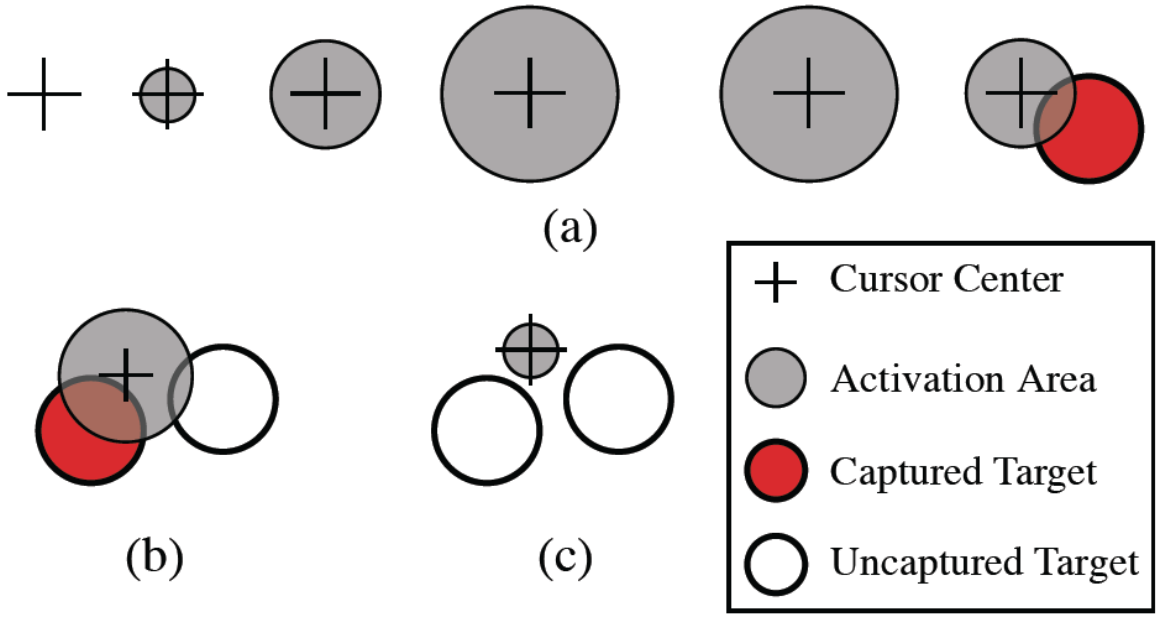
Bubble Cursor: Was zeigen die Voronoi-Zellen?



Grossman and Balakrishnan: The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. CHI 2005.

DynaSpot?

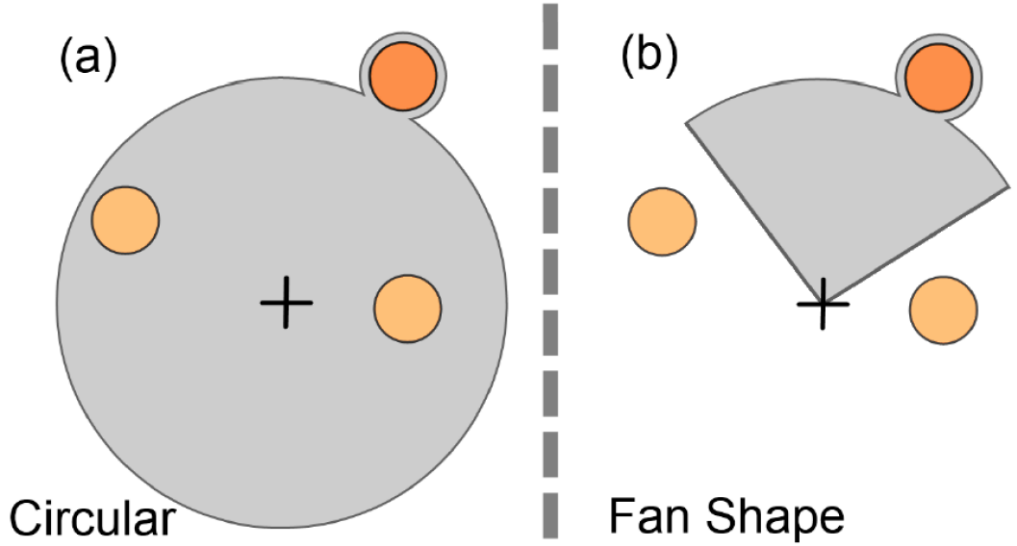
- Wie funktioniert DynaSpot?
- Welches Problem löst DynaSpot?

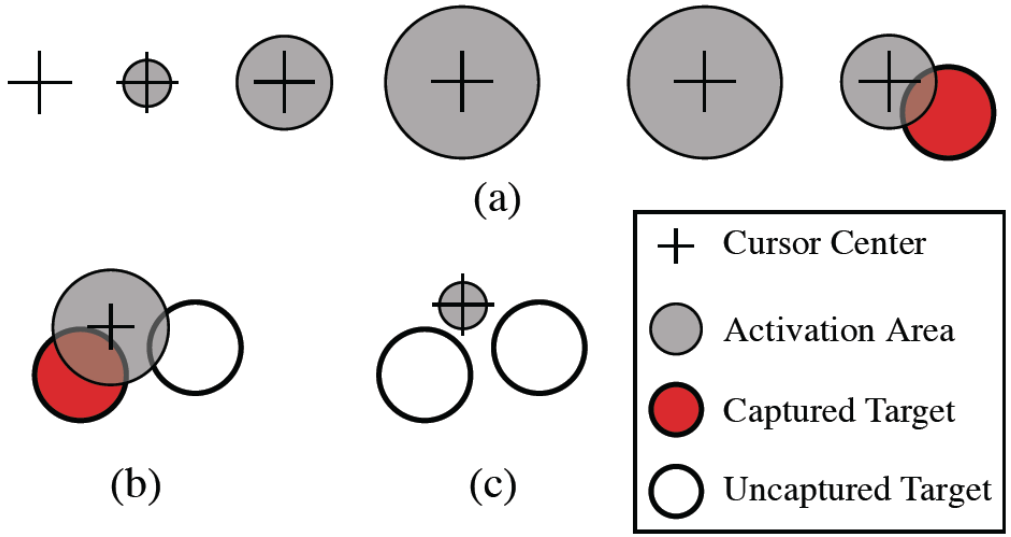


Chapuis, Labrune, Pietriga. DynaSpot: Speed-Dependent Area Cursor. CHI 2009.

Welchen Cursor würden Sie präferieren?

A. Bubble Cursor
B. Implicit Fan Cursor
C. DynaSpot





ASSIGNMENT 5

Assignment 5

- Deadline next Monday (06.05.22 23:59)