



# Virtual Reality & Physically-Based Simulation Interaction Metaphors

G. Zachmann

University of Bremen, Germany

[cgvr.cs.uni-bremen.de](http://cgvr.cs.uni-bremen.de)



# The First(?) Interactive Computer Program

- First really interactive, real-time computer game (arguably):
  - Spacewar, 1961, MIT
  - Two players, two spaceships ("wedge" and "needle"), each can fire torpedos
  - With it came the first real interaction devices and metaphors





# A Classification of Interaction Tasks

- *Basic interaction tasks (BITs) in 2D GUIs* [Foley / vanDam]:
  - Selection (objects, menus, ..)
  - Positioning (incl. orientation) or manipulation
  - Entering quantities (e.g., numbers)
  - Text input (via keyboard or speech input)
- *Universal Interaction Tasks (UITs) in VEs* [Bowman]:
  1. Navigation = change of viewpoint
  2. Selection = define object or place for next task
  3. Manipulation = grasp, move, manipulate object
  4. System control = menus, widgets sliders, number entry, etc.
    - Model and modify geometry (very rare; not in Bowman's UITs)

# More , Non-UIT Interaction Tasks

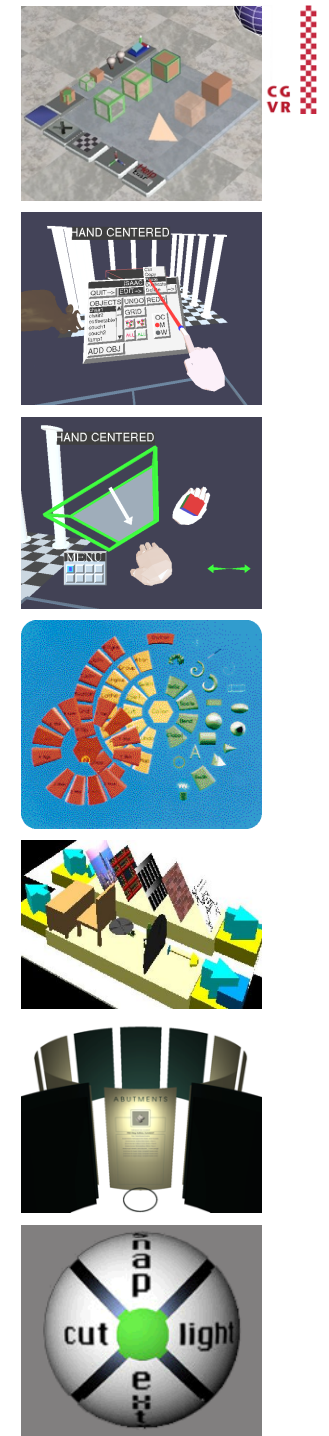


- Search
  - E.g., searching a scene for a specific object, excluding the navigation
- Ambient, implicit, playful, non-purposeful interaction
  - E.g., playing around with a virtual spraying can
- Sculpting / modeling surfaces
- Making an avatar dance by whole body interaction

# Digression: Classification of Widgets for 3D UIs

<b>Direct 3D Object Interaction</b>	
Object Selection	
Geometric Manipulation	
<b>3D-Scene Manipulation</b>	
Orientation and Navigation	
Scene Presentation Control	
<b>Exploration and Visualization</b>	
Geometric Exploration	
Hierarchy Visualization	
3D Graph Visualization	
2D-Data and Document Visualization	
Scientific Visualization	
<b>System / Application Control</b>	
State Control / Discrete Valuator	
Continuous Valuator	
Special Value Input	
<b>Menu Selection</b>	
Containers	

<b>Menu Selection</b>	
Temporary Option Menus	
	<i>Rotary Tool Chooser</i>
	<i>Menu Ball</i>
	<i>Command &amp; Control Cube</i>
	<i>Popup Menu</i>
	<i>Tool Finger</i>
	<i>TULIP</i>
Single Menus	
	<i>Ring menu</i>
	<i>Floating Menu</i>
	<i>Drop-Down-Menu</i>
	<i>Revolving Stage</i>
	<i>Chooser Widget</i>
	<i>3D-Palette, Primitive Box etc.</i>
Menu Hierarchies	
	<i>Hands-off Menu</i>
	<i>Hierarchical Pop-Up Menus</i>
	<i>Tool Rack</i>
	<i>3D Pie Menu</i>
	→ Hierarchy Visualizations



- There are two main approaches:
  - **Natural** interaction:
    - Try to resemble reality and the interaction with it as closely as possible
  - **"Magic"** interaction
    - Give the user new possibilities beyond reality
    - Challenge: keep the cognitive overhead as low as possible, so that users don't get distracted from their task!
- Tools:
  - Direct *user action* (e.g., tracking of the body, gesture, head turning, ...)
    - Pro: well suited if intuitive; con: possibilities are somewhat limited
  - Physical devices (e.g., steering wheel, button, ...)
    - Pro: haptic feedback affords precise control
    - Con: not easy to find/devise novel & useful devices
  - Virtual devices (e.g., menus, virtual sliders, etc., embedded in the VE)
    - Pro: very flexible, reconfigurable, "anything goes"
    - Con: can be difficult to use because of lack of force feedback



- Goals (in particular in VR):

1. Intuitive / natural interaction (**usability**)

- By definition: easy to learn
- Adjust to the user's expertise (**expert vs. novice**)

2. Efficient interaction (**user performance**)

- Precision, speed, productivity of the users

- Problems (especially in VR):

- No physical constraints (interaction in mid-air)
- In particular: no haptic feedback
- Efficient interaction with objects outside of the user's reach
- Noise / jitter / imprecision in tracking data
- Fatigue
- No standards

*There has never been a **high performance** task done in the history of this planet, to the best of my knowledge, that has ever been done well with an **intuitive** interface.*

[Brian Ferran]

- Is basically a simple classification problem:
  - Given: a **flex vector**  $x \in \mathbb{R}^d$ ,  $d \approx 20$  = joint angles
  - Wanted: pose  $G(x) \in \{ \text{“Fist“}, \text{“Hitch-hike“}, \dots \}$
- Wanted: an algorithm that is ...
  - .. user **in**dependent
  - .. robust (> 99%)
  - .. Fast
- General solution: machine learning algorithms (e.g, neural network)

# An Extremely Simple Pose Recognition Algorithm

- Neural network is fine, if lots of gestures, or some of them are inside the parameter space
  - However, experience shows: users can remember only a small set (e.g. 5)
- Consider only a few poses near the border of parameter space

- Discretize the flex vector

$$f \in [0, 1]^d \rightarrow f' \in \{-1, 0, +1\}^d$$

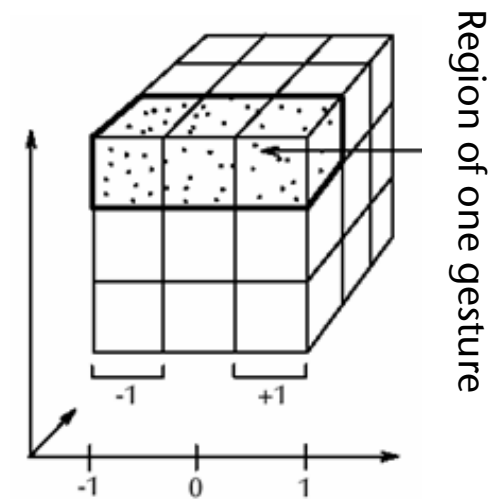
0 = flex value is "somewhere in the middle"

- Pose = a region of  $d$ -dimensional parameter cube
    - Represent each region/pose by a discrete vector:

$$g \in \{-1, 0, +1\}^d \quad 0 = \text{don't care}$$

- Recognize  $f$  as pose  $i \Leftrightarrow f'$  "matches"  $g^i$
      - $\Leftrightarrow \forall j : f'[j] = g[j]$

and ignore those  $f'[j]$  where  $g[j] = 0$



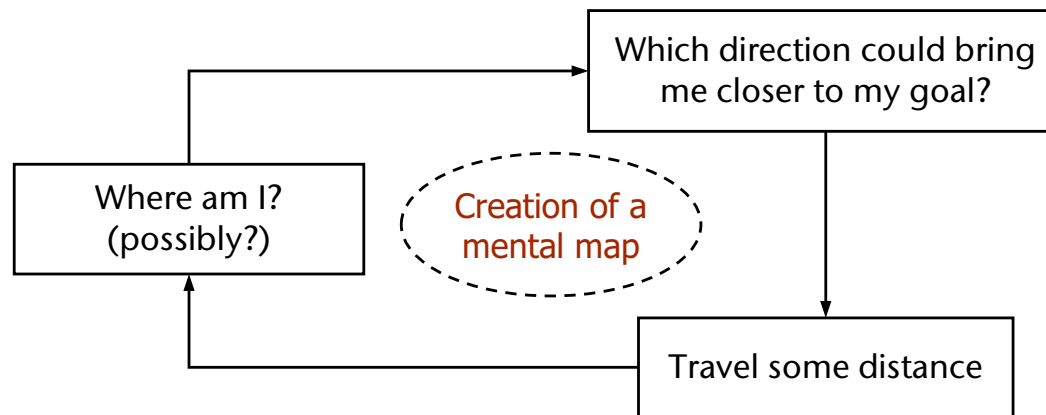
- Implementation details:
  - Do automatic calibration at runtime to fill the range [0,1]:
    - Maintain a running min/max and map it to [0,1]
    - Over time, shrink min/max gradually (for robustness against outliers)
  - Ignore transitory gestures
- Dynamic gestures =
  1. Sequence of static poses/postures (e.g., sign language)
  2. Path of a finger / hand
    - Utility for VR?



- Comprises: *Wayfinding* & *Locomotion*
- Locomotion / Travel =
  - Cover a distance (in RL or in VR)
  - Maneuvering (= place viewpoint and/or viewing direction exactly)
- Wayfinding =
  - Strategy to find a specific place (in an unknown building / terrain)
  - Comprises: experience, cognitive skills, ...

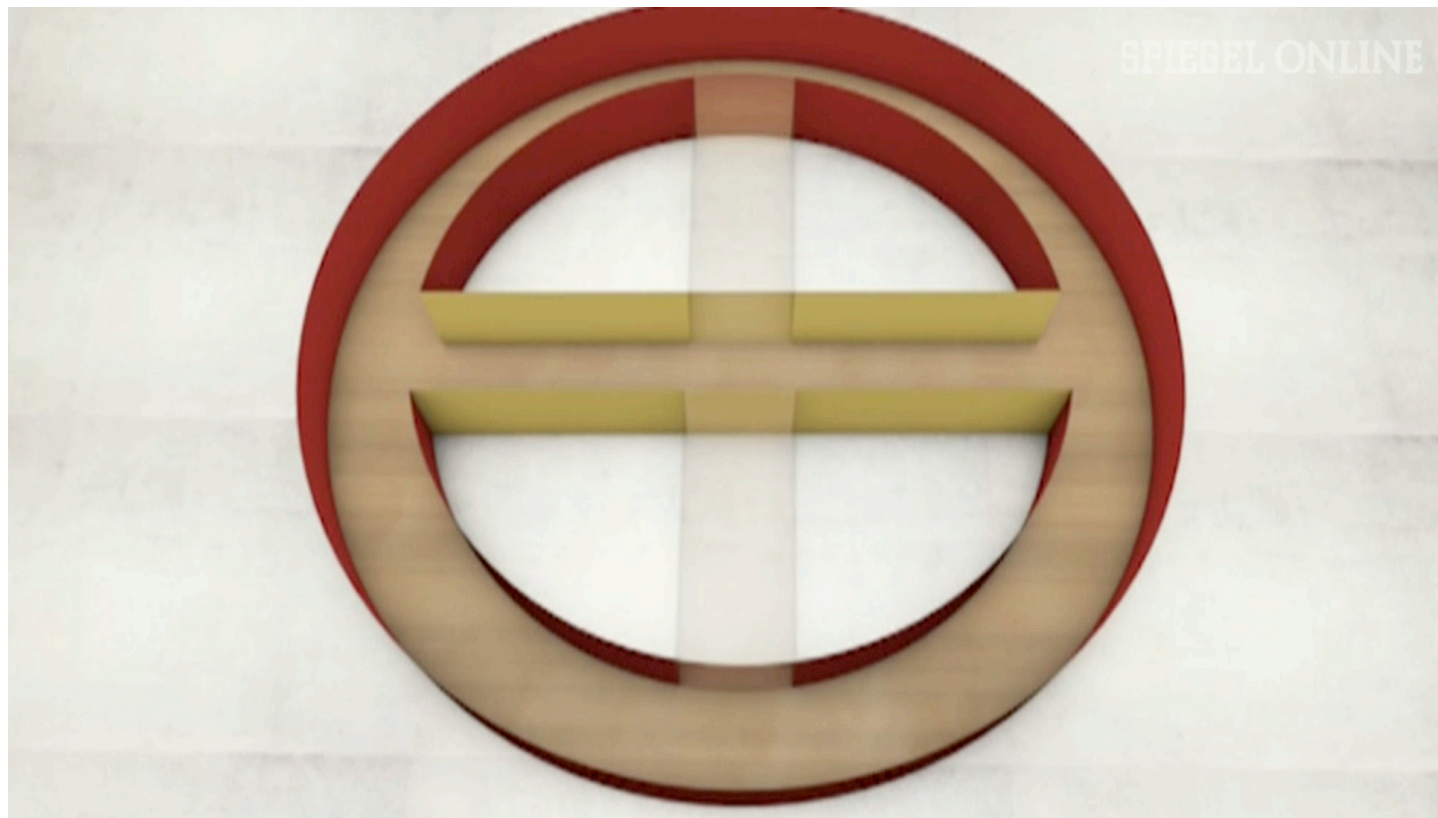
# How do People Solve a Wayfinding Task

- How do people find their way:
  - Natural hints/clues
  - Signs (man-made)
- A simple user model for way finding:



- In VEs, there can be two kinds of wayfinding aids:
  - Aids for improving the user's performance in the **virtual environment**
  - Aids that help increase the user's performance later in the **real world** (i.e., that increase the training effect)

- Question: do humans create a mental map of their environment in order to solve wayfinding tasks?
- Answer: probably yes, but not like a printed street map; rather like a **non-planar graph** that stores edge lengths



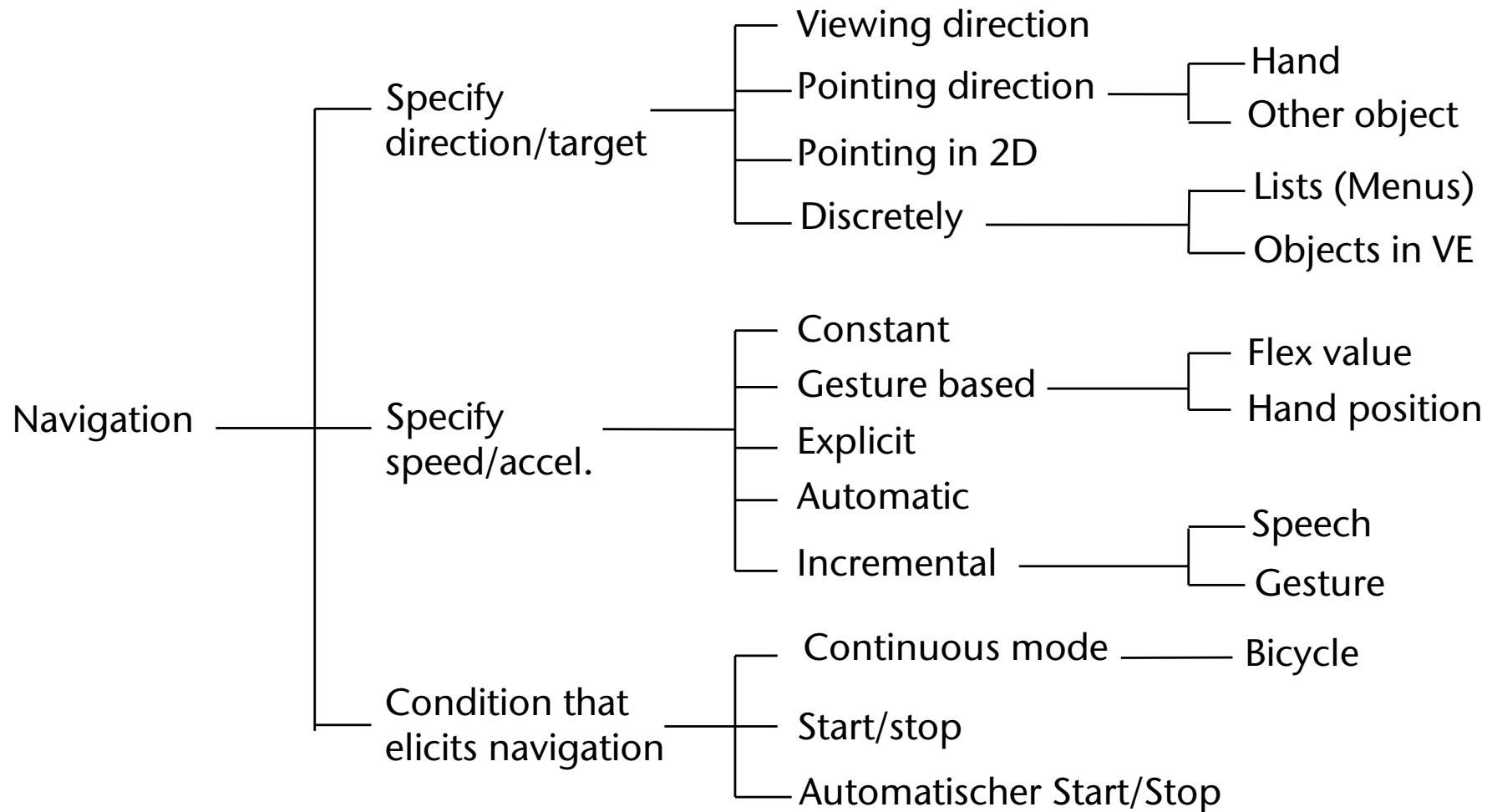
<http://www.spiegel.de/wissenschaft/technik/0,1518,739416,00.html>

Kerstin Schill, Neuro-Informatics, Uni Bremen

- A.k.a. "viewpoint control"
- Real user navigation, e.g., walking, turning head, ...
- *Point-and-fly* (especially in Caves and HMDs)
- *Walking in place*
- *Scene-in-hand*
- *World-in-Miniature*
- Orbital mode
- And some more ...



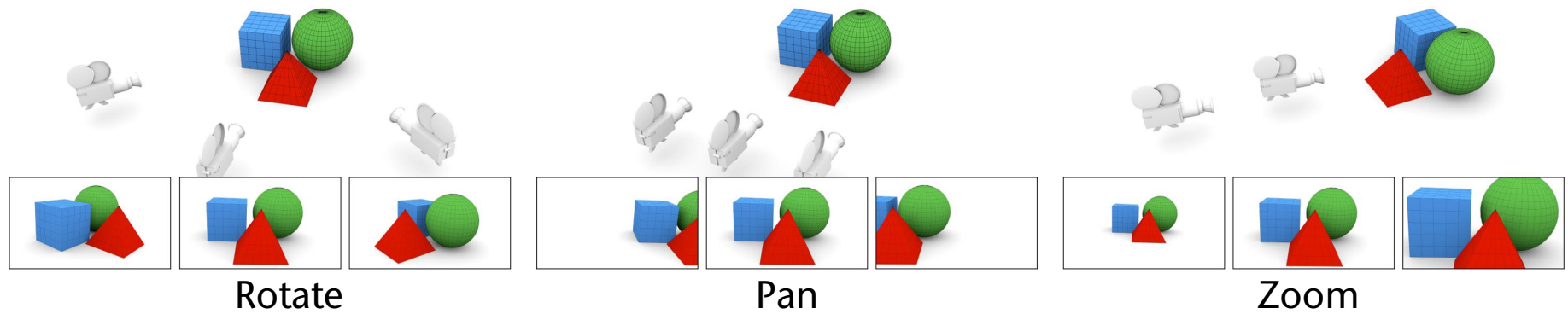
# A Taxonomy for this Interaction Task



- Taxonomies are a technique to explore the *design space* of an interaction task!

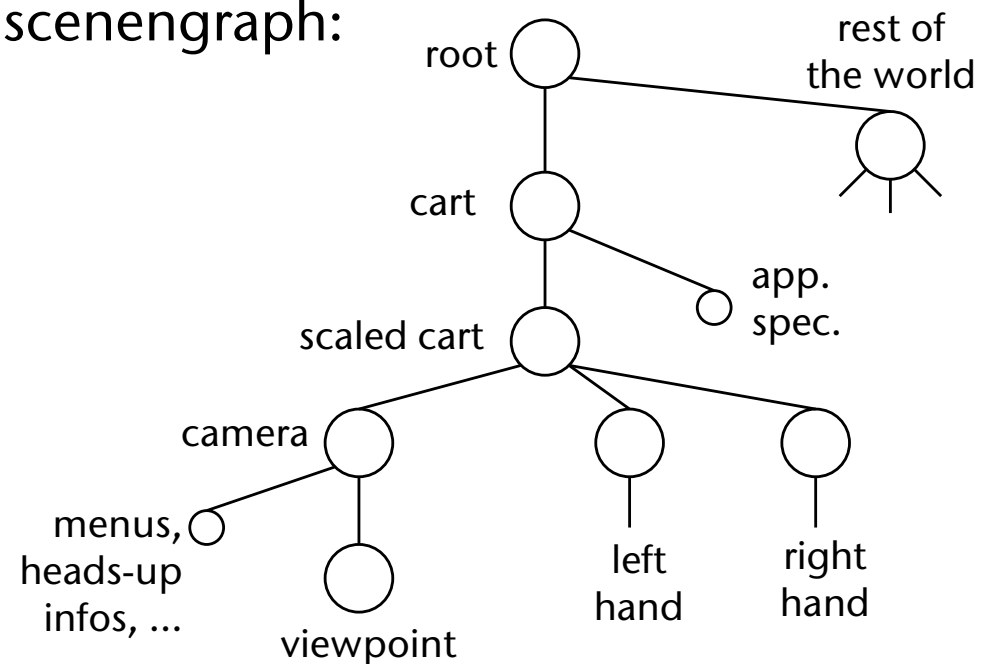
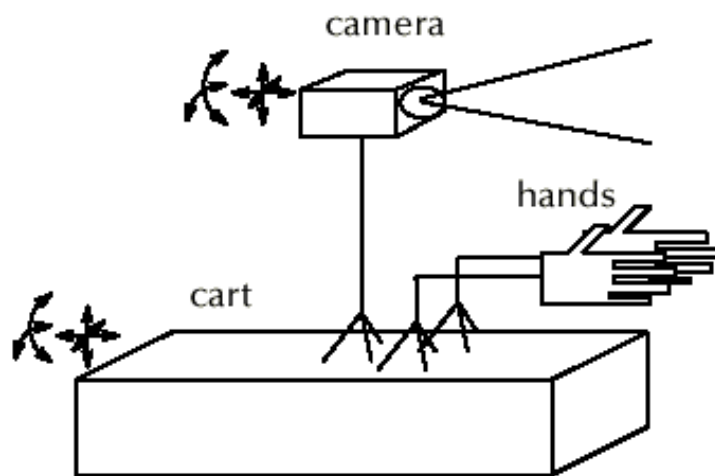
Interaction Task	Interaction Devices	Interaction Metaphor
Define scaling on tablet (e.g., for photo)	Touch screen	Pinch gesture
Define position on screen of desktop PC	Mouse	Move mouse on table, watch mouse pointer on screen, click
Start a timer	Hour glass	Turn hour glass
Change gears in car	Pedals and gear stick	First, push left pedal, move gear stick in desired position, release left pedal slowly
Switch room lights	Microphone	Clap hands three times

- Here, by the example of the Rotate-Pan-Zoom technique (a.k.a. Rotate-Scale-Translate):



# An Abstract Representation of the User

- User = head, hand, perhaps whole body (avatar)
- The "flying carpet" metaphor :
  - User = camera
  - Camera is placed on a carpet / cart / wagon
- Representation as (part of) a scenengraph:





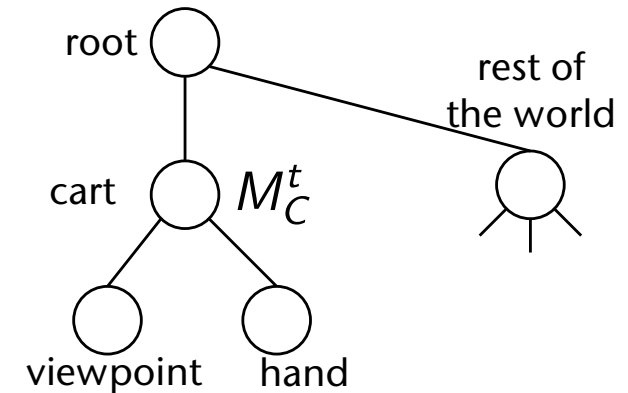
# The Point-and-Fly Metaphor

- Controlling sensors:
  - Head sensor → viewpoint
  - Hand sensor → moves cart:

$$M_C^t = M_C^{t-1} \cdot \text{Transl}(s \cdot \mathbf{t})$$

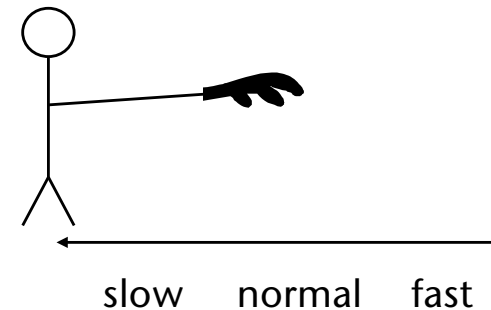
$s$  = speed,

$\mathbf{t}$  = direction, e.g., pointing direction of hand tracking sensor



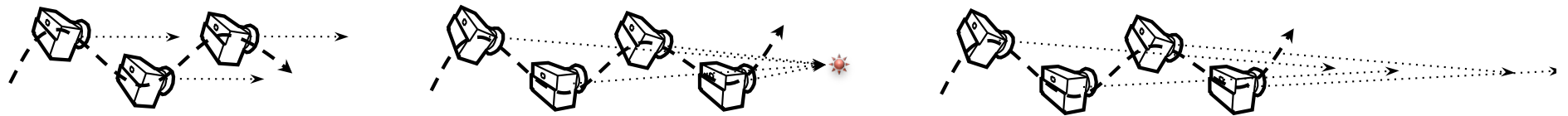
- Generalization: use graphical objects instead of sensor to derive translation direction (e.g., controller)

- Specification of the speed:
  - Constant (e.g. with Boom)
  - Flexion of the thumb
  - Depending on distance |hand – body|
  - Make it independent of framerate



# Perception of the Distance Travelled in VR

- Question: how can the sense of presence be increased while navigating in a VE? (using point-and-fly)
- Idea:
  - Make the viewpoint oscillate like in reality
  - (First-person-shooter games invented this earlier ;-)



- Results:
  - Only vertical oscillation helps increase presence
  - Users prefer slight oscillation over no oscillation
  - Short "travel distances" can be estimated more precisely (~ factor 2)

# The Scene-in-Hand / Eyeball-in-Hand Metaphor

## ■ *Scene-in-hand:*

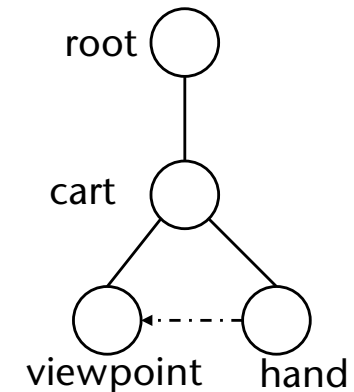
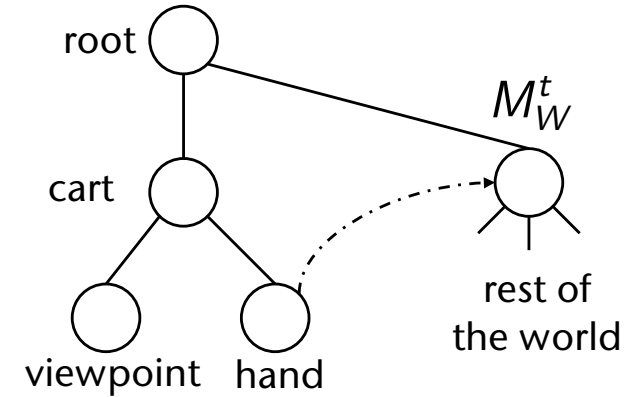
- "Grabbing the air" technique
- Cart remains stationary, scene gets rotated by hand sensor about a specific point in space
- The transformation:

$$M_{W}^t = M_H^t \cdot (M_H^{t_0})^{-1} \cdot M_{W}^{t_0}$$

- Instead of user's hand, use specific device, e.g. the CAT

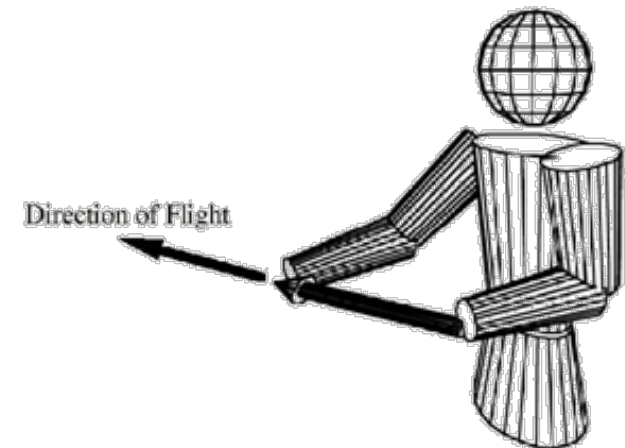
## ■ *Eyeball-in-hand:*

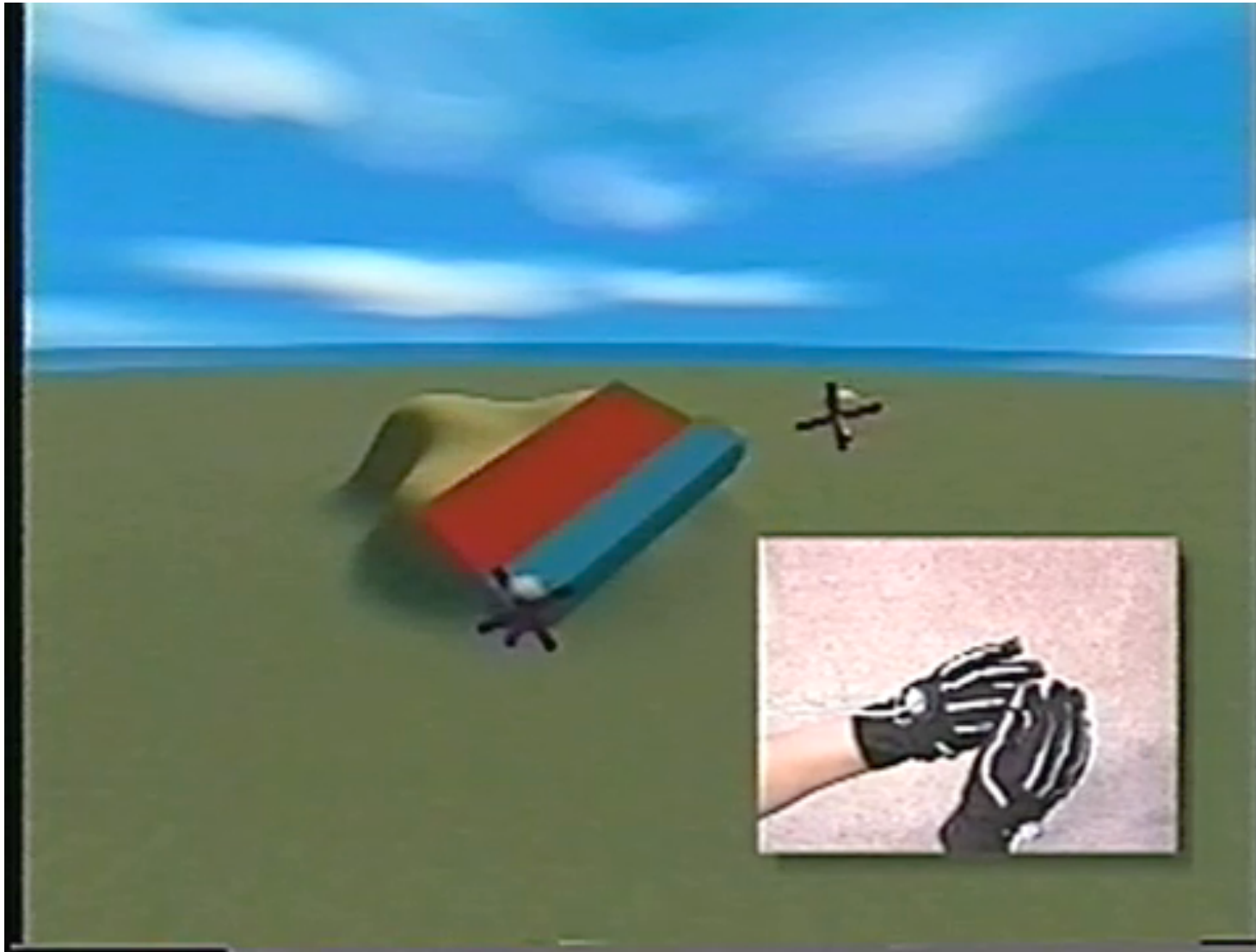
- Viewpoint is controlled directly by hand
- Can be absolute or relative (accumulating) mode



# Two-Handed Navigation (a.k.a. 3D Multitouch)

- Question: how to navigate with both hands?
  - Goals: increase input bandwidth, use simple input devices,
- Idea: we only need 2 points and 1-2 triggers (→ pinch gloves)
- Idea: use and modify "*scene-in-hand*" technique
  - 1 trigger, 1 moving point → translate the scene
  - 2 trigger, 1 fixed point, 1 moving point → rotate the scene
  - 2 trigger, 2 Punkte bewegt → scale the scene
- Not well-established in VR (probably because pinch gloves have not prevailed)
  - But: is the standard today on handhelds! ;-)
- Variation:
  - Direction = vector between both hands
  - Speed = length of vector

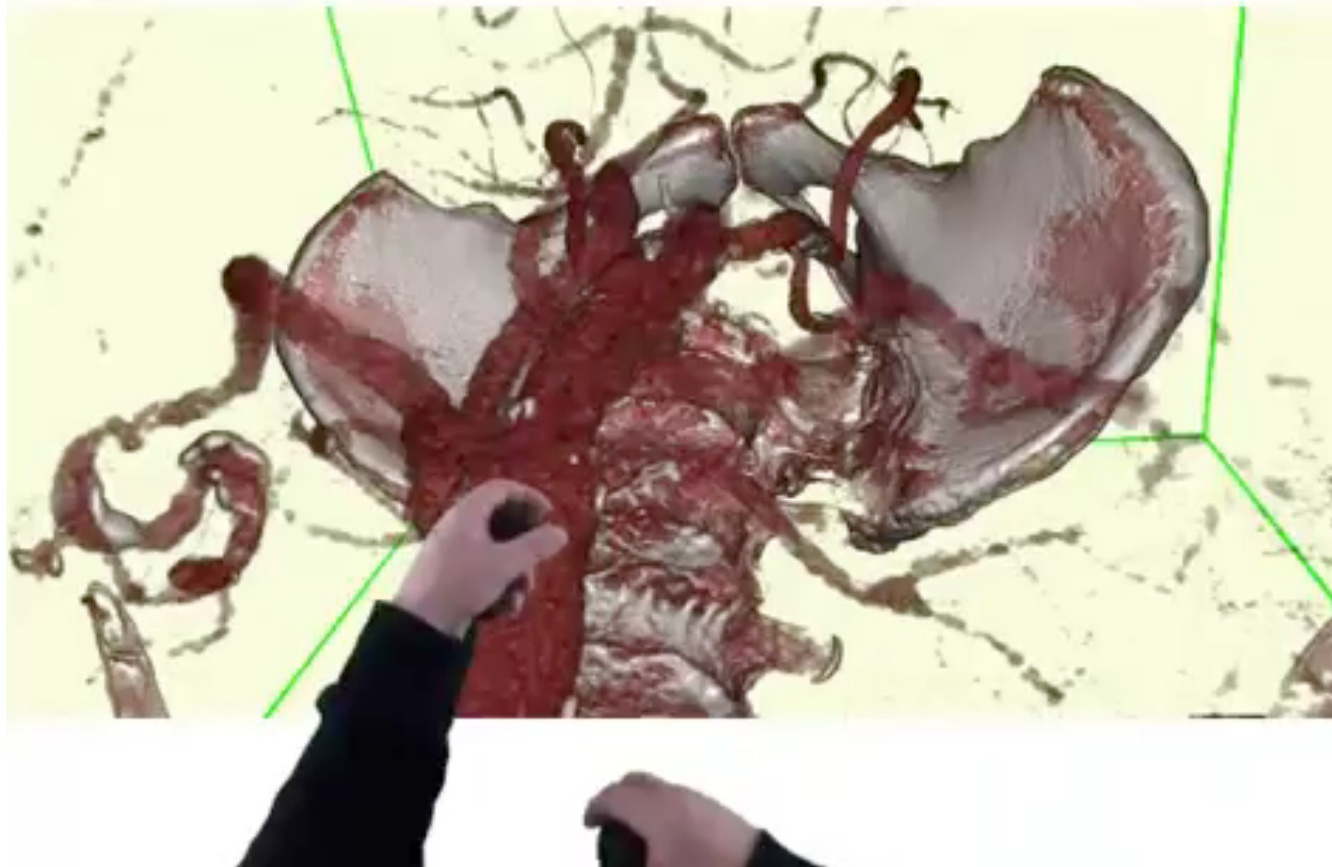




SmartScene, MultiGen, Inc.

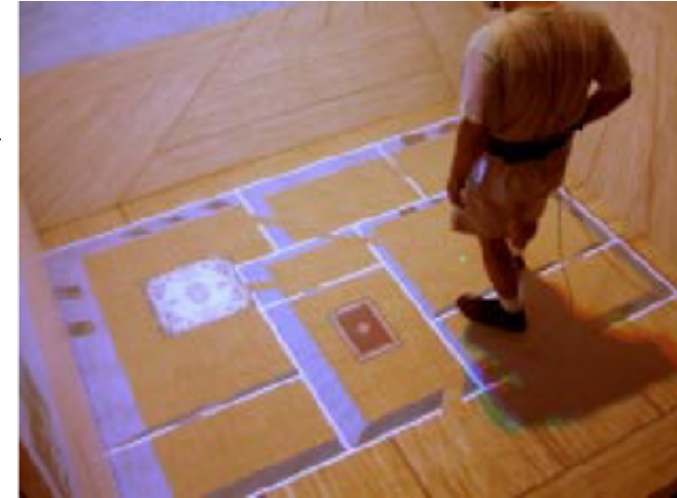
Digital ArtForms

# iMedic



# Real Walking on a Map

- Idea: project a scaled down version of the VE on the floor (map) and use feet
- Coarse navigation: teleportation → user walks to the new place/viewpoint on the map and triggers teleportation
- System commands involved:
  1. Bring up map = look at floor + trigger
  2. Teleportation = look at floor + trigger
  3. Dismiss map = look up + trigger
    - Trigger = speech command or "foot gesture"
- Accurate navigation: "lean" towards desired direction; speed = leaning angle





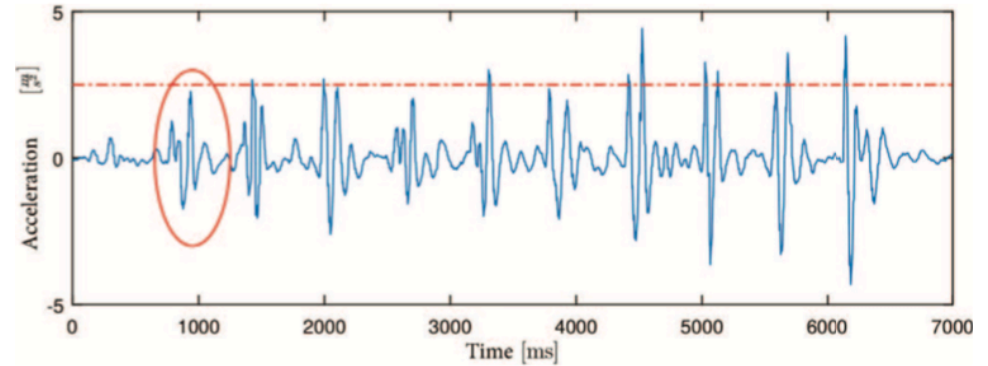
# Walking in Place (WIP)

- The technique:
  - Head of user is tracked (optical tracker, or accelerometer) → time series
  - Classifier recognizes walking pattern in time series (and speed)
  - Cart is moved in gaze direction
  - Only forward direction is possible
- Extension: **TILT-WIP**
  - WIP pattern just provides triggers
  - Tilting angle of head provides navigation direction (omnidirectional)
- Can increase the level of subjective presence in the VE
  - Reason is probably that proprioceptive information from human body movements better matches sensory feedback from the computer-generated displays
- Works only, of course, with a virtual ground (terrain, etc.)



# Detecting the Walking Motion

- Simple thresholding of accelerometer data from head
  - Project acceleration onto vertical axis
  - Step := acceleration > threshold

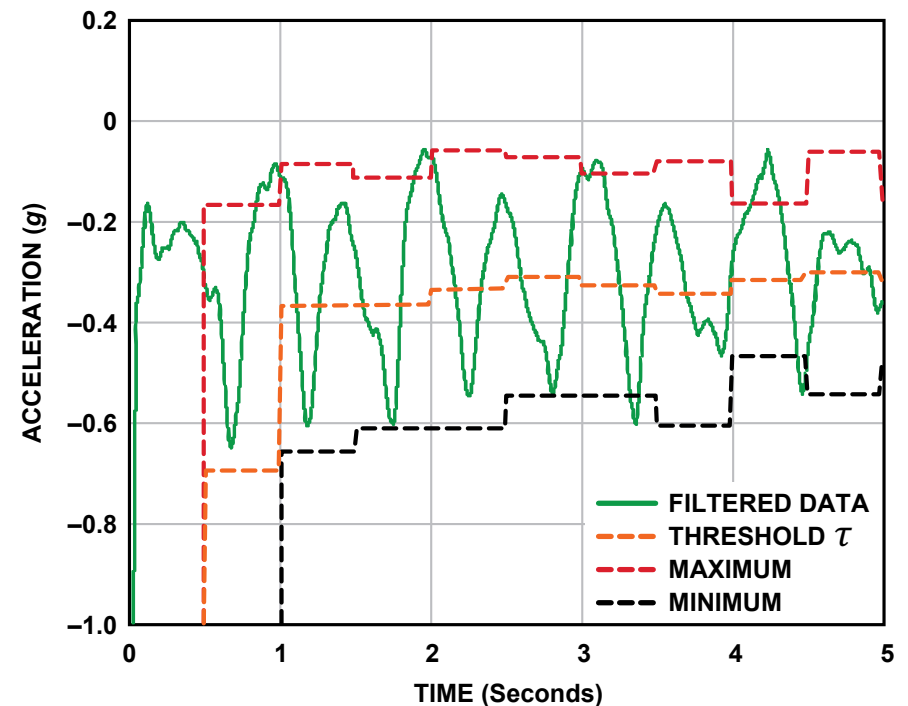


- More sophisticated thresholding:

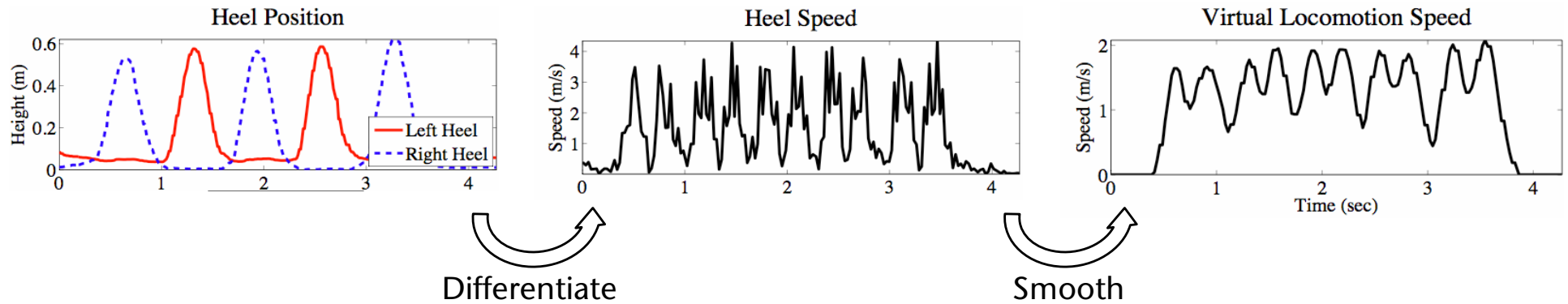
- Every 1/2 second, calculate min/max over last 1/2 sec of data
- At the same time, calculate threshold  $\tau = (\max + \min) / 2$
- With every new sample  $a_i$ , check

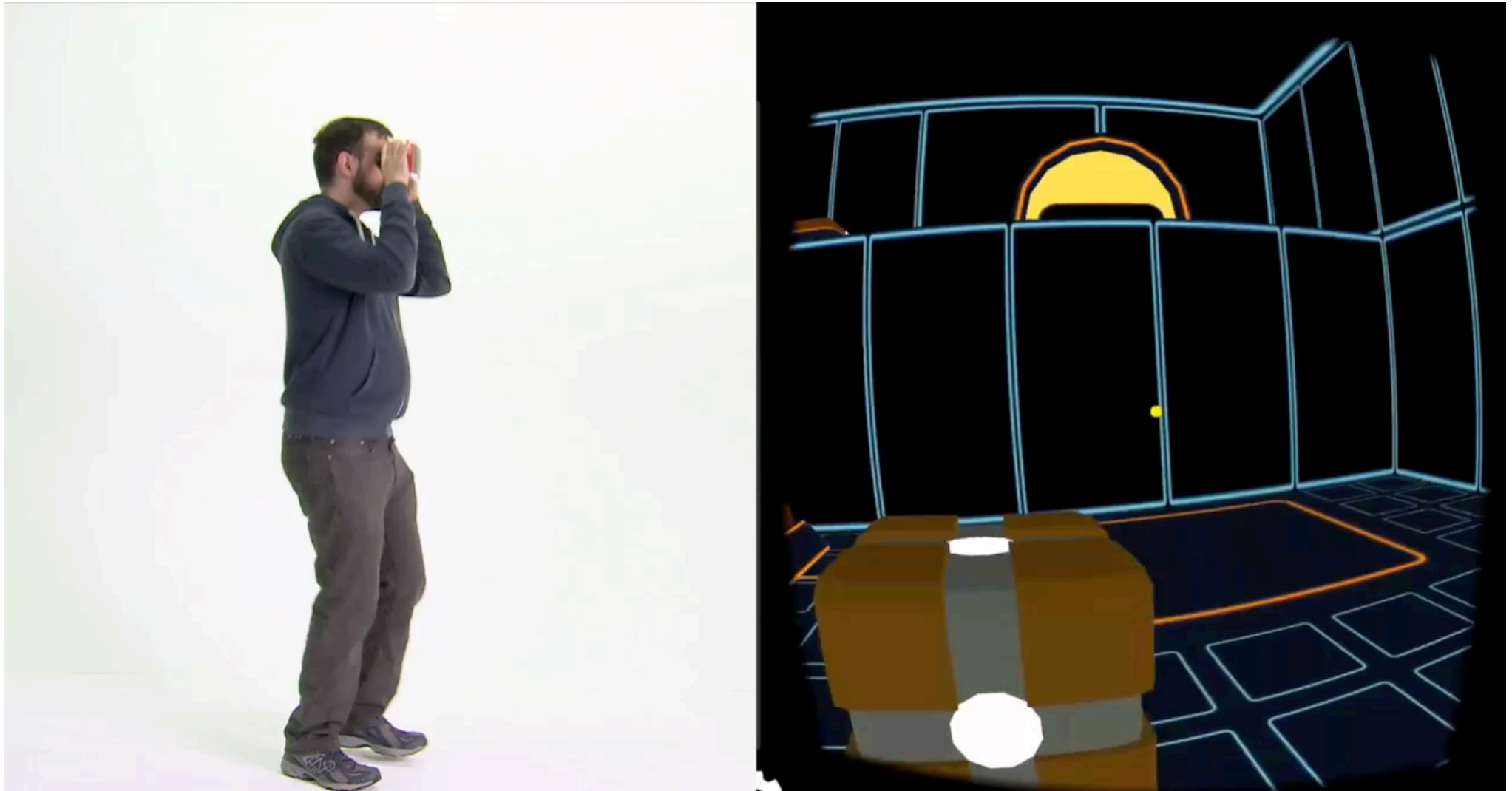
$$a_i < \tau \wedge a_{i-1} > \tau$$

⇒ step detected



- Alternative to head tracking: track the user's heels

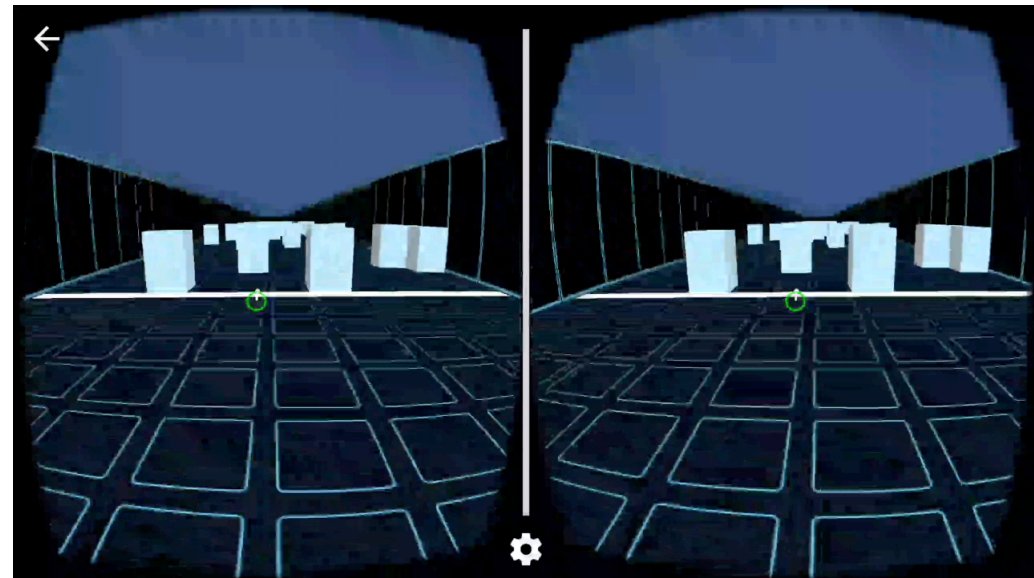
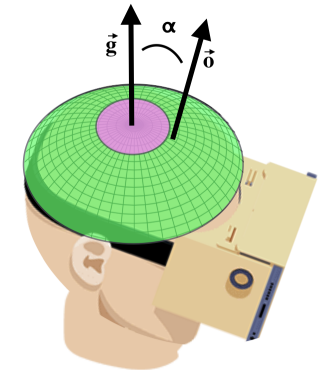




Eelke Folmer: VR-STEP

# Head Tilt Based Navigation

- Idea: angle between head's vertical axis and gravitational "up" vector determines navigation speed (if bigger than threshold)
  - Project vector  $\mathbf{o}$  onto XZ plane  $\rightarrow$  navigation direction
  - Disadvantage: navigation cannot be uncoupled from gaze direction
- Could be combined with walking-in-place
  - Walking = trigger , head tilt = direction
  - Allows for omnidirectional navigation (sideways, etc.)



- Task/goal: present a second viewpoint (like inset in an image) intuitively in a VE, and allow for its manipulation
- Idea: use the mirror as a metaphor → "magic mirror"
  - One object serves as hand mirror (could even look like it)
  - Keeps a fixed position **relative to camera** (follows head motions)
  - Can be manipulated like any other object in the VE
- Additional features (not possible with real mirrors):
  - Zooming
  - Magnification / scaling down of image in mirror
  - Clipping of objects in front of mirror (which occlude mirror)
  - "*Un-mirror*" scene visible in mirror ("Richtig-herum-Drehen")
  - Switch between main viewpoint and mirror viewpoint

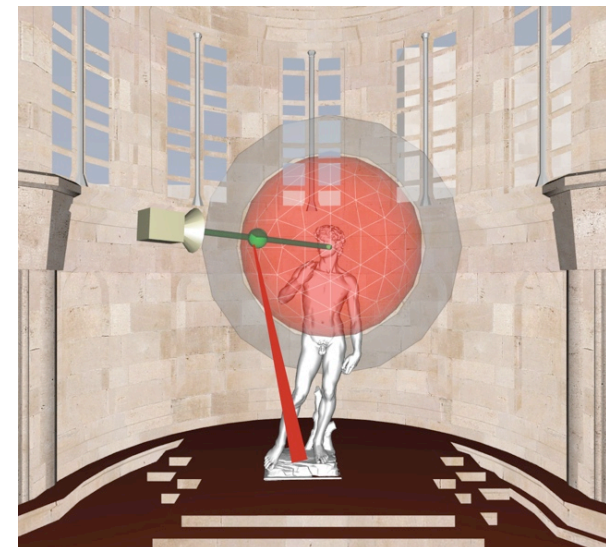
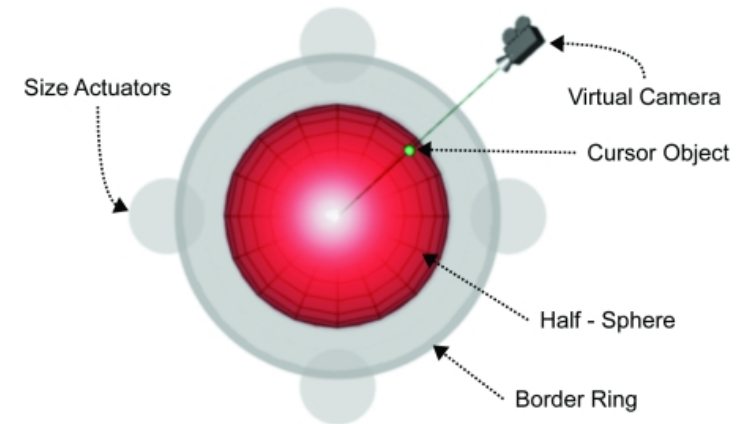
- Examples:
- Implementation:
  - Render scene 2x
  - First, render only into a small viewport (in the shape of the mirror) with mirrored viewpoint
  - Save as texture
  - Second, render into complete viewport from main viewpoint
  - Third, render texture on top of mirror object (no z test)
- Or, use method presented in Computer Graphics class





# The Immersive Navidget – Example for Task Decomposition

- Metaphor for defining the viewpoint directly
- Input device: *wand* with wheels and buttons
- **Decomposition** of the task:
  1. Define the **center of interest (COI)**
    - E.g. by ray casting: shoot ray into scene, intersection point = new COI
    - Will be the center of a sphere
  2. Define radius of sphere = distance of new viewpoint from COI
    - Here: specified using wheel on wand
  3. Define viewpoint on sphere (using ray)
  4. Animate viewpoint on path towards new viewpoint (= smooth teleportation)
    - Switch phases using a button on wand





# Navidget for Immersive Virtual Environments

Sebastian Knödel, Martin Hachet  
[iparla.labri.fr](http://iparla.labri.fr)



- Idea: if we had a model of how users "work", then we could predict how they will interact with a specific UI and what their **user performance** will be
- Advantage (theoretically): no **user studies** and no **UI mock-ups** necessary any more
- Related fields: **psychophysics, user interface design, usability**

# The Power Law of Practice

- Describes, what time is needed to perform an activity after the  $n$ -th repetition:

$$T_n = \frac{T_1}{n^a}$$

$T_1$  = time needed for first performance of the activity,

$T_n$  = time for  $n$ -th repetition,

$a \approx 0.2 \dots 0.6$

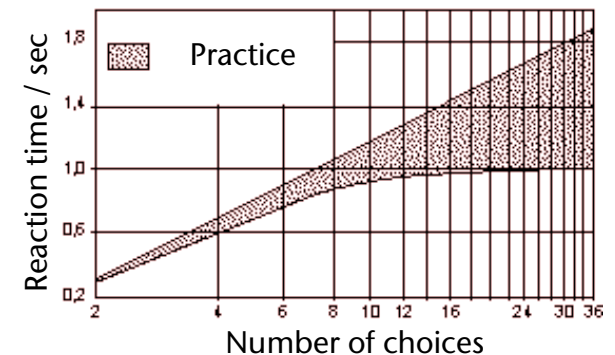
- Warning:
  - Applies only to mechanical activities, e.g. :
    - Using the mouse, typing on the keyboard
  - Does not apply to cognitive activities, e.g., learning for exams! ;-)
- This effect, called **learning effect**, must be kept in mind when designing experiments!

- Describes the time needed to make a **1-out-of- $n$  selection**, but there **cannot** be any **cognitive workload** involved:

$$T = I_c \log_2(n + 1) \quad , \quad I_c \approx 150 \text{ msec}$$

where  $n$  = number of choices

- Example:  $n$  buttons +  $n$  lights, one is lighted up randomly, user has to press corresponding button
  - Assumption: the distribution of the choices is **uniform!**
- Warning: don't apply this law too blindly!
  - E.g., practice has a big influence on reaction time
- Sometimes, Hick's law is taken as proof that one large menu is more time-efficient than several small submenus ("rule of large menus") ... I argue this is — mathematically — correct **only because of the "+1"**, for which there is no clear experimental evidence! Besides, there are many other factors involved in large menus (clarity, Fitts' law, ...)



# Fitts' Law



- Describes the time needed to reach a target
- Task: reach and hit a specific target as quickly and as precisely as possible with your hand/ pencil/ mouse/ etc., from a resting position → "target acquisition"

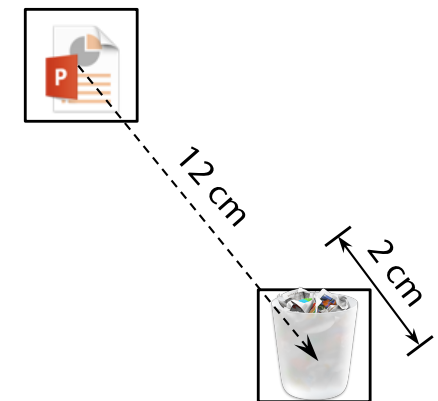
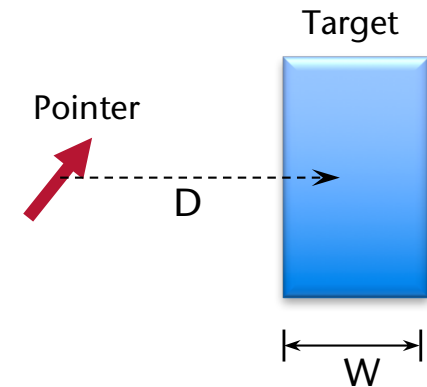
- The law:

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

where  $D$  = distance between resting position and target,  $W$  = diameter of the target

- The "index of difficulty" (ID) =

$$\log_2\left(\frac{D}{W} + 1\right)$$



- Assume a control loop muscle-eye-brain with
  1. Constant processing power of brain
  2. Inaccuracies of movement are proportional to target distance
- Simplification: discretize control loop over time
- Distance of pointer from target

- $D_0 = D$  ,  $D_i = \lambda D_{i-1} = \lambda^i D$

- After  $n$  steps, the pointer is inside the target:

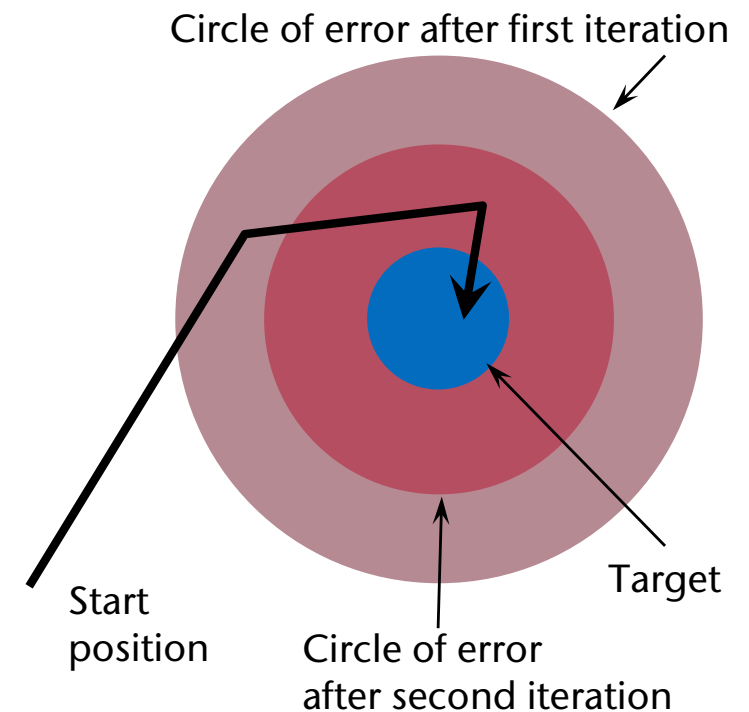
- $D_n = \lambda^n D < W$

- Solving for  $n$  yields  $n = \log_{\lambda}(\frac{W}{D})$

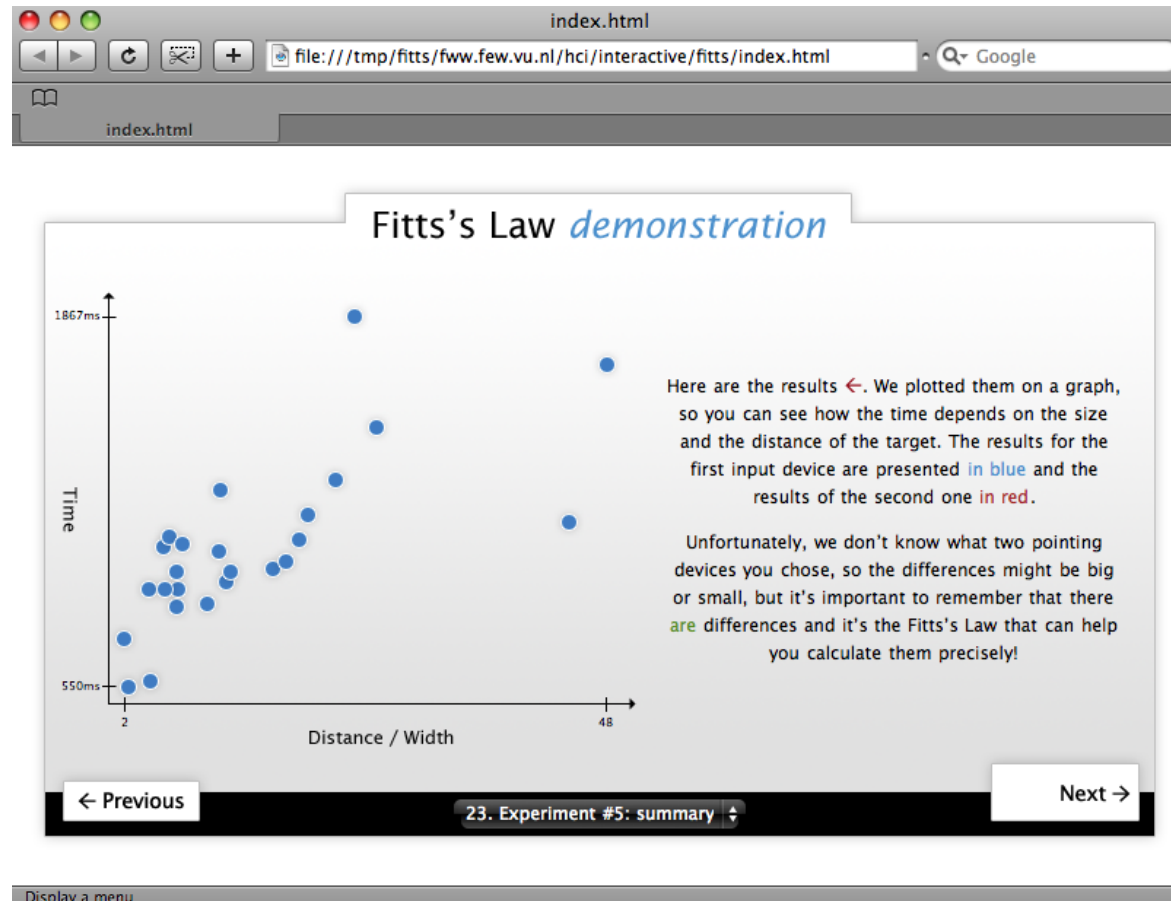
- Each steps takes time  $\tau$ ,

brain's "setup" time =  $a$ , overall:

$$T = a + n\tau = a + \tau \log_{\lambda}(\frac{W}{D}) = a + b \log(\frac{D}{W})$$



- Fitt's Law does apply directly to mouse movements required to hit icons and buttons

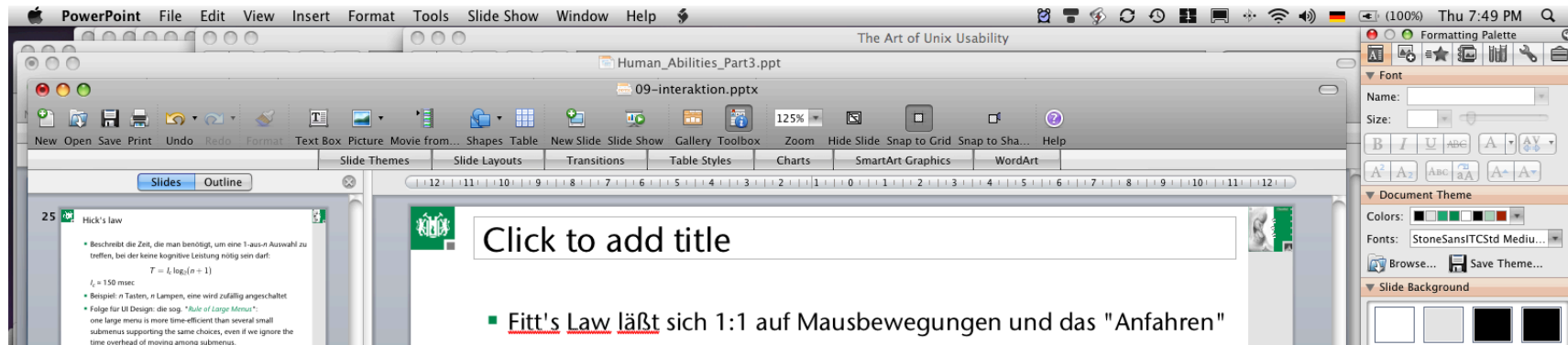


Marcin Wichary , Vrije Universiteit: <http://fww.few.vu.nl/hci/interactive/fitts/>

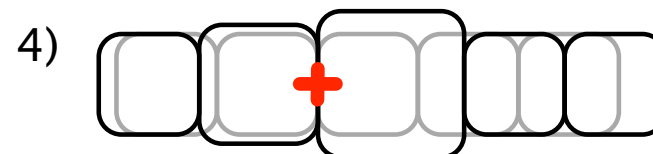
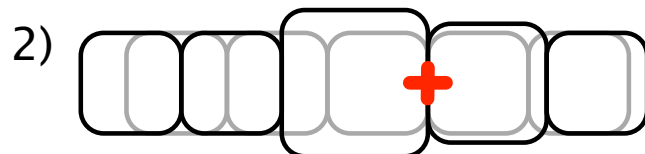
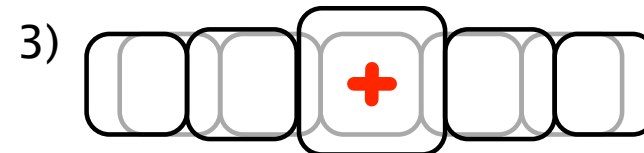
# Applications of Fitts' Law



- "Rule of Target Size": The size of a button should be proportional to its expected frequency of use
- Other consequences:
  - "Macintosh fans like to point out that Fitts's Law implies a very large advantage for Mac-style edge-of-screen menus with no borders, because they effectively extend the depth of the target area off-screen. This prediction is verified by experiment."*
  - [Raymond & Landley: "The Art of Unix Usability", 2004]



- *Tear-off menus* and *context menus*: they decrease the average travel distance  $D$
- Apple's "Dock": the size of the icons gets adjusted dynamically

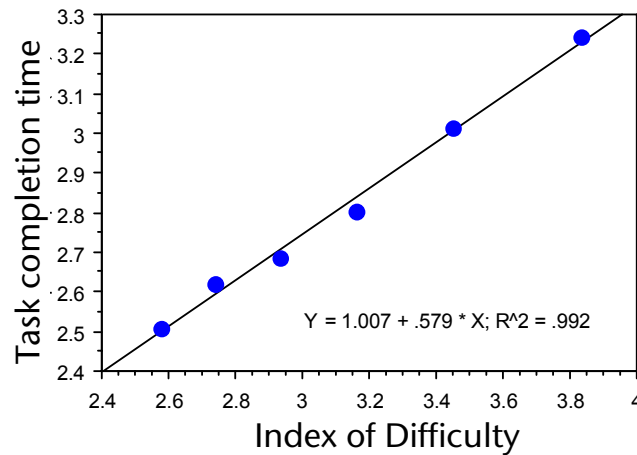




- Obvious limitations of Fitts's Law:
  - Fitts's Law cannot capture all aspects/widgets of a GUI
    - E.g. moving target (like scrollable lists)
  - There are many other decisions with regards to the design of a UI that are contrary to an application of Fitts's law

Fun and instructive quiz on the homepage of this VR course: "A Quiz Designed to Give You Fitts"

- Fitts' law also holds for hand-off tasks:
  - One user grabs a virtual cube using a point probe, moves it towards the other user's point probe, then the other user tries to grab it
  - Result:

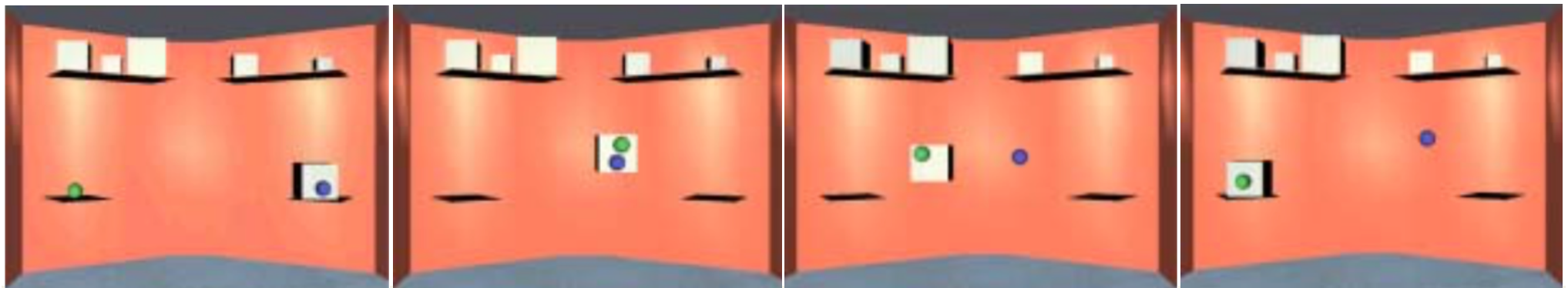


1

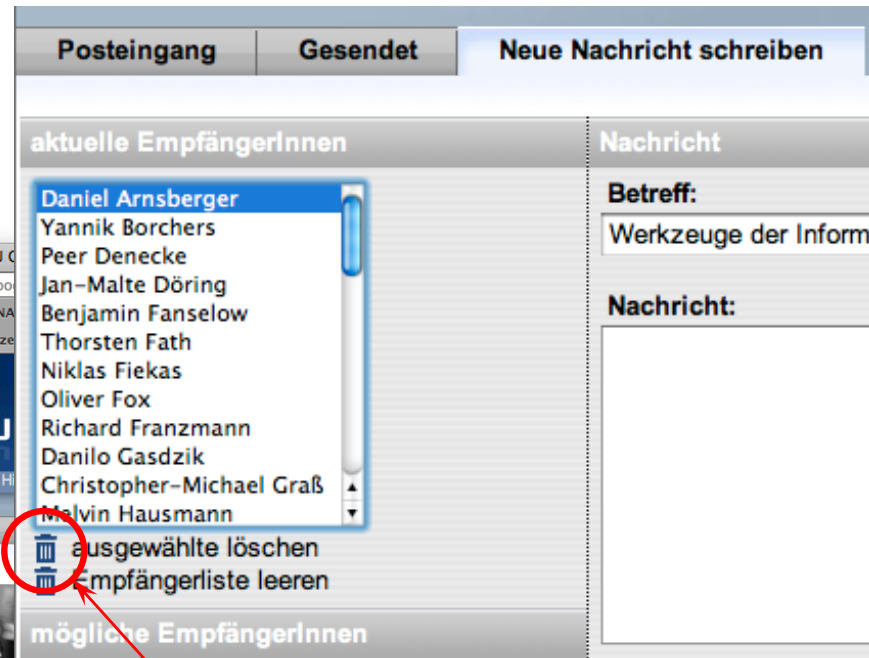
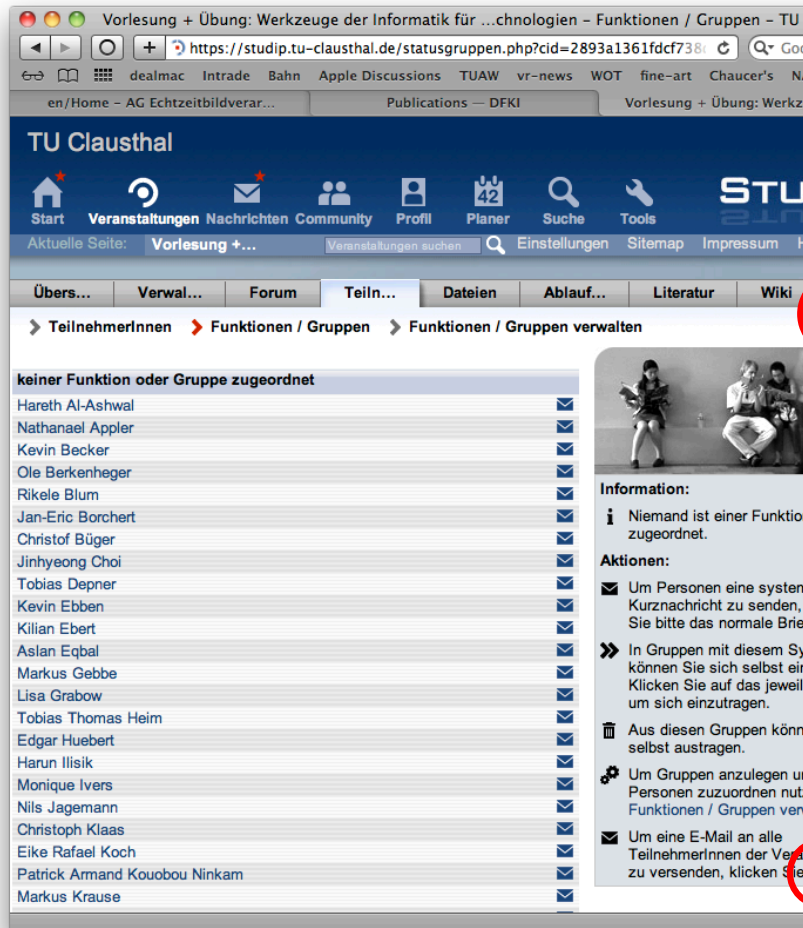
2

3

4



- Screenshots of Studip:



This small symbol is a button!

This little word is a button!  
(and hard to distinguish from the rest of the text/background!)

# Fitts' Law with Lag

- In many interactive systems (VR, telepresence, etc.), more or less lag is present
- Fitts' law with lag:

$$T = a + b(l_h + l_s) \cdot ID$$

where  $l_s$  = system lag ("from motion to photon"),

$l_h$  = "human lag" ("from photon to motion"),

$a$  = time to initialize and terminate task (task dependent),

$b$  = "number of sensory-motor control loop cycles in brain".

Starting values by rule of thumb:

$$b \approx 1.6$$

$$l_h \approx 0.1, \dots, 0.3$$

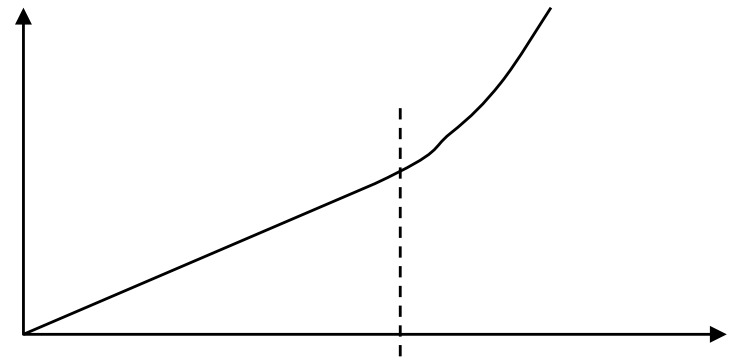
# Isomorphic vs Non-Isomorphic Techniques

- **Display space** = virtual space containing the virtual "pointer"
- **Control space** = physical space containing the tracker/controller/user
- **Control-Display ratio (C-D ratio)** = the ratio of the movement in physical space over the resulting movement in display space
- **Isomorphic** mapping:
  - 1:1 correspondence between physical space and virtual space
  - Better suited for **direct interaction techniques**
- **Non-isomorphic** mapping:
  - "Magic" tools (interaction metaphors) expands working volume or precision
  - Better suited for **remote interaction techniques**
  - Most interaction techniques are non-isomorph

# Direct Selection/Manipulation with Non-Linear Mapping (the "Go-Go Technique")



- **Direct selection/manipulation** requires direct touching & grasping objects with virtual hand
- Goal: increase working volume
- Idea:
  - Scale tracking values non-linearly outside the "near field"
  - Keep linear scaling in near-field for better precision
- Suitable for head and hand tracking
- Works only with absolute input devices
- Disadvantages:
  - Proprioception gets lost
  - No remote precision handling



The Go-Go  
Interaction  
Technique:

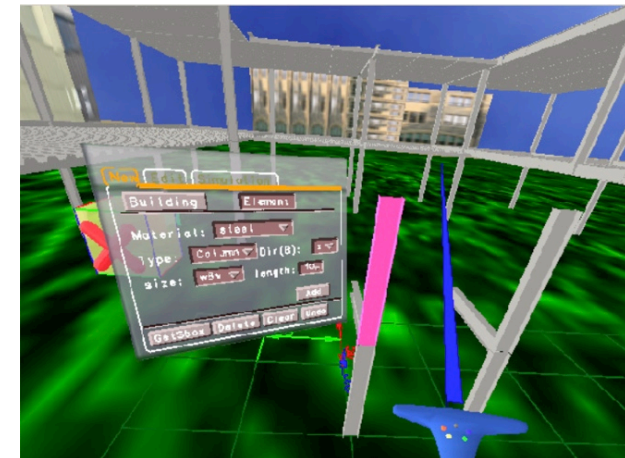
... to reach farther in  
virtual environments

- *Task decomposition:*

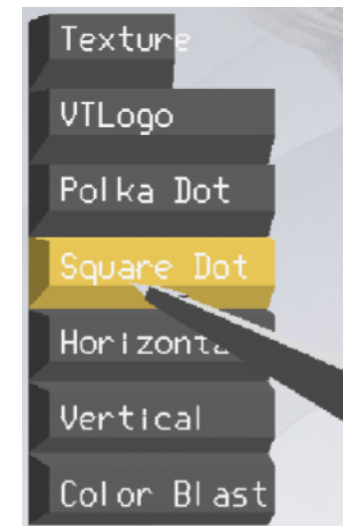
1. Switch selection mode on
2. Identify object(s) to be included in selection
  - Often, this is some form of target acquisition (but not necessarily!)
  - Give some kind of feedback during this step
3. Confirm/cancel
4. Feedback: which objects are actually selected

# Some Possibilities for Step 2 (Identifying Objects)

- **Direct** = touch with hand
- **Ray-based (ray casting)**
  - E.g., shoot "laser pointer" from virtual hand into scene
  - Or: extend conceptual ray from current viewpoint through finger tip (a.k.a. **occlusion technique**)
- **Volume-based**, e.g. using a cone around the ray
- Speech (name objects)
- Menues (how to select items in menu?)
- Mixed techniques:
  - Image plane interaction (later)
  - World-in-Miniature (dito)
  - Etc.



Laser pointer



Menu with ray technique



- Variables used in the following:

H = hand position

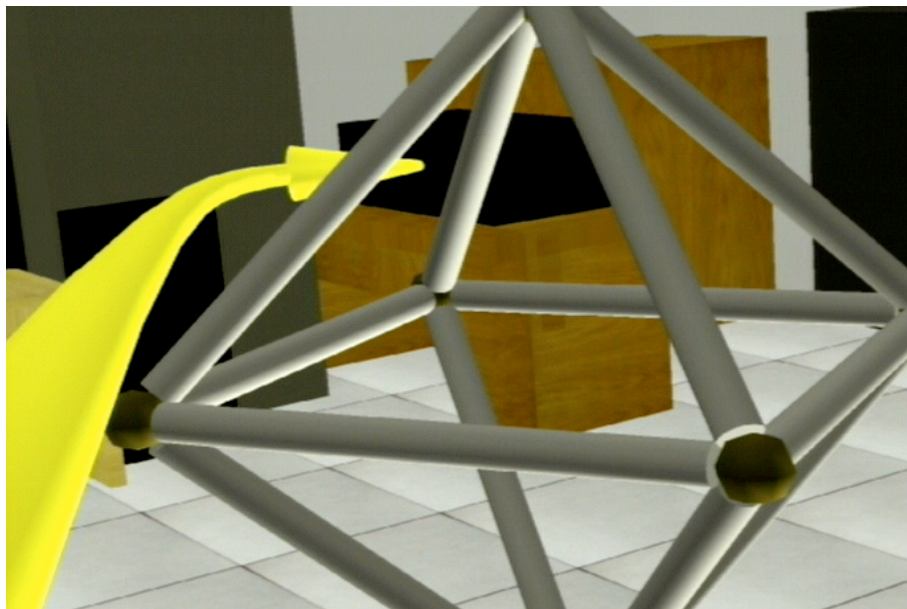
E = viewpoint

h = "pointing direction" of the hand

H<sub>2</sub> = position of the left hand

Technique	Volume	Origin	Direction
Raycasting	ray	H	h
Flashlight	cone	H	h
Two-handed pointing	ray	H <sub>2</sub>	H - H <sub>2</sub>
Occlusion selection	ray	E	H - E
Aperture	cone	E	H - E

- Observation: people try to describe a "curve" with their pointing gestures, when object pointed at is not in line of sight
- Metaphor in VR: bent selection ray
- Problem: intuitiv and easy specification of a curvature using the available input devices (dataglove, trackers, ...)



## The Flexible Pointer

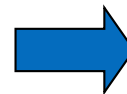
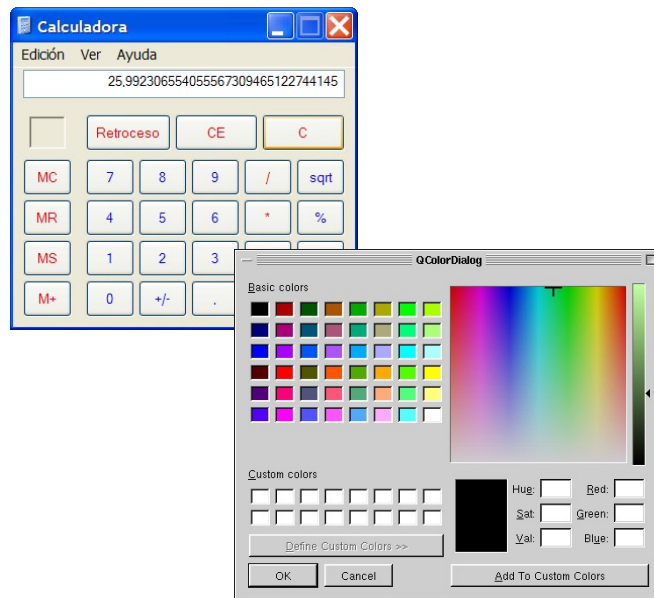
An Interaction Technique for Selection  
in Augmented and Virtual Reality

Alex Olwal & Steven Feiner

Computer Graphics & User Interface Lab  
Columbia University, New York

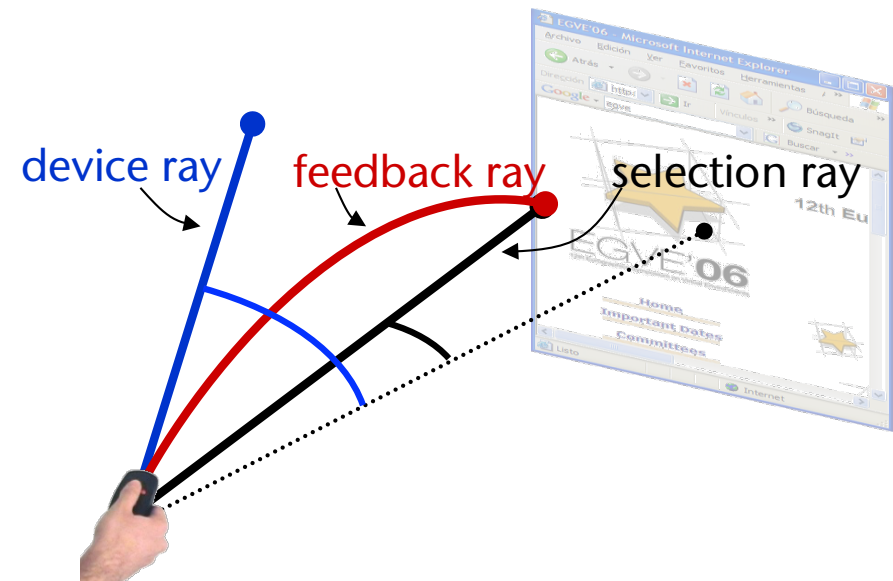
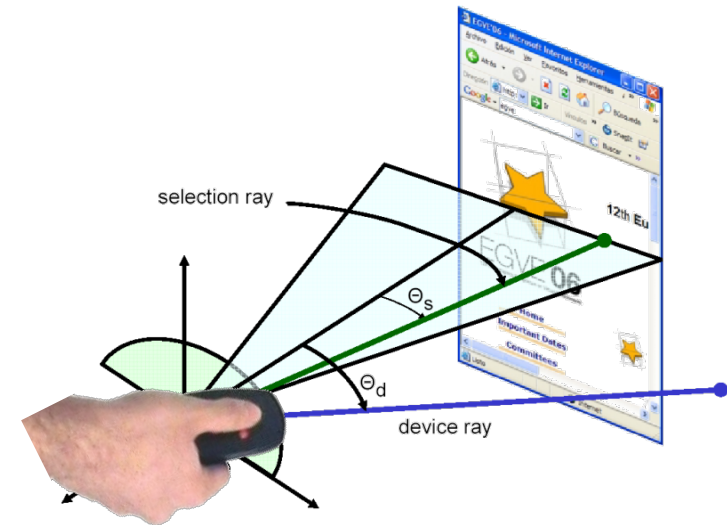
Conference supplement of UIST 2003

- Task here: control so-called **hybrid interfaces**
  - Goal: control 2D GUIs of desktop apps directly in VR
  - Implementation: a modified VNC client

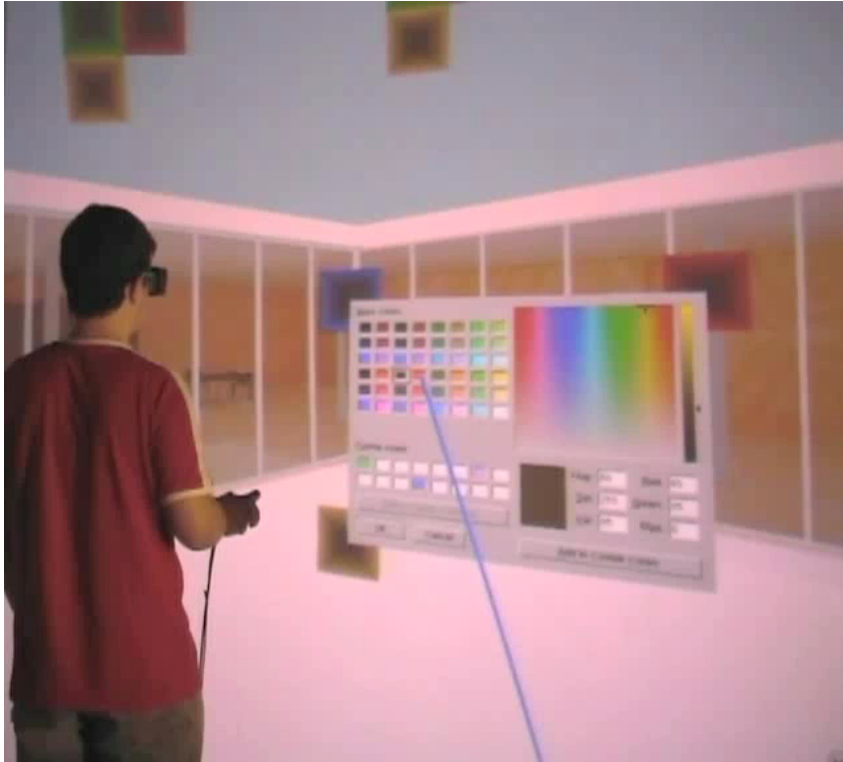


- Problem: the target width (here: solid angle!) is extremely small

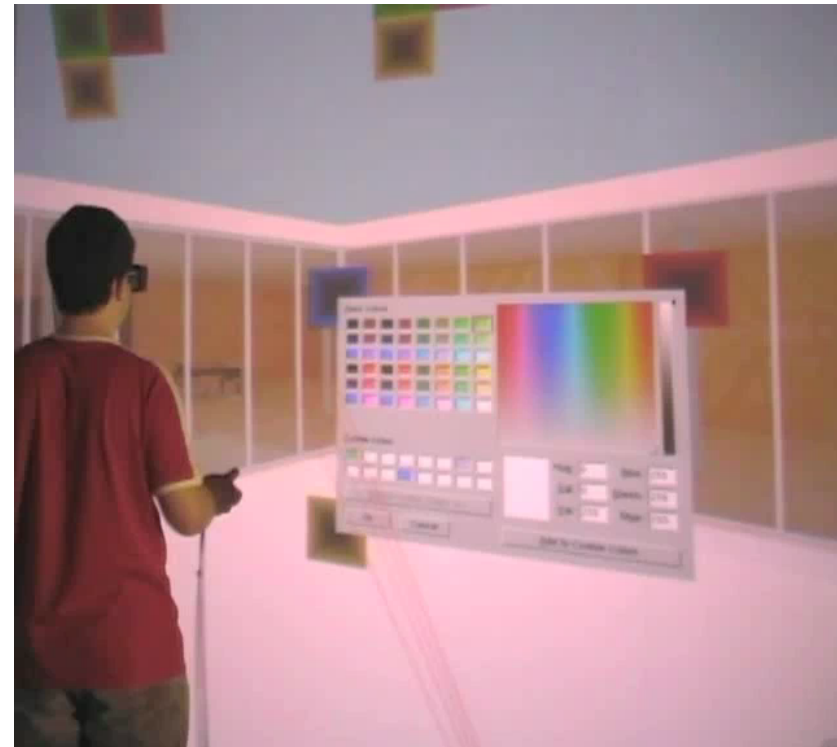
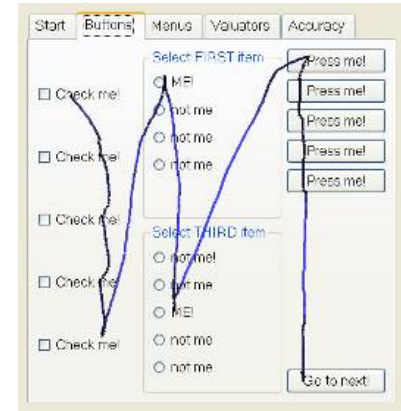
- Idea:
  - Scale the C-D ratio down, as soon as user interacts with a 2D window (in VE)
  - Problem: how to make the non-isomorphism intuitive, how to bridge the noticeable difference between motor & display space?
- Two rays were irritating to users
- Solution: show just **one** ray, but make it **bend**



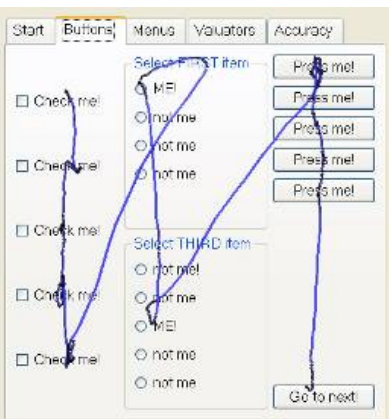
■ Result: much increased user efficiency:

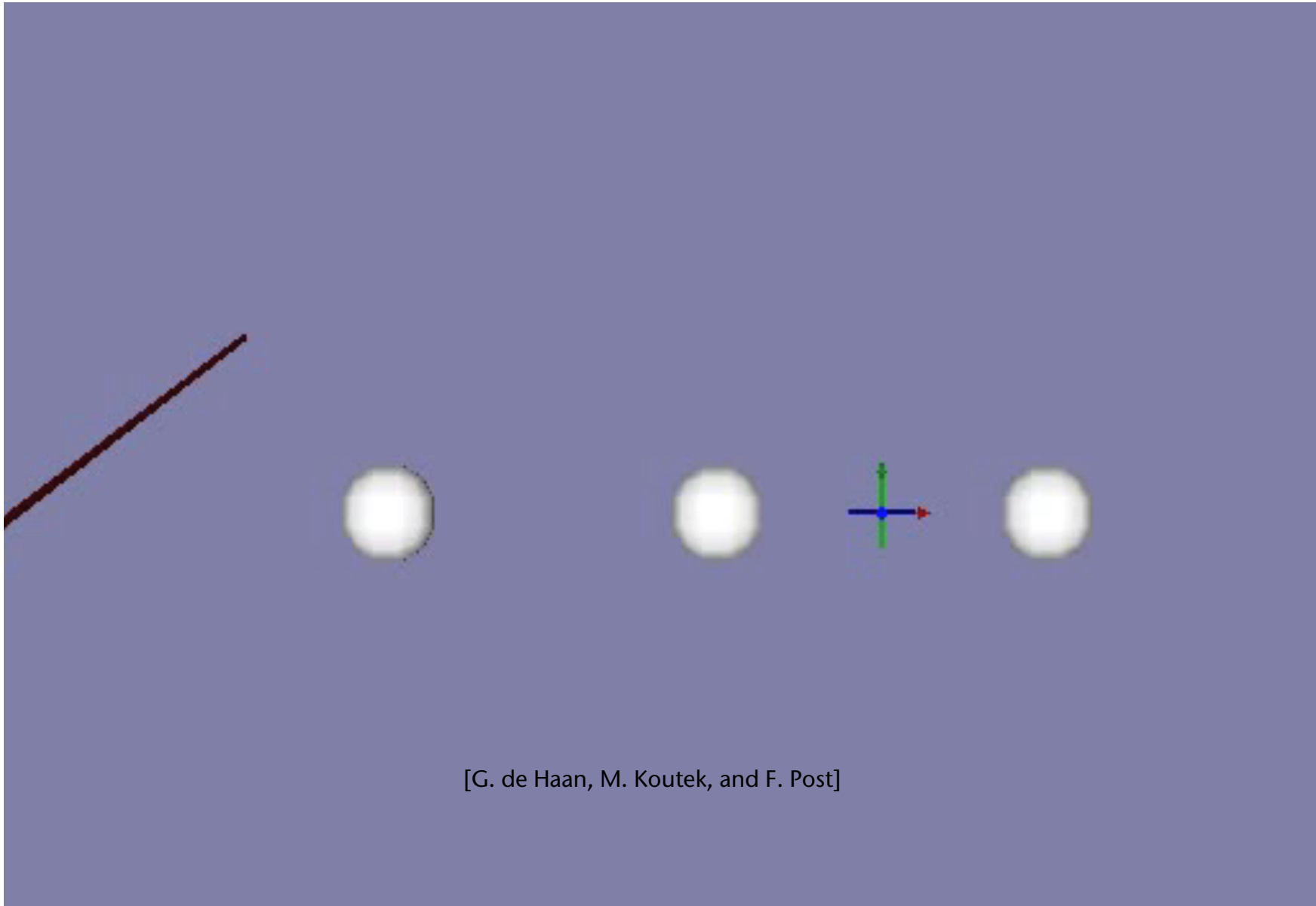


C-D ratio < 1

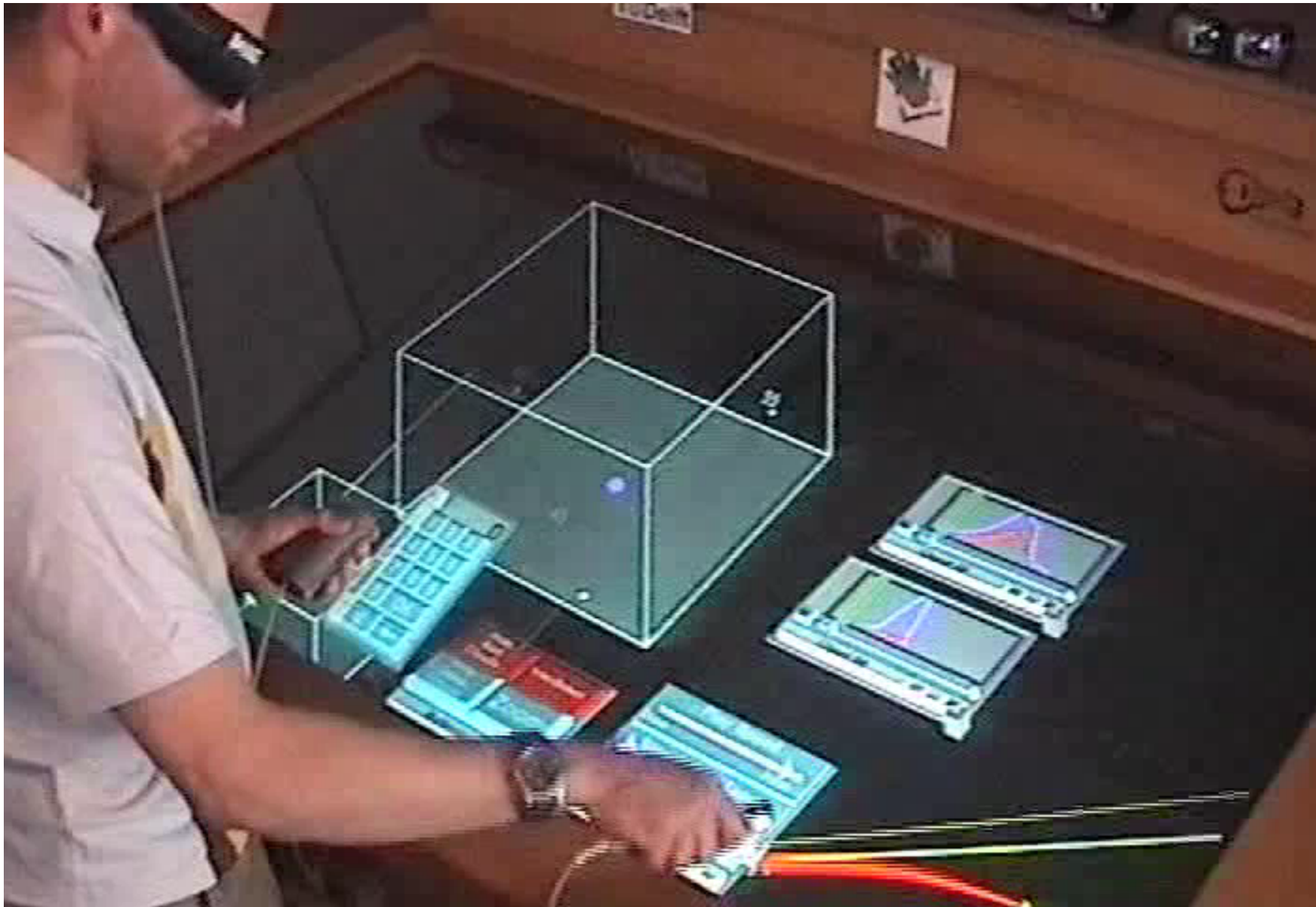


C-D ratio = 1









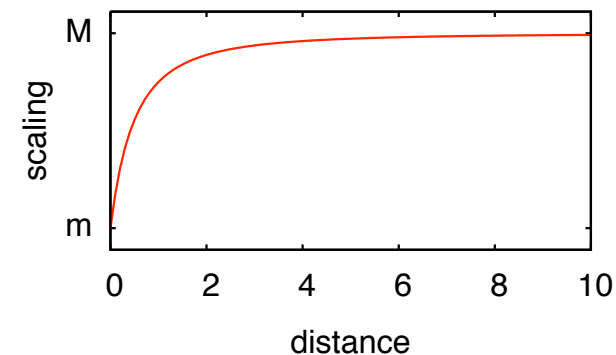
# "Semantic" Pointing (Non-Constant C-D Ratio) Optional



## ■ Idea:

- Modify C-D ratio depending on distance between pointer and closest target
- Large distance  $d \rightarrow$  scale motions from motor space up
- Small distance  $d \rightarrow$  scale motion downs = high precision in display space
- E.g. with a function like this one:

$$s(d) = M + \frac{m - M}{(1 + d)^\alpha}$$



## ■ Visual feedback:

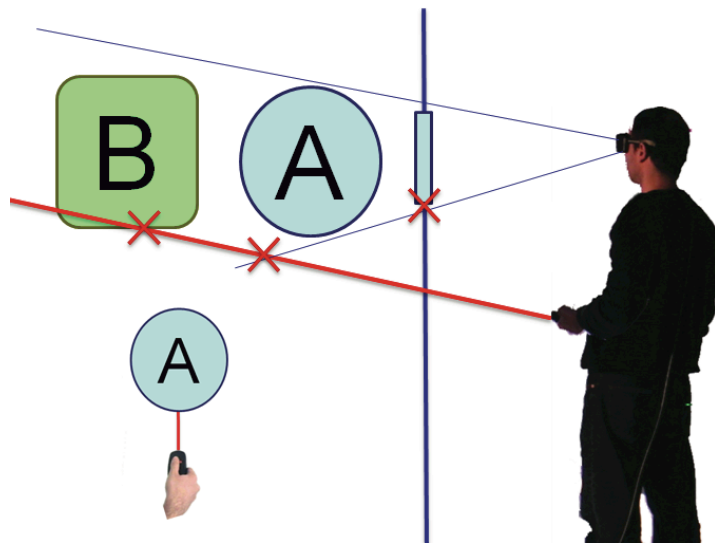
- Cursor size ~ C-D ratio
- Color of pointer visualizes distance of target (e.g. "red" = "very close")



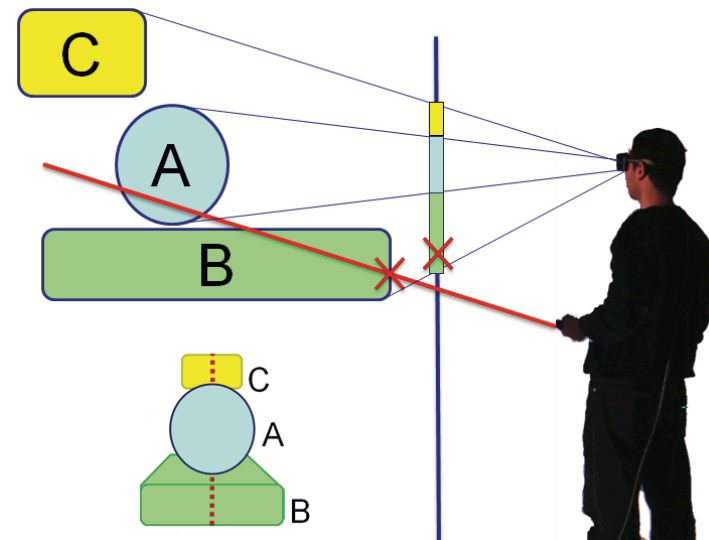


# The Eye-Hand Visibility Mismatch

- Two obvious problems of all techniques where ray emanates from the user's (virtual) hand :
  1. The set of objects visible from viewpoint *E* is different from set of objects "visible" from hand position *H*



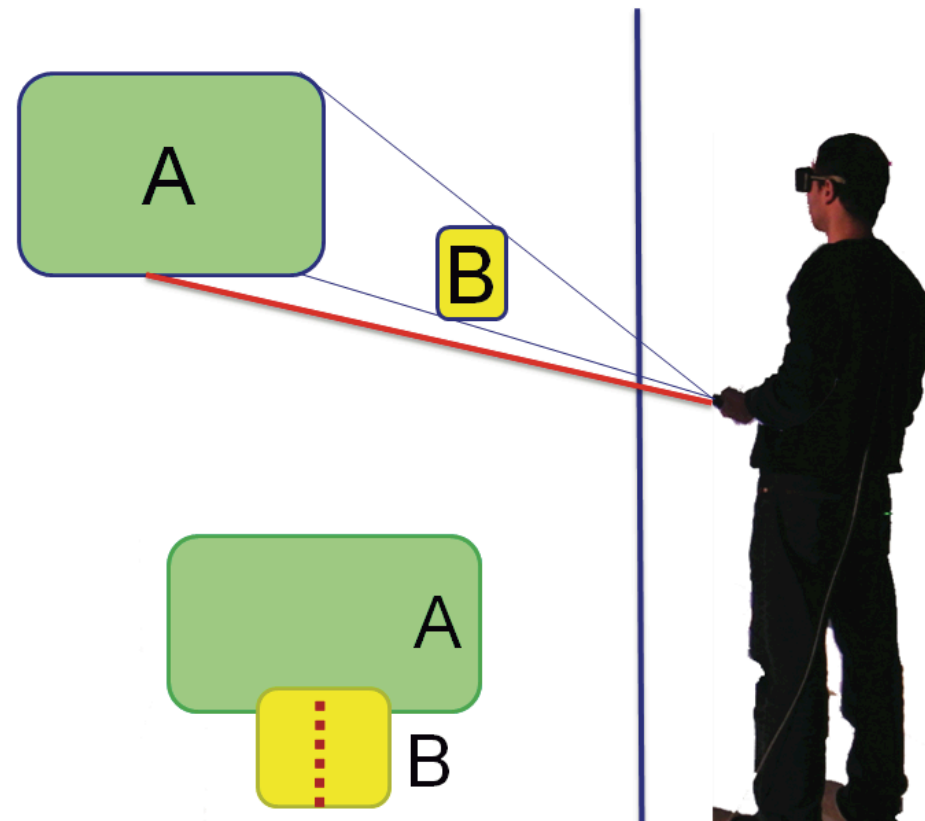
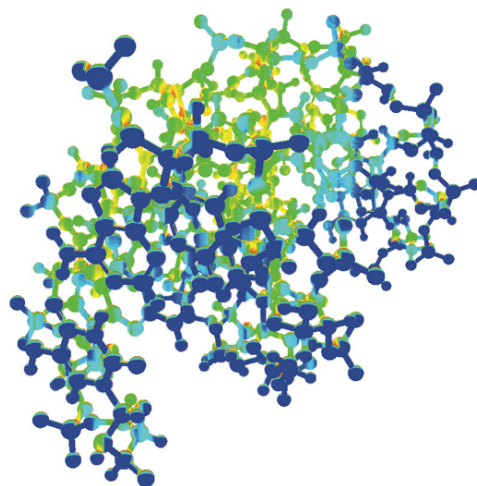
Object B is selectable,  
but not visible



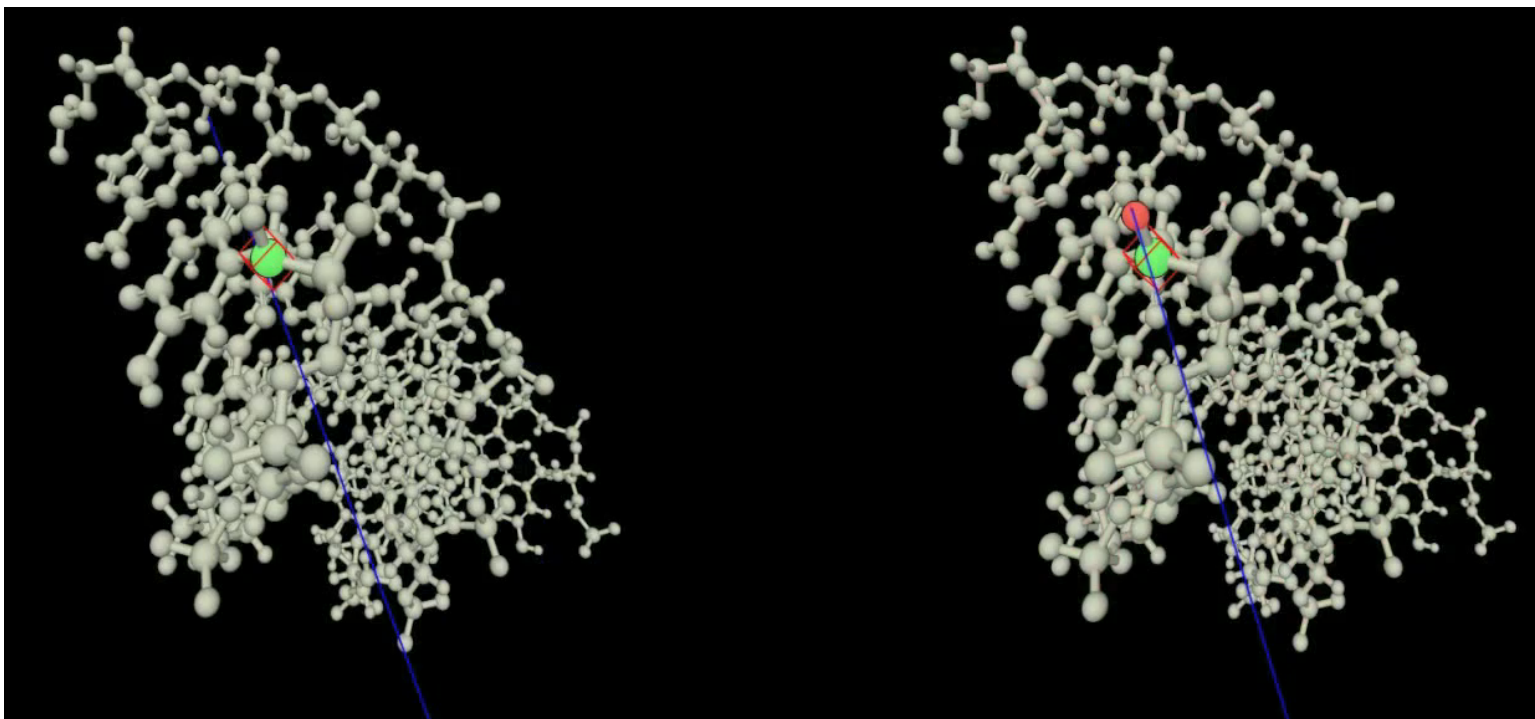
Object C is visible,  
but not selectable

2. The surface of an object "visible" from  $H$  is different than surface visible from  $E$ ; consequences:

- Real target width is different than visible target width
- Perhaps no/insufficient feedback during selection process



- Idea:
  - Shoot selection ray emanating from viewpoint  $E$  into hand direction  $h$
  - Visual feedback: ray emanating from  $H$  to first intersection of selection ray
- Experiment shows: users are ca. 15-20% faster than with normal ray



Argelaguet, Andujar, Trueba

# IntenSelect: Ranking + Filtering

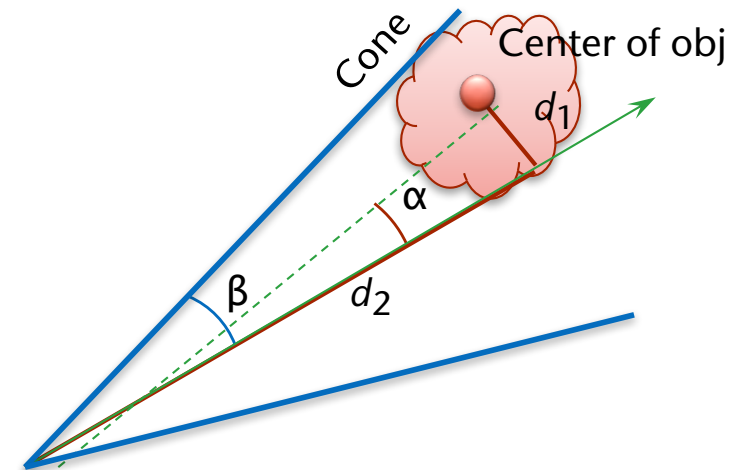
- Assumptions:
  - Cone is better than ray
  - In general, a lot of objects are in the cone (i.e., dense environment)

- Idea for **disambiguation**:

- Define scalar field inside cone
- Compute a "score" for each object
- Create ranking of objects

- Simple scoring function:

$$s = 1 - \frac{\alpha}{\beta}$$

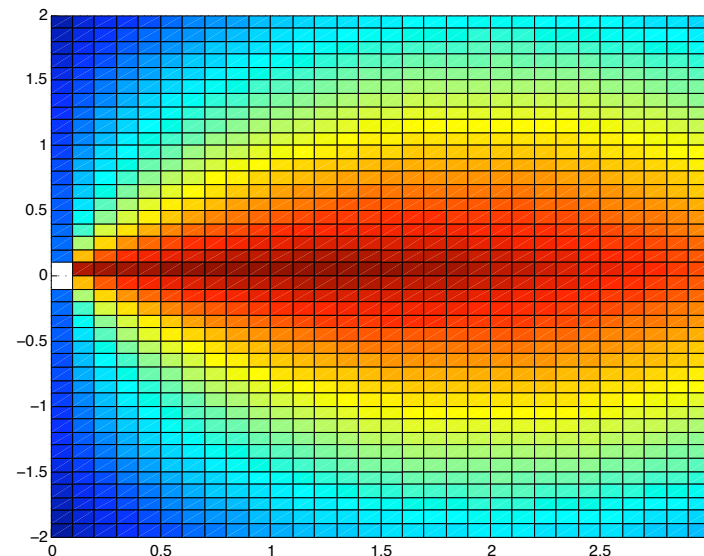
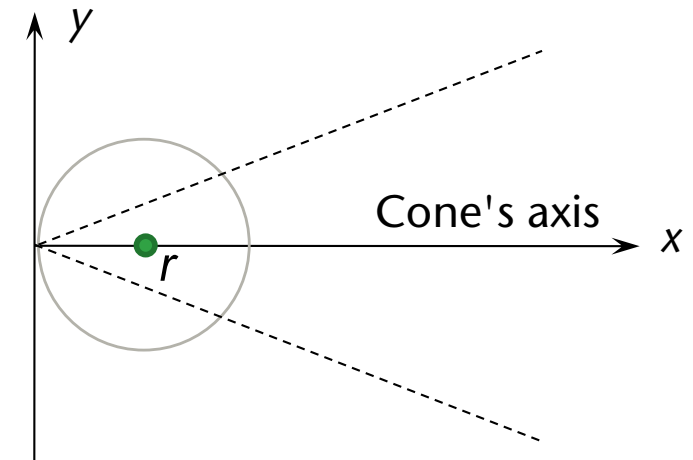


- A scoring function that prefers near objects:

$$s = 1 - \frac{1}{\beta} \tan^{-1} \left( \frac{d_1}{(d_2)^k} \right), \quad k \in \left[ \frac{1}{2}, 1 \right]$$

- Even more preference of near objects:

$$s = \left( 1 - \frac{1}{\beta} \tan^{-1} \left( \frac{d_1}{(d_2)^k} \right) \right) + 0.1 \left( 1 - \frac{(x - r)^2 + y^2}{r^2} \right)$$



- Problem: jitter in user's hand leads to frequent changes in ranking
- Solution: filtering

$$s = s(t)$$

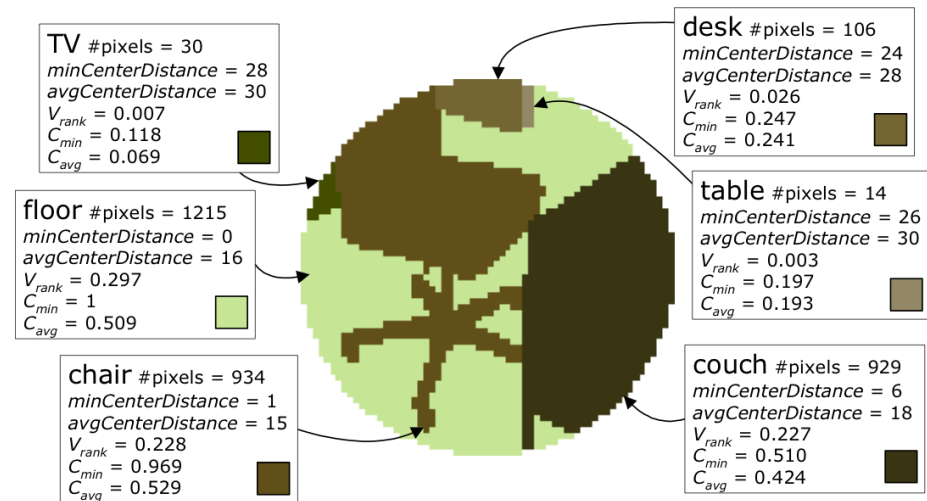
$$\hat{s}(t) = \sigma \hat{s}(t - 1) + \tau s(t)$$

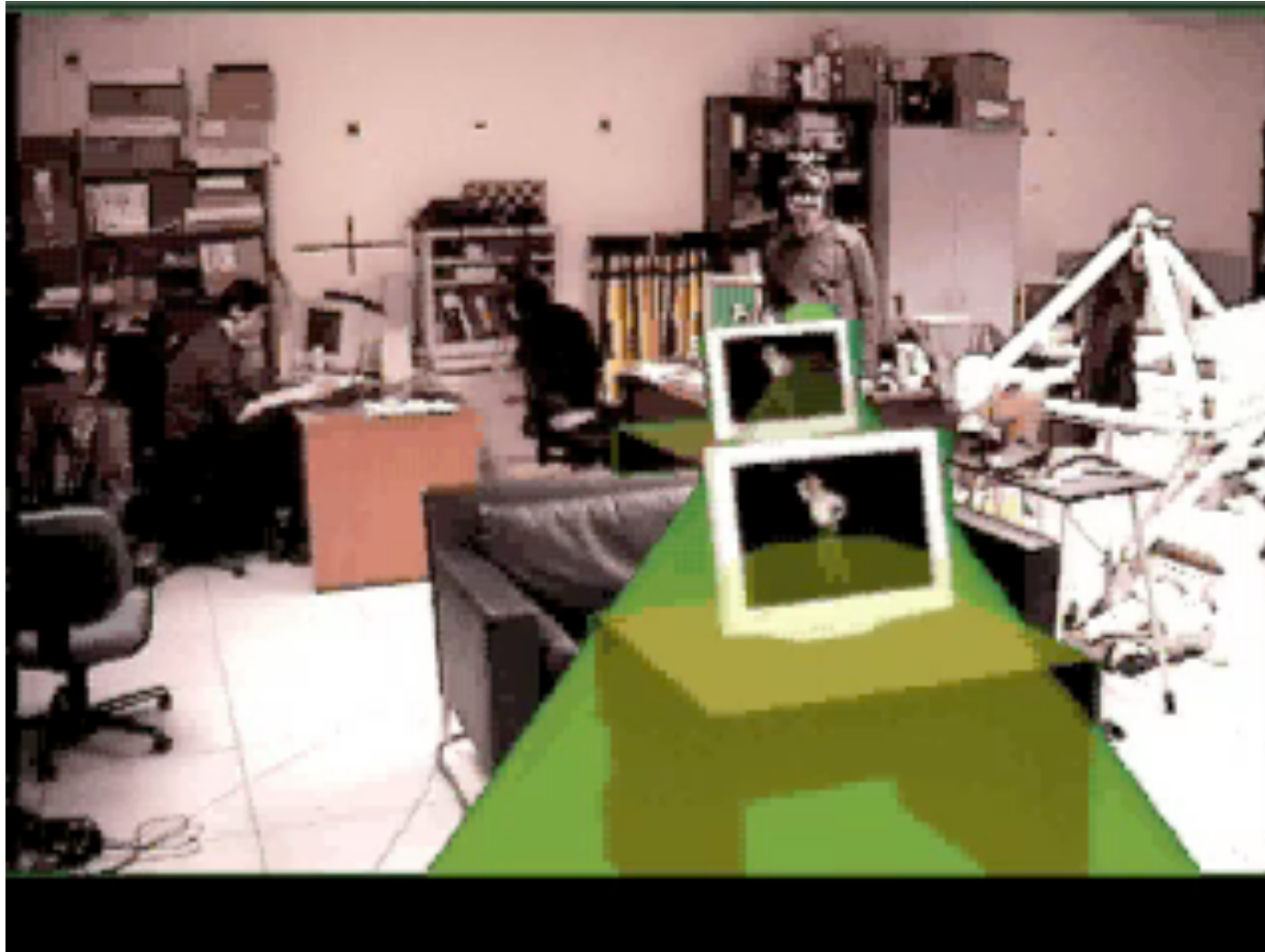
$s(t)$  = score over time,  $\sigma$  = "stickiness",  $\tau$  = "snappiness"

- Generalization: FIR filter (see Chapter 7)
- Feedback to user:
  - Bend ray towards object with highest ranking
  - Show straight ray for cone's axis

- Other distance functions, e.g., prefer far-away objects
- Better computation of "distance" from cone's axis:
  - Render object with low resolution into off-screen frame buffer with "viewpoint" = apex of cone, viewing direction = cone's axis
  - Compute average distance of all pixels of object from center (= cone's axis) :

$$s' = 1 - \frac{\frac{1}{n} \sum_{\text{pixel } p} d(p)}{\text{radius}}$$

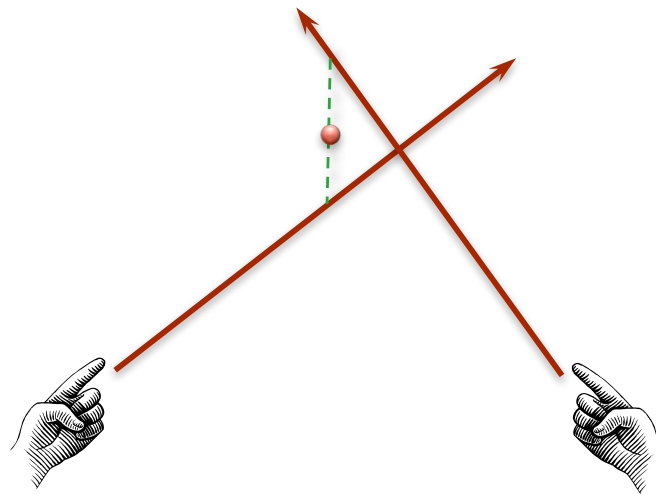


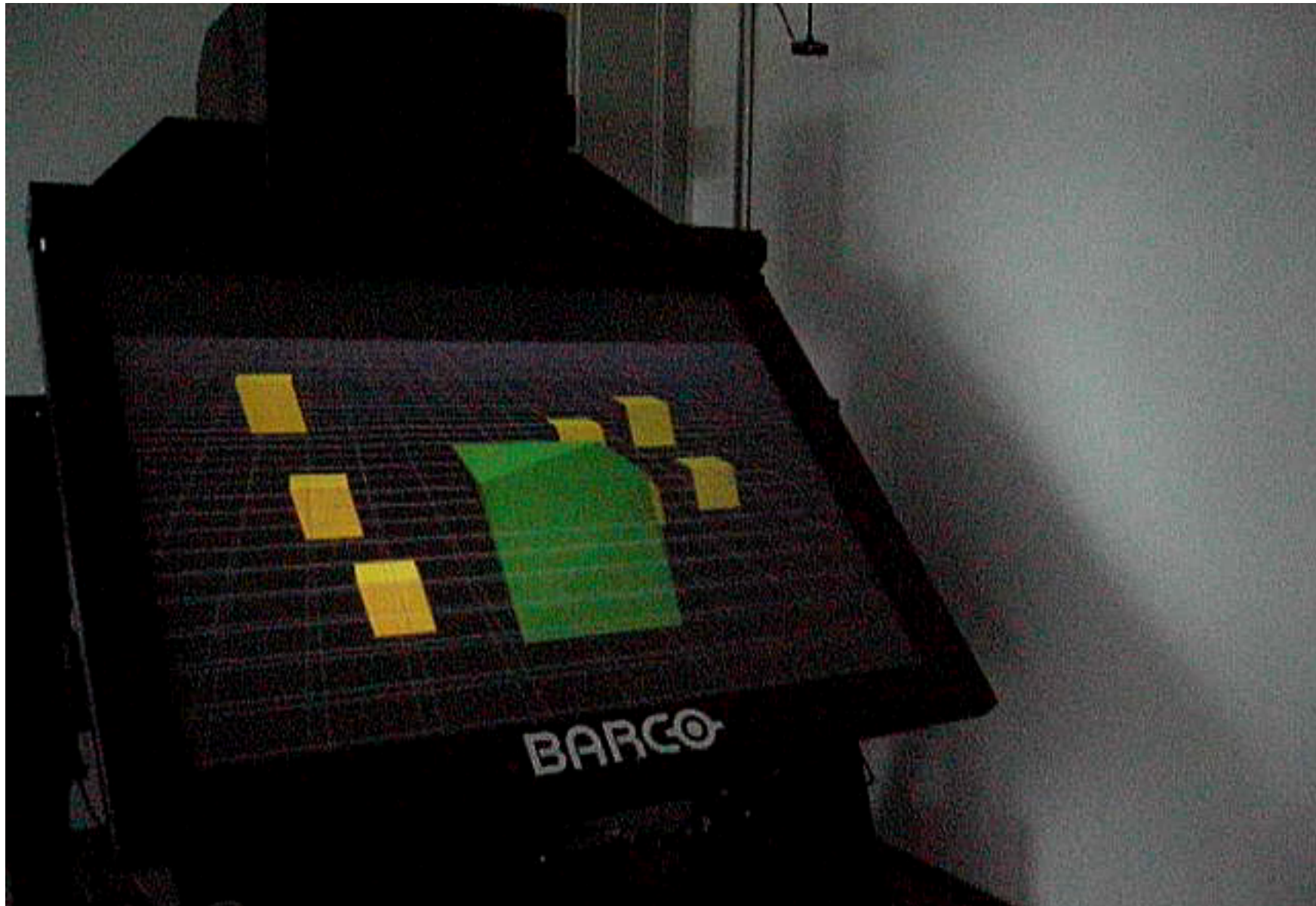


Here, disambiguation is done via voice commands



- Idea: intersection of two rays defines "selection center"
- Practical implementation:
  - One ray per hand
  - Trigger selection mode as soon as distance between rays  $<$  threshold
  - Midpoint of the shortest line between the rays = "selection center"
  - Select all objects "close enough" to this selection center

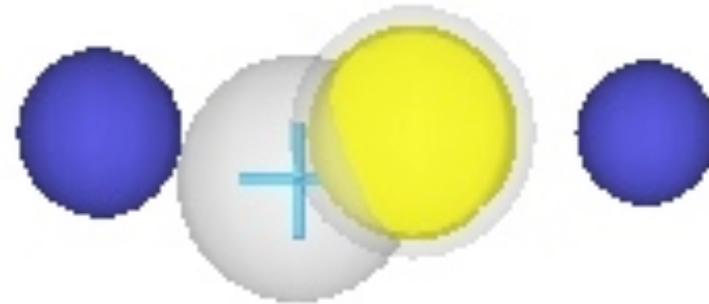




- Another method to increase the **effective target size**

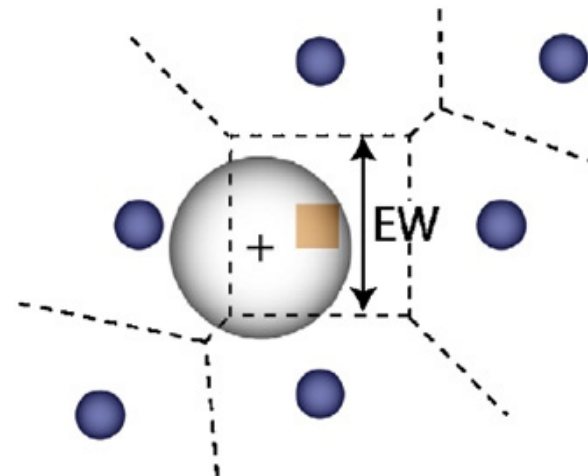
- **Bubble Cursor:**

- 3D cross hairs
- Select always the closest object
- Make radius of transparent sphere around 3D crosshairs = distance to closest object (feedback for "density")
- Feedback to indicate active object: transparent sphere around it

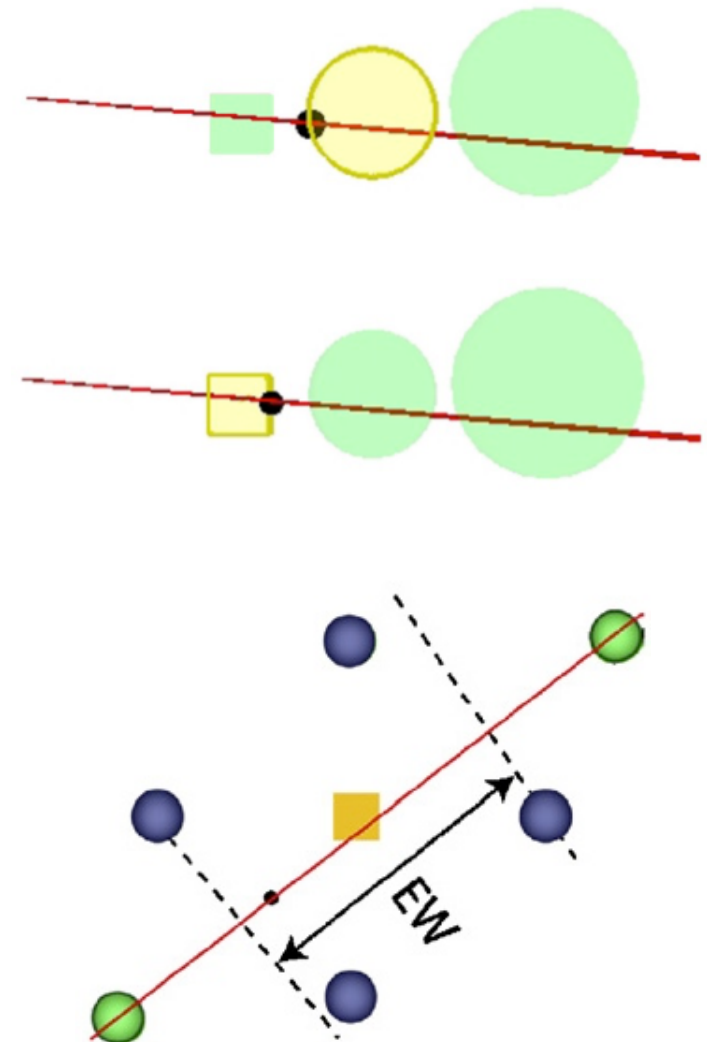


- Effective target size = Voronoi region of object →

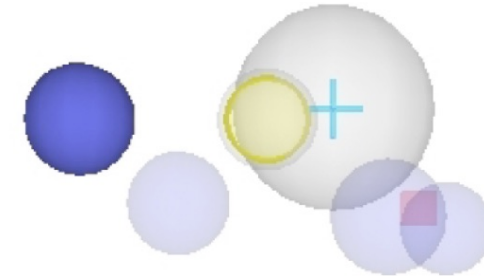
- (For Voronoi regions, see course "Geometric Data Structures for CG" )



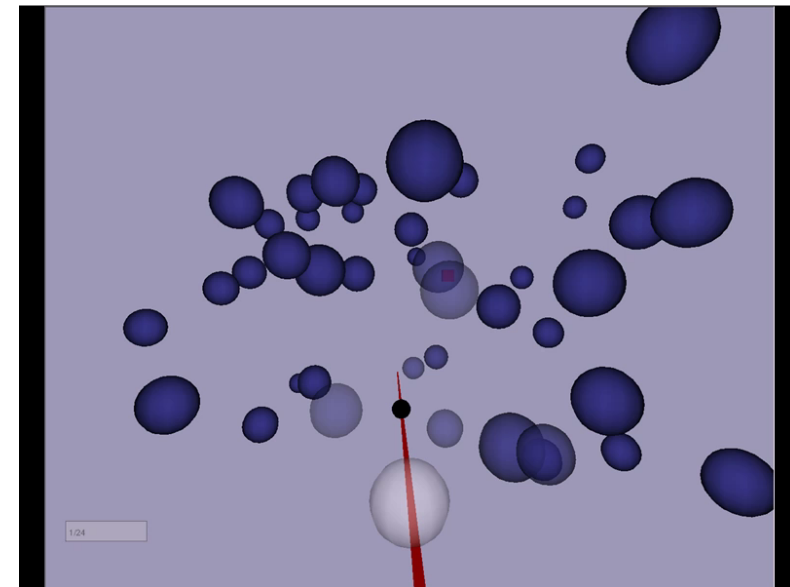
- **Depth Ray:**
  - Only consider objects being "stabbed" by the ray
  - User moves a "depth marker" along the ray
  - Of all "stabbed" objects, take the one closest to the depth marker
- Effective target size = segment on the ray



- Handling occlusion: make occluders in front of the depth marker transparent (possibly depending on the distance from the depth marker)



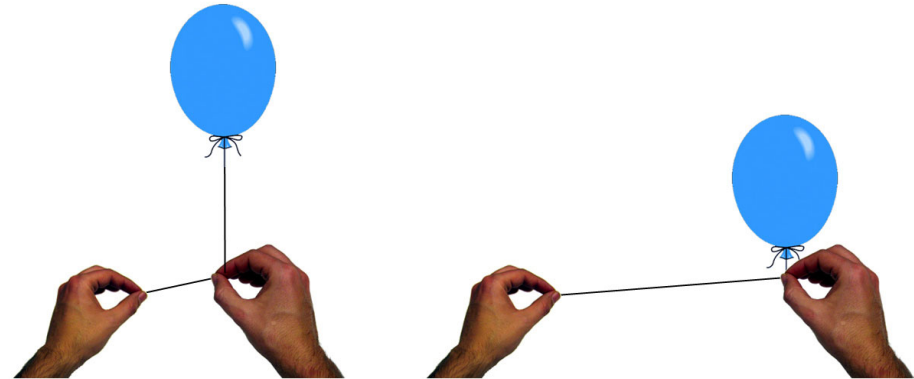
Bubble cursor [Lode van Acken]



Depth Ray

# Balloon Selection

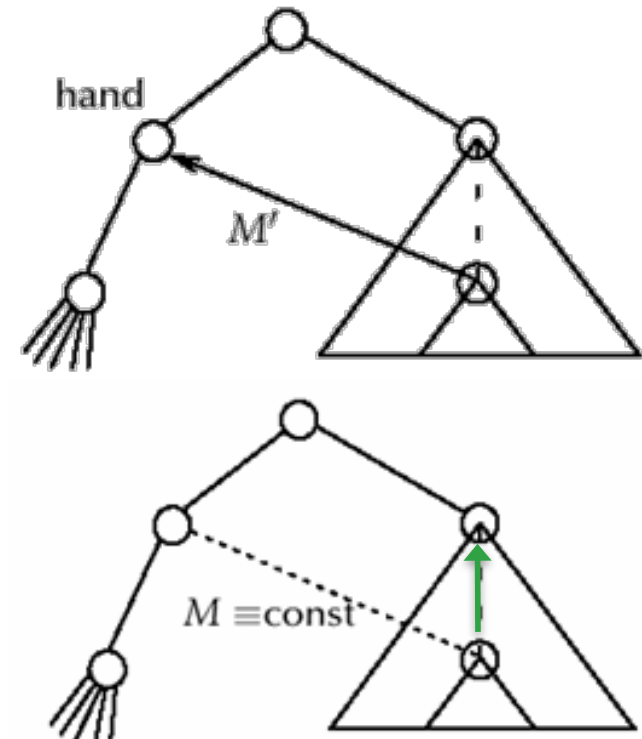
- Idea: control a helium balloon
  - Dominant hand controls 2D position
  - Non-dominant hand controls 1D height
  - Meant for usage on work bench
- Implementation:
  - Right/left index finger defines position / height, resp.
  - Both index fingers remain on the table
  - *System control* (e.g., triggers) by contacts in data glove
- Advantage:
  - **Decomposition** of a 3D tasks in two separate **low-dimensional** tasks
  - Natural constraint (table, a.k.a. *passive haptics*)





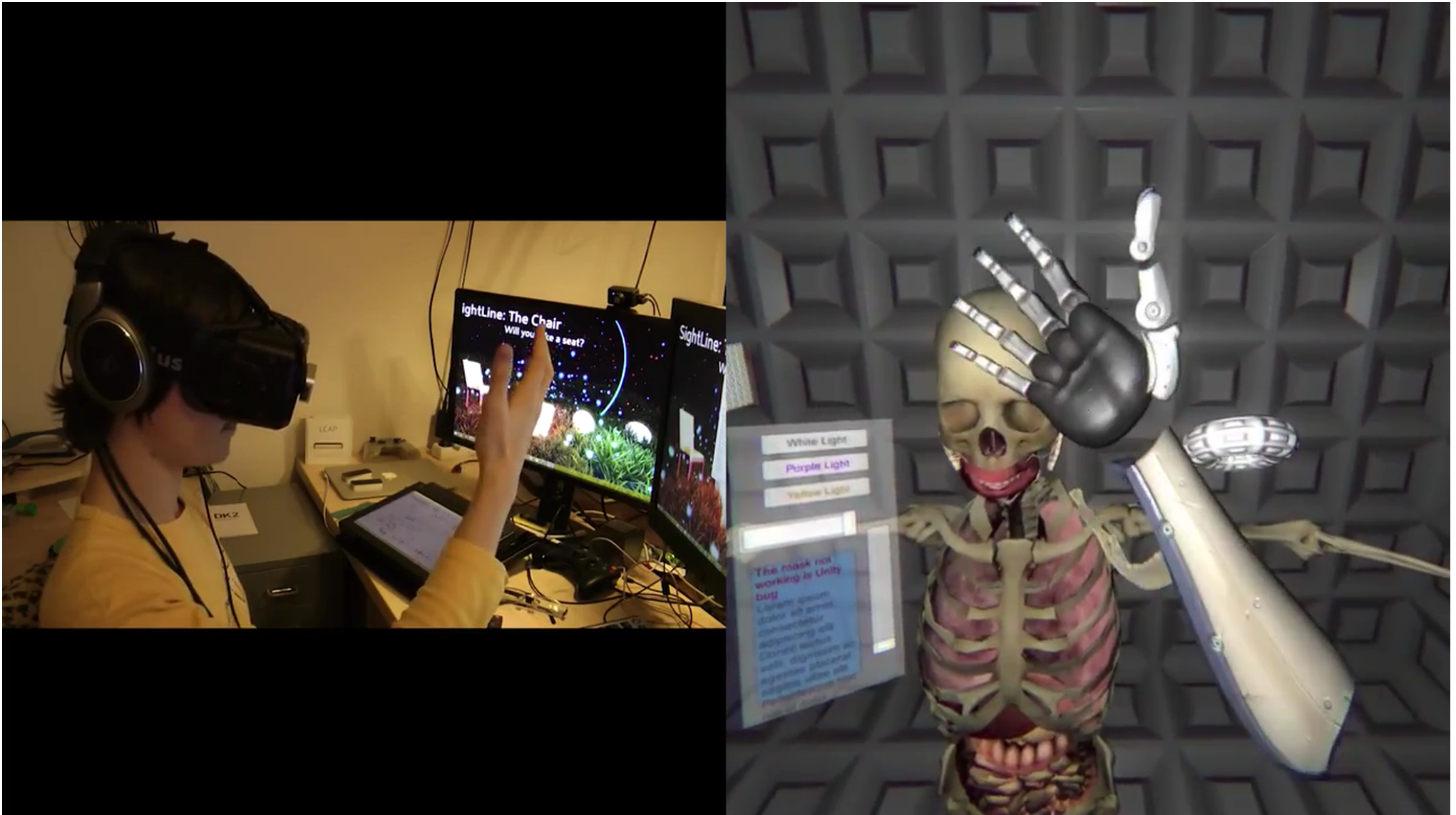


- Another pretty frequent interaction task
- Simple, **direct** grasping using "sticky metaphor" (not realistic):
  1. Select object
  2. Trigger grasping (via gesture, speech command, ...)
  3. Wait for collision between hand and object
  4. Make object "stick" to hand
  5. Trigger release
- How to implement the "sticking"?
  - Either, re-link object to hand node,
  - Or, maintain transformation invariant between hand and object
  - My experience: with non-trivial applications, re-linking causes trouble!





# Example Video for "Stick-to-Hand" Grasping



World of Comenius

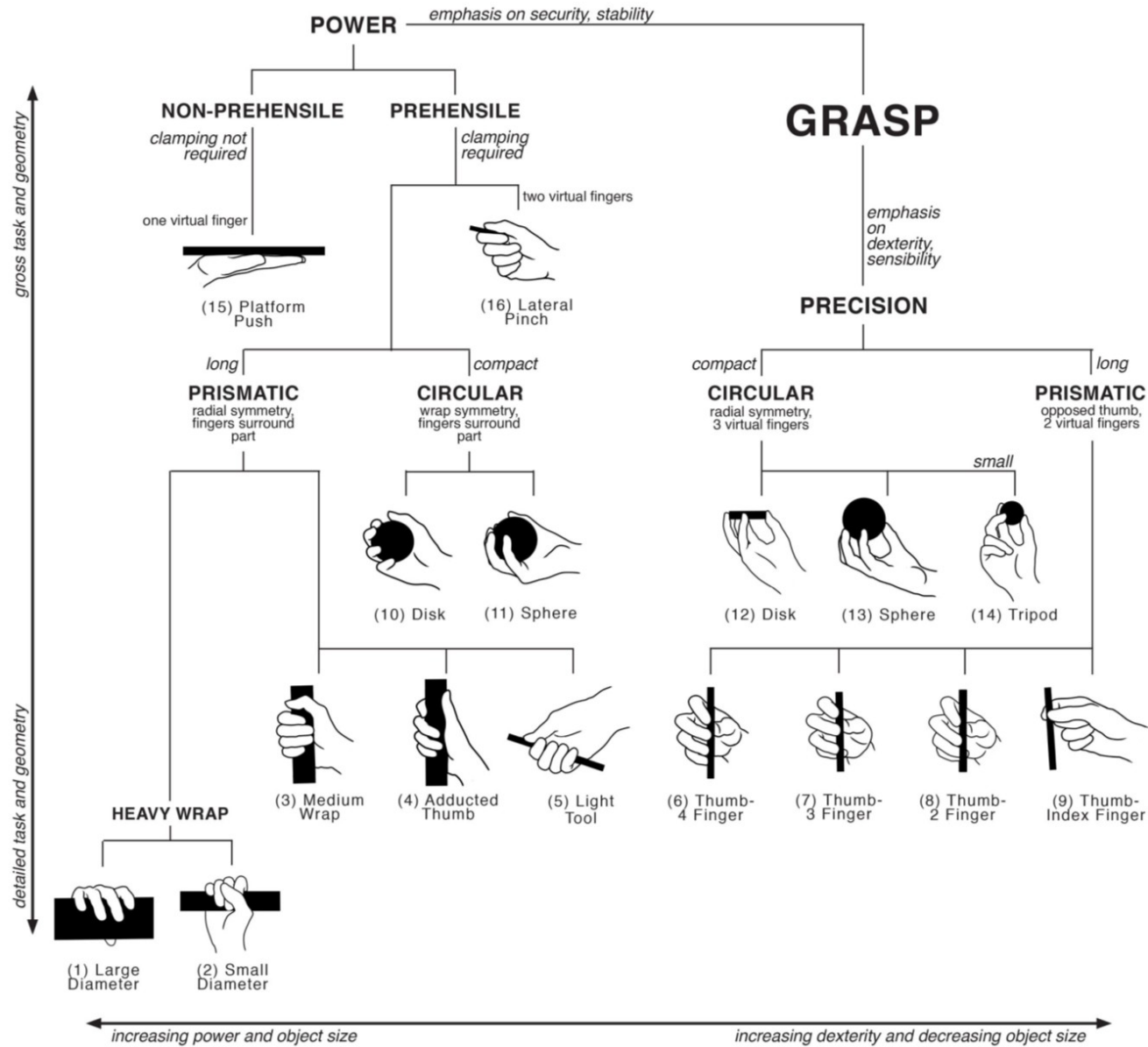
# A Vision: Natural User Interaction (NUI)

- One of the goals: interact with virtual objects using our real hands as if they were real objects, i.e., **no interaction metaphor**



Courtesy of Volkswagen AG







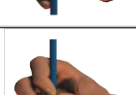
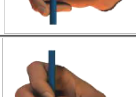


- Originally proposed by Ben Shneiderman in 1980 as **Direct Manipulation**
  - For GUIs, but the same principles and benefits apply to today's NUIs
- Direct manipulation benefits:
  - **Novices** can learn basic functionality quickly, usually through a demonstration by a more experienced user
  - **Experts** can work extremely rapidly to carry out a wide range of tasks, even defining new functions and features
  - **Error messages** are rarely needed
  - **Feedback**: users can see immediately, if their actions are furthering their goals, and if not, they can simply change the direction of their activity

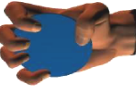





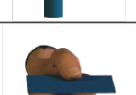
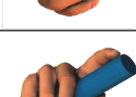












Cutkosky & Howe, 1990; and Zheng et al, 2011

# Optional

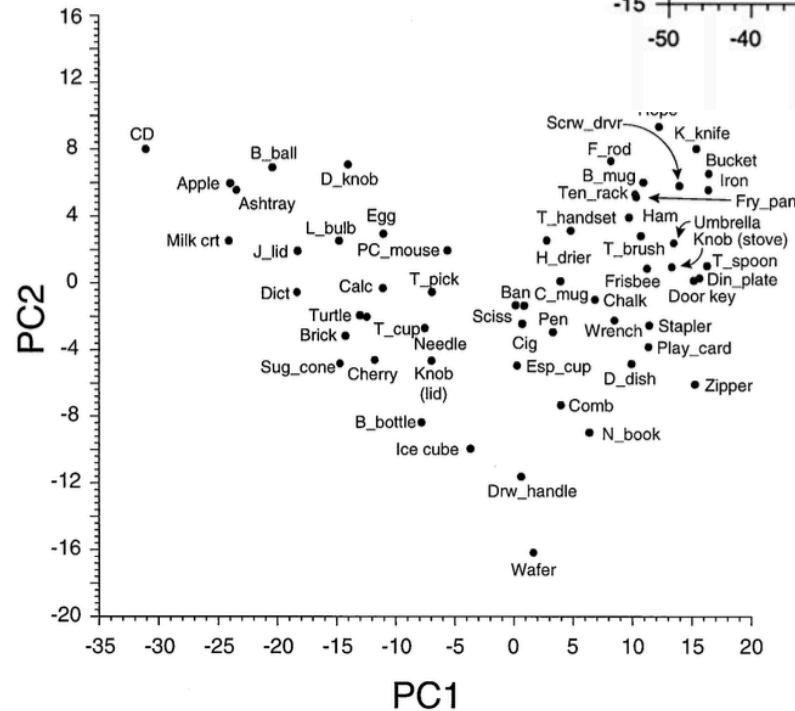
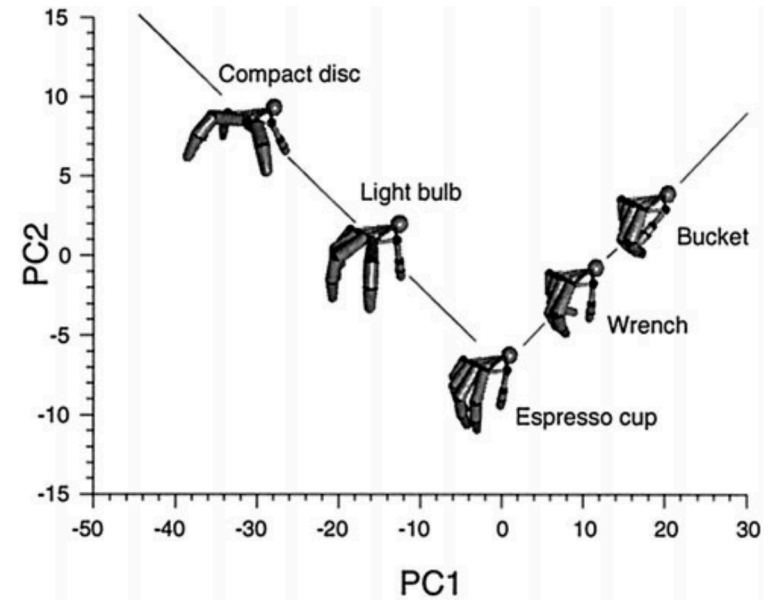
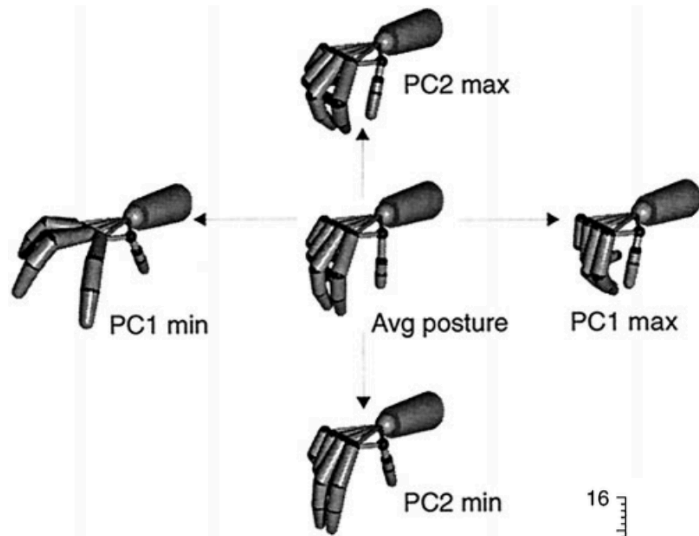


Picture	Type	Opp. Type	Thumb Pos.
	Power	Palm	Abd
	Power	Palm	Abd
	Power	Palm	Abd
	Power	Palm	Add
	Power	Palm	Add
	Precision	Pad	Abd
	Precision	Pad	Abd
	Precision	Pad	Abd
	Precision	Pad	Abd
	Precision	Pad	Abd

Picture	Type	Opp. Type	Thumb Pos.
	Power	Palm	Abd
	Power	Palm	Abd
	Precision	Pad	Abd
	Precision	Pad	Abd
	Precision	Pad	Abd
	Power	Palm	Add
	Intermediate	Side	Add
	Power	Palm	Add
	Power	Pad	Abd

Picture	Type	Opp. Type	Thumb Pos.
	Power	Pad	Abd
	Precision	Side	Abd
	Intermediate	Side	Abd
	Precision	Pad	Add
	Intermediate	Side	Abd
	Precision	Pad	Abd
	Intermediate	Side	Add
	Power	Pad	Abd
	Precision	Pad	Abd





Human grasping postures are a low-dimensional manifold in posture space

- The technique:
  - Uses a tracked, translucent box as a prop with a world-fixed display
  - The manipulated object is rendered on the wall *behind* the box, such that it looks like it is *inside* the box



- Advantages:
  - Direct, intuitive object manipulation
  - Passive haptic feedback
- Disadvantages:
  - Tethering of the box
  - Reflections on the box
  - Users felt the object to be *less present*
- Unclear which technique is faster (object manipulation using specimen box or one-handed "sticky" object grabbing)

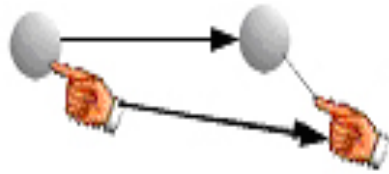


- Rubberband metaphor with automatic adjustment of precision
- Idea:
  - Detect when the user is trying to be precise and when not
  - Adjust C-D ratio ( $k$ ) accordingly
- More concretely:
  - Let  $D_O$  = translation distance of manipulated object,  
 $D_H$  = translation distance of user's hand,
  - Set distance

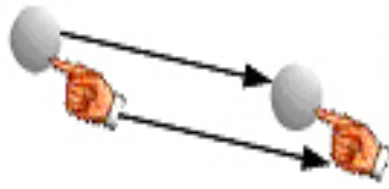
$$D_O = k \cdot D_H \quad k = \begin{cases} 1 & , V_H > S \\ V_H/S & , \min < V_H < S \\ 0 & , V_H \leq \min \end{cases}$$

with  $V_H$  = average speed of hand during past ½ second,  
 $S$  = some threshold;

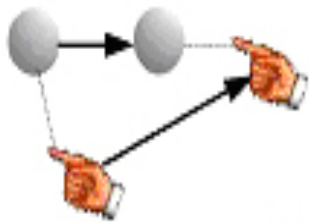
- Additional idea:
  - Do the scaling **independently** for each coordinate axis
  - Advantage: helps to move objects exactly along an axis of the coord system
- Recovery from offsets/drifts:
  - Problem: the positions of hand and object drift apart over time
  - Solution: reduce the offset while the user moves the hand very fast
    - Make the object move even faster
    - During fast movement, the user doesn't notice
- Remark: this technique works almost exactly analogously for rotations
  - Just convert the orientation of the user's hand to axis + angle (see CG1 course), scale the angle, then convert back to rotation matrix
  - In the original PRISM paper, it was implemented differently



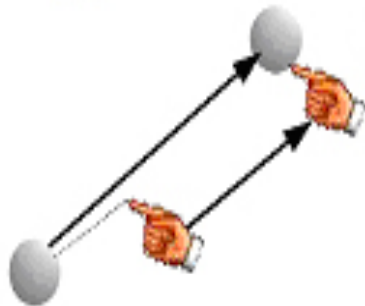
**a.** User moves hand slowly to the right and down. Some movement is scaled down in the horizontal direction, nearly all is scaled down in the vertical.



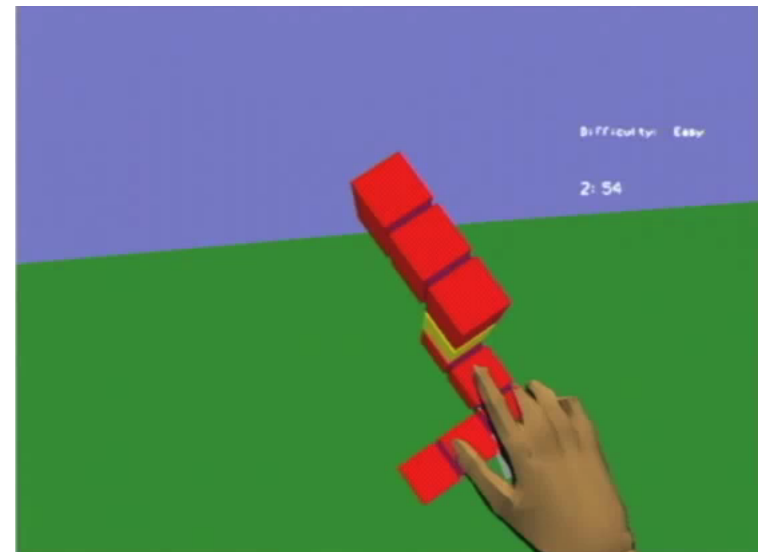
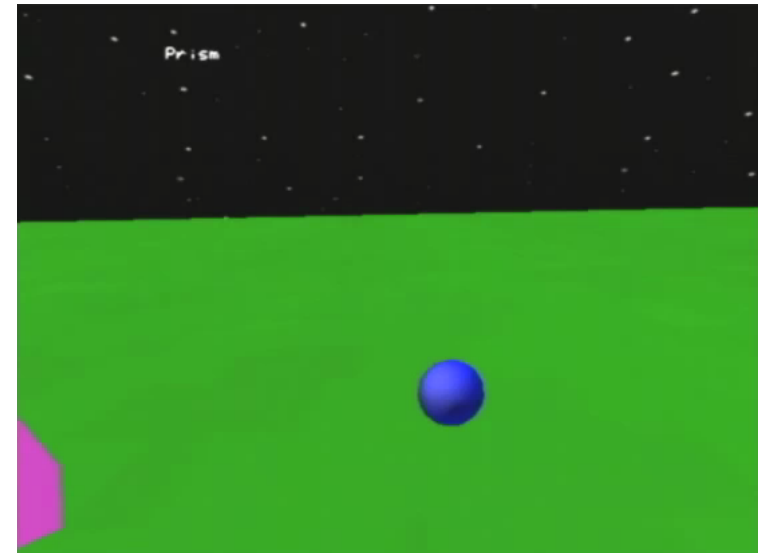
**b.** User moves in the same direction as in (a), this time quickly. Since interaction is in direct mode, object follows hand to its exact location.



**c.** User slowly moves to the right and up back towards object, vertical offset is recovered. Scaled motion performed in horizontal direction.

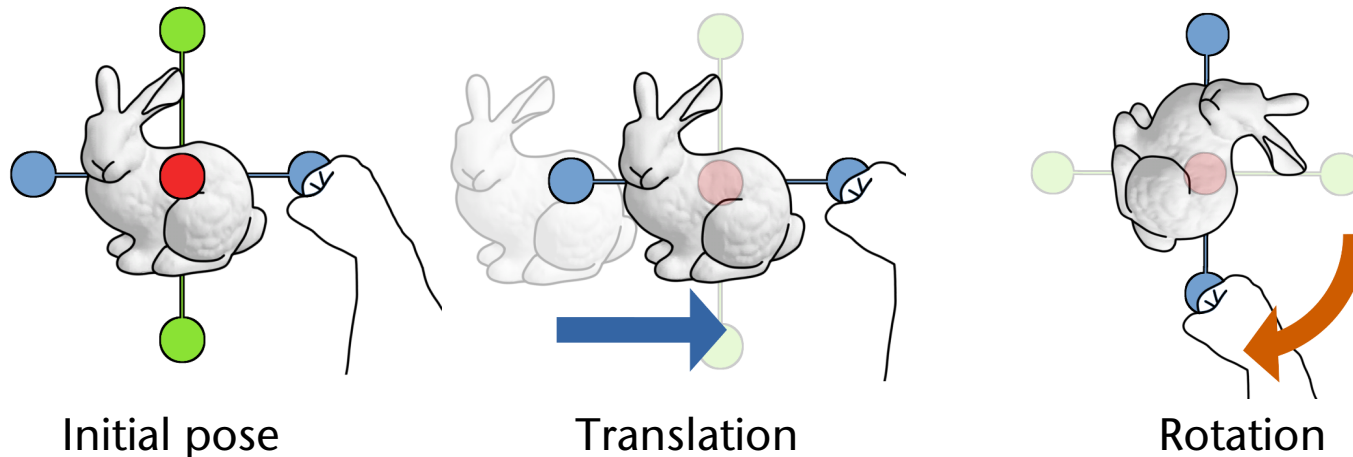


**d.** After accumulating offset, user moves hand quickly up and to the right. Offset is eliminated and mode has switched into direct.



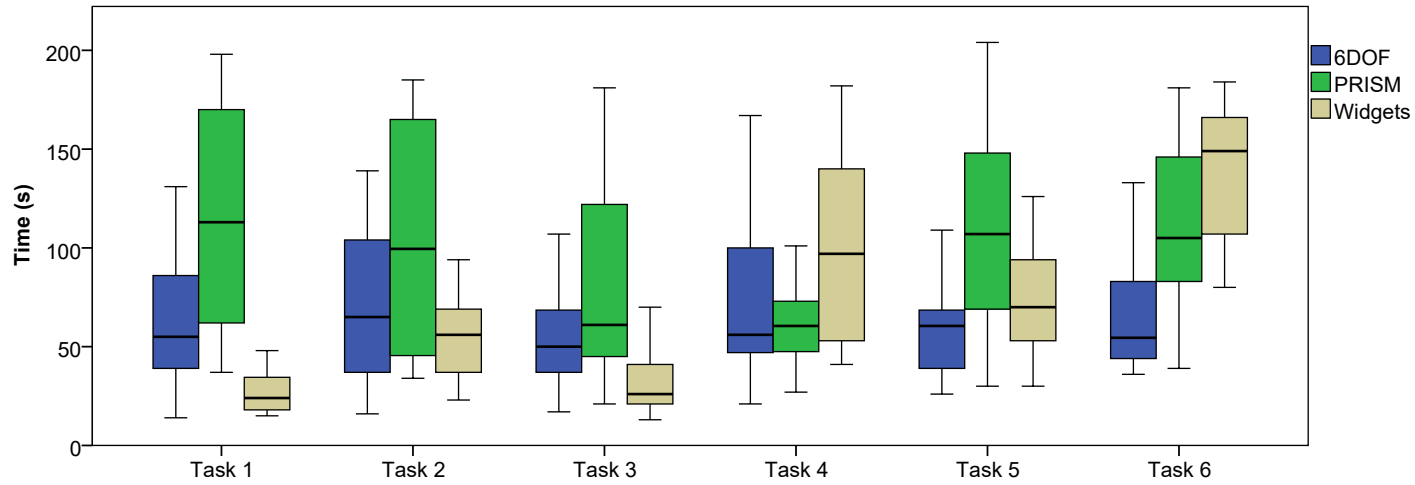
Frees, Kessler, Kay ( <http://give.ramapo.edu/prism/prism.html> )

- Idea: provide 3D widgets (with handles) so users know which DOF they are manipulating
- Example:

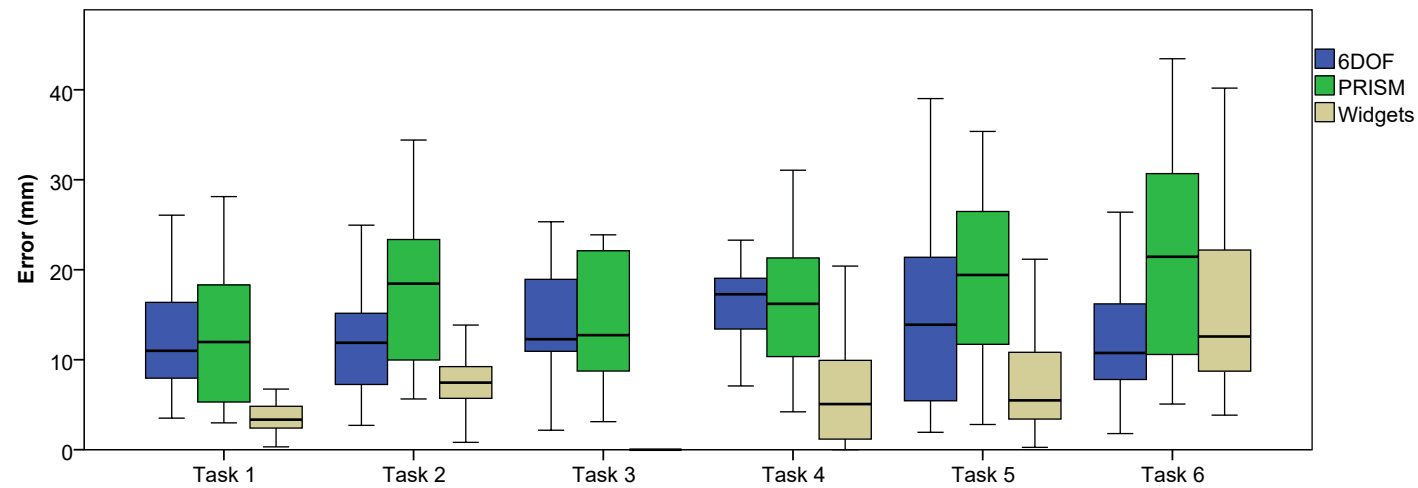


- Results:
  - Comparison for time-to-completion (TTC) between widgets technique ( 1 DOF) and 6 DOF manipulation (unconstrained, unseparated)
  - For translational tasks:  $TTC(\text{widget}) < TTC(6 \text{ DOFs})$
  - For rotational tasks:  $TTC(\text{widget}) > TTC(6 \text{ DOFs})$
  - For both task kinds:  $\text{error}(\text{widget}) \ll \text{error}(6 \text{ DOFs})$

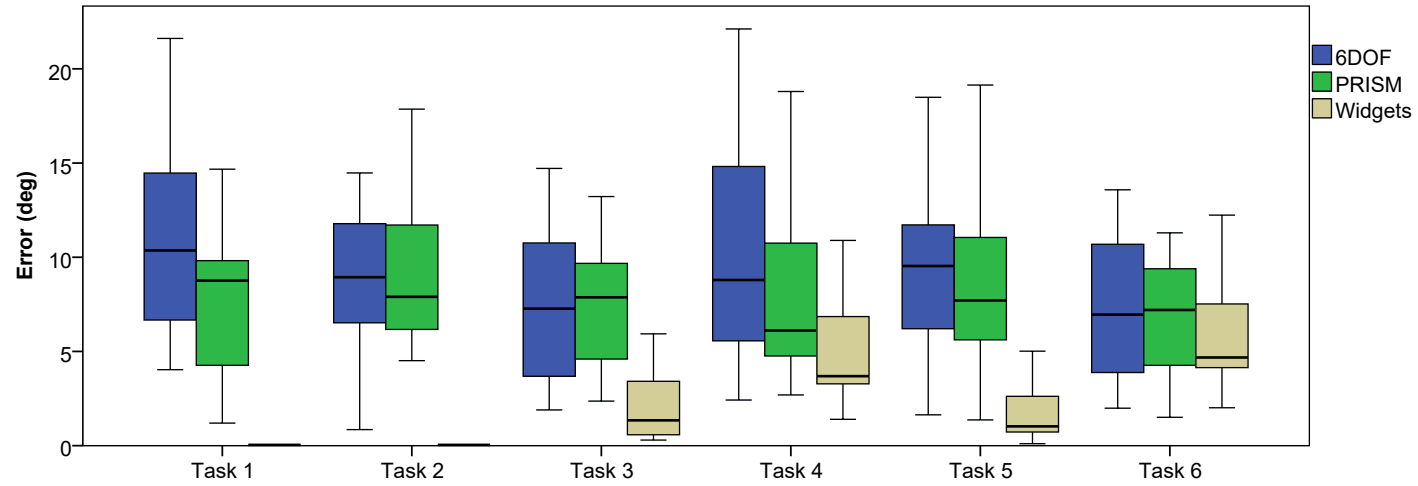
## Time-to-completion

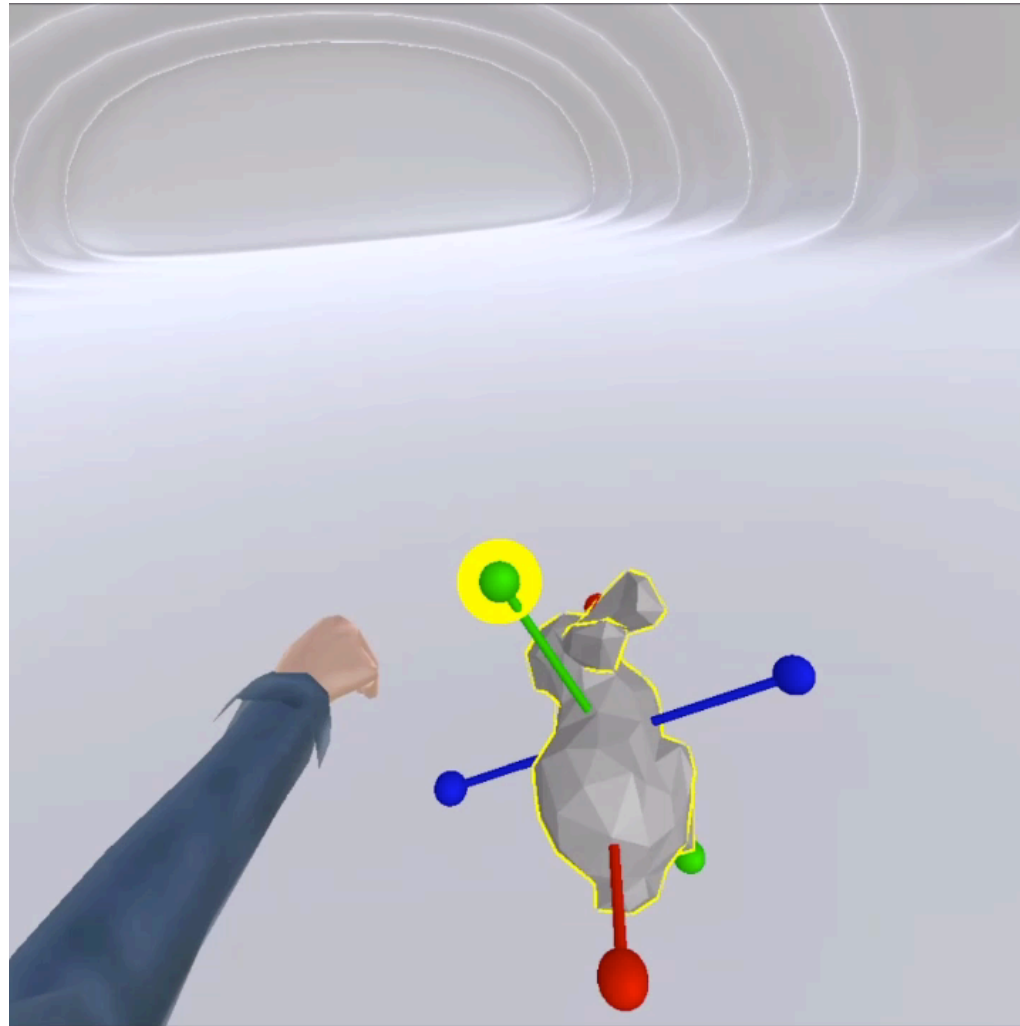


## Translational error



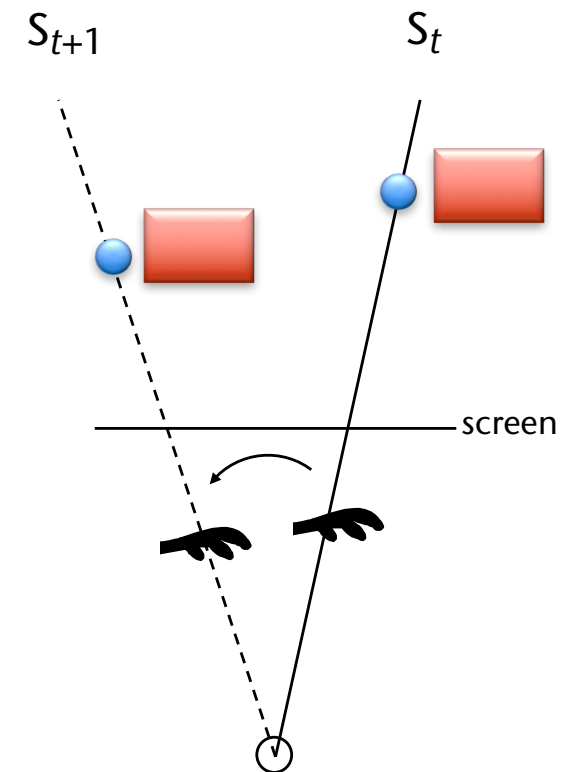
## Rotation error





# The Action-at-a-Distance Principle

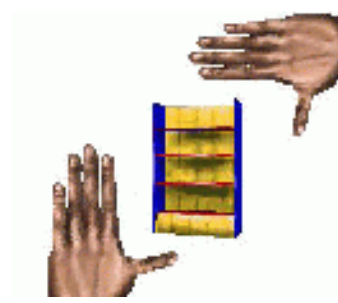
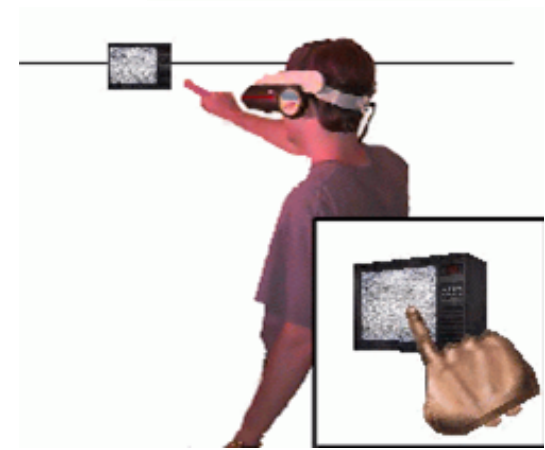
- Is a **general** VR interaction design principle
- Example: move objects in the distance
  - Idea: scale motion of hand such that **on the screen** (2D) the relative position between the hand and the object does not change
  - Computations:
    1.  $d_O^t$  = distance obj – viewpoint at time  $t$
    2.  $P^t$  = point on ray  $S_t$  with distance  $d_O^t$
    3.  $d_H^t$  = distance hand – viewpoint at time  $t$
    4.  $d_H^{t+1}$  = distance hand – viewpoint at time  $t+1$
    5. Calculate  $d_O^{t+1}$  such that  $\frac{d_O^t}{d_H^t} = \frac{d_O^{t+1}}{d_H^{t+1}}$
    6. Calculate  $P^{t+1}$  = point on ray  $S_{t+1}$  with distance  $d_O^{t+1}$
    7. Translation for the object =  $P^{t+1} - P^t$





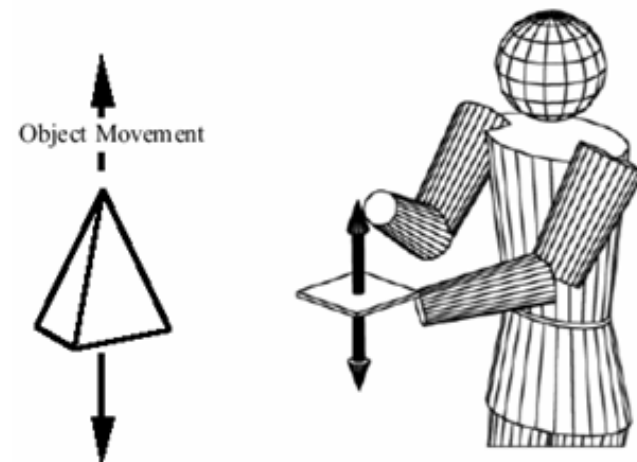
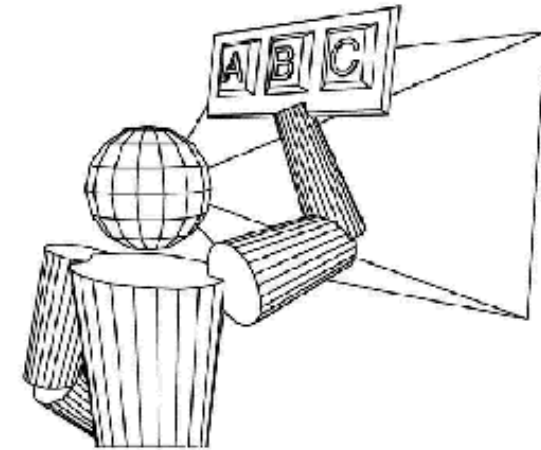
# Image Plane Interaction

- General idea: user does not interact with 3D objects, but instead with their 2D image
- "Image plane" selection metaphors:
  - Shoot a ray between thumb and forefinger
  - Shoot ray from eye through fingertip
  - "*Lifting palm*"
  - Frame the object with both hands



# Proprioceptive Interaction

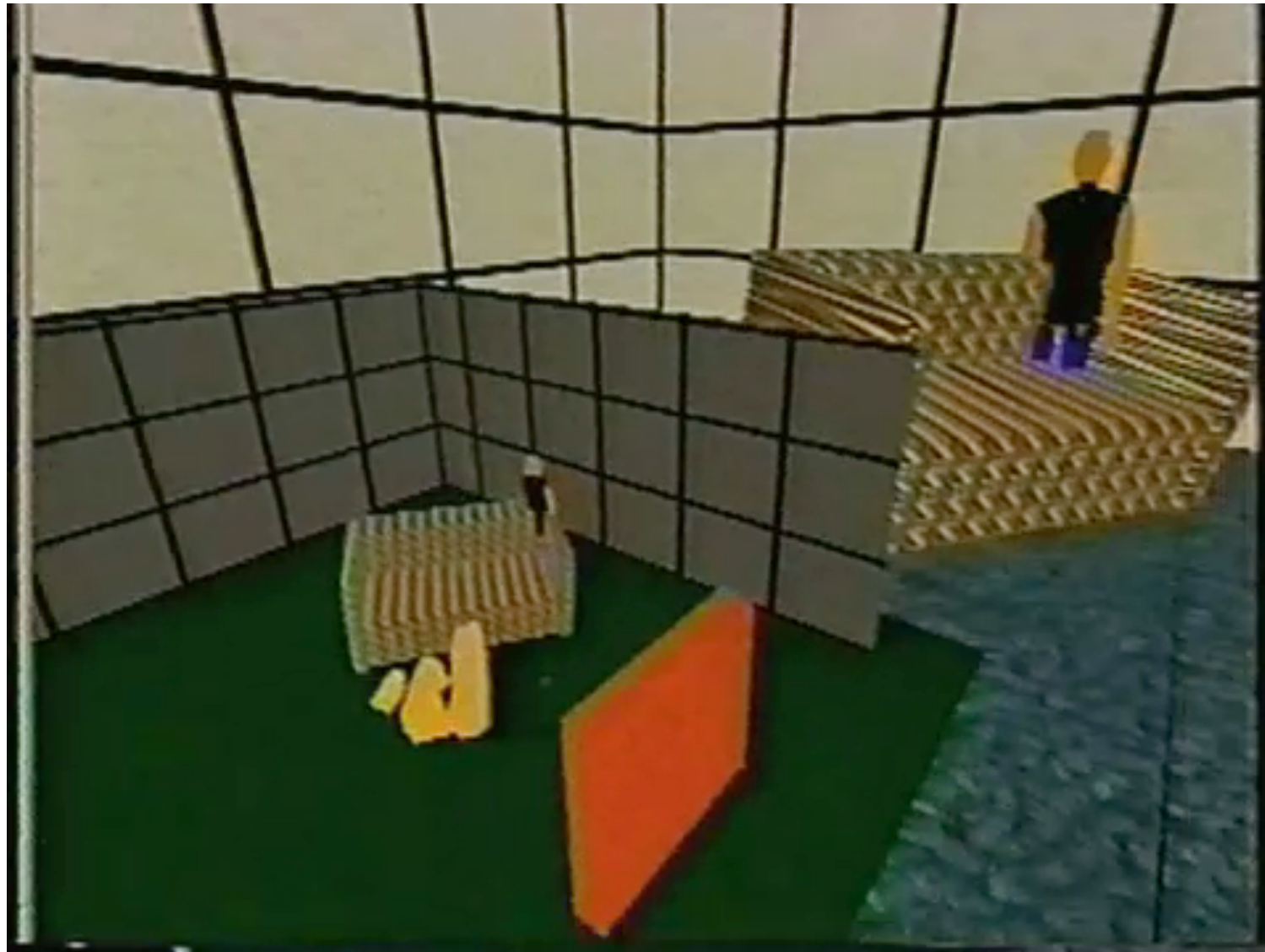
- *proprius* (lat.) = (adj.) *self*
- Idea: utilize the fact that humans know exactly where their limbs are, even with **closed eyes**
- Metaphors derived from that:
  - "Real pulldown" menus: user reaches up, makes grasping gesture, then pulls her hand down → menu appears
  - Deleting objects: grasp object, throw over the shoulder
- Manipulate remote objects by handheld proxy widgets (= action-at-a-distance)



# The World-in-Miniature Paradigm

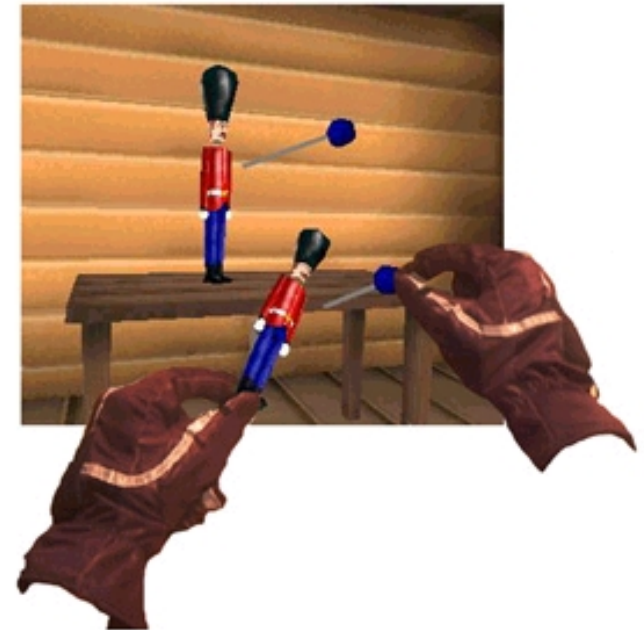
- Idea:
  - Use a 3D miniature "map" (analogously to 2D mini-maps) → World-in-Miniature (WIM)
  - All interactions in and with the WIM are mapped to the "real" VE
  - Attach the WIM to the non-dominant hand
- Object manipulation = grasp & move the miniature object in the WIM
- Navigation = move the frustum in the WIM, or select a point in the WIM





Doug Bowman

- Technique for remote manipulation / positioning of objects
- Idea: create a temporary copy (= voodoo doll) of the remote object
- Task decomposition:
  - Create copy of the referenced object, attach it to the left hand
    - The original of that object will *not* be moved
  - Create copy of the object to be manipulated, attach it to the right hand
    - The original of that object *will* be moved
  - Movement of the voodoo doll
    - **relative to the copy of the reference** object —  
is mapped to the original



- How to create the copy of the object to be manipulated:
  - Use image-plane technique: make pinch gesture "in front" of the object
  - Size of the copy  $\approx$  size of the virtual hand
- How to create a copy of the reference object(s):
  - Use some framing technique (= image-plane technique again)
  - Make copy of all objects within the frame





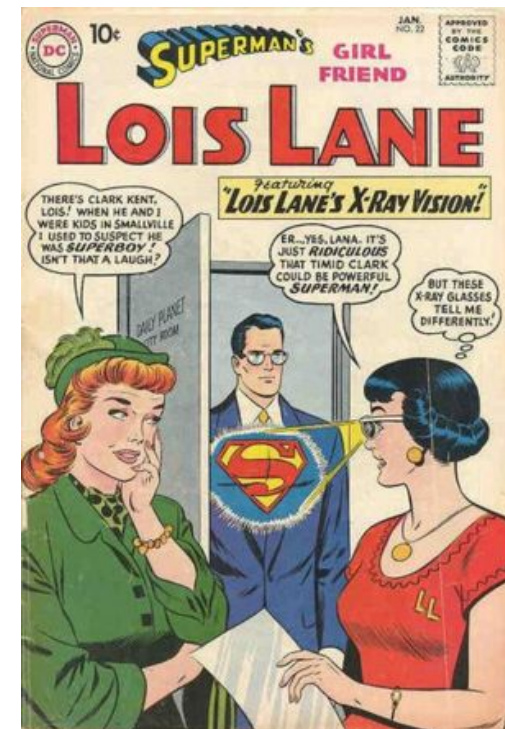
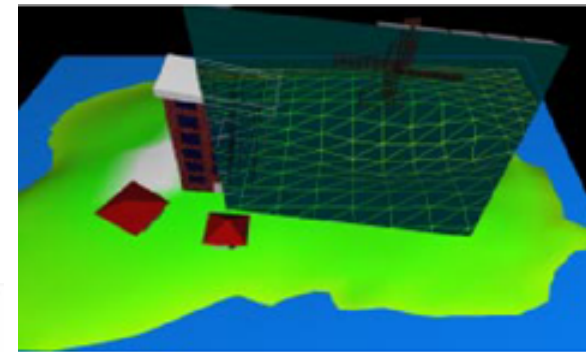
- Example of a manipulation:
  1. User "grasps" table with pinch gesture of the left hand
  2. System creates copy, attaches it to left hand, plus some other "context" objects in the surroundings of that object (e.g., telephone and monitor)
  3. User "grasps" telephone with right hand (pinch)
  4. System creates copy of telephone and attaches it to right hand
  5. User puts *copy* of telephone at some other place on the *copy* of the table
  6. System maps the translation to the original telephone

- Advantages:
  - Left hand is being used exactly for what it was "designed"
  - User can work on different scales, without having to specify the scaling explicitly
    - The scaling happens implicitly by selection of the reference objects



# Magic Lenses

- Idea: user sees a different version of the VE through the magic lens
- Where "different" could mean:
  - Other rendering parameters
  - Other geometry
  - Another viewpoint
  - Different scaling, ...
- Examples:
  - Wireframe rendering
  - Magnification
  - Additional viewpoints (like magic mirror)
  - Geometry beneath the surface
  - Preview window for *eyeball-in-hand* or *scene-in-hand* navigation
  - "X-Ray vision"
- Magic lenses can also be specified by volume



## X-Ray Tool

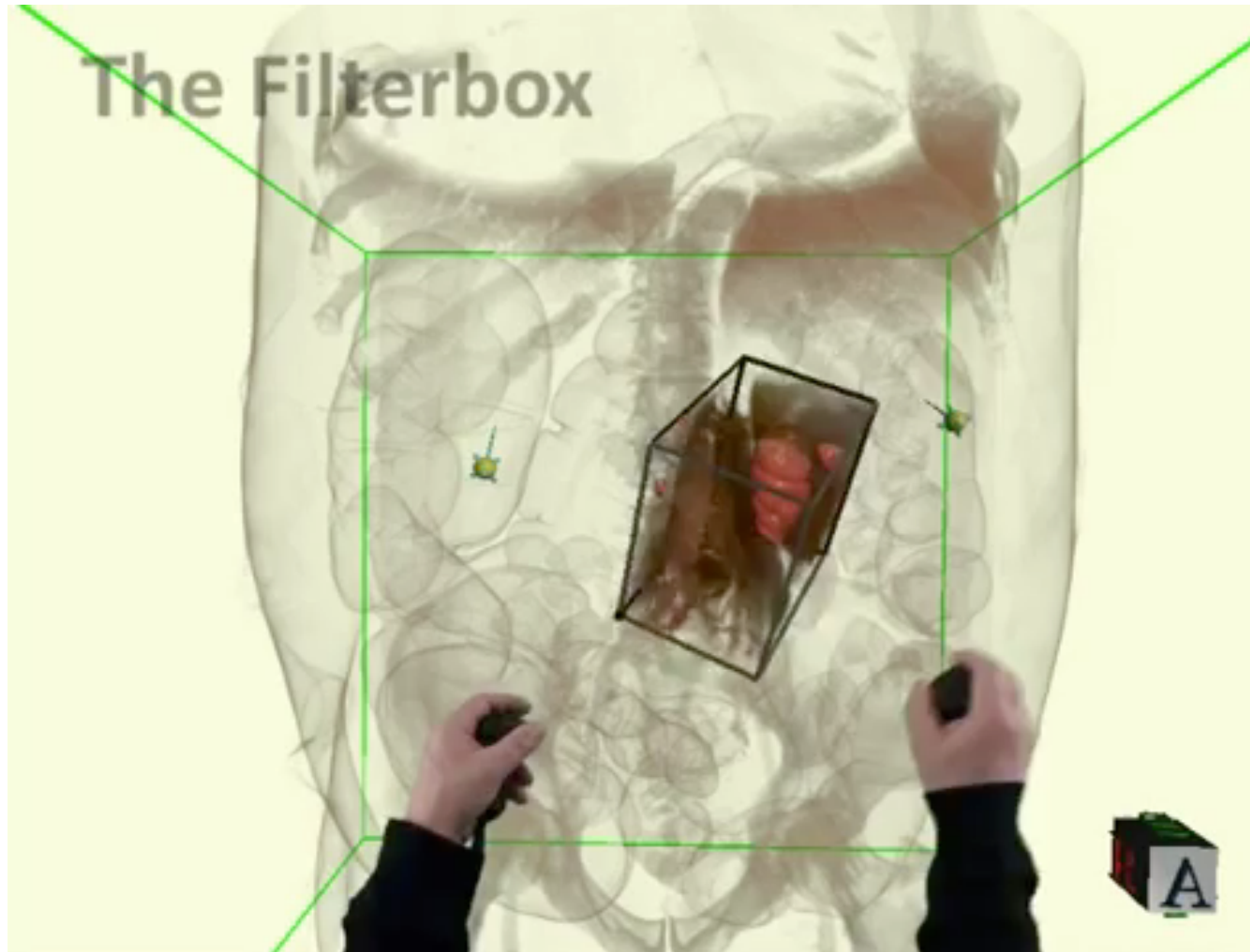
- Laying underground cables

## Window Tool

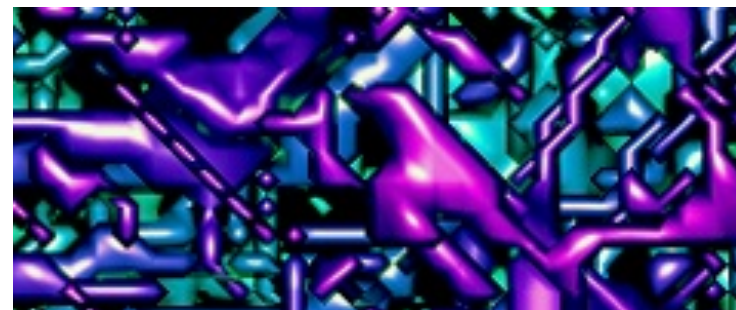
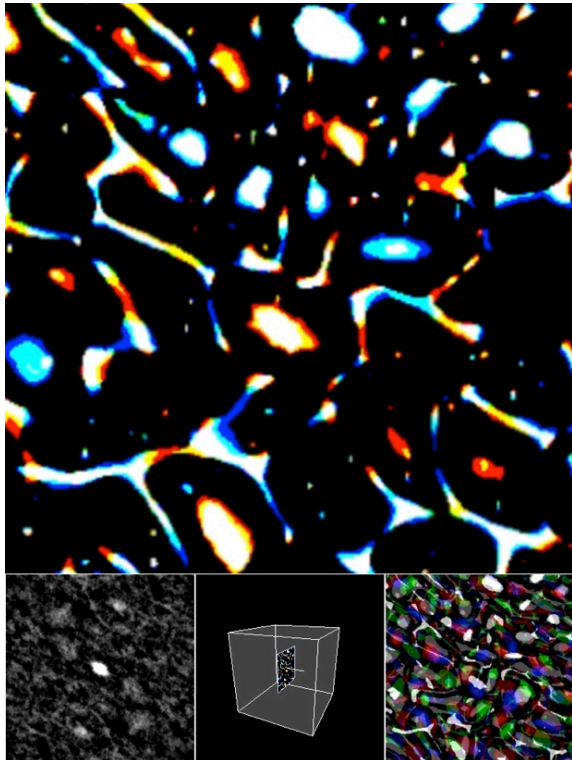
- Comparing different versions of a scene

## Window Tool

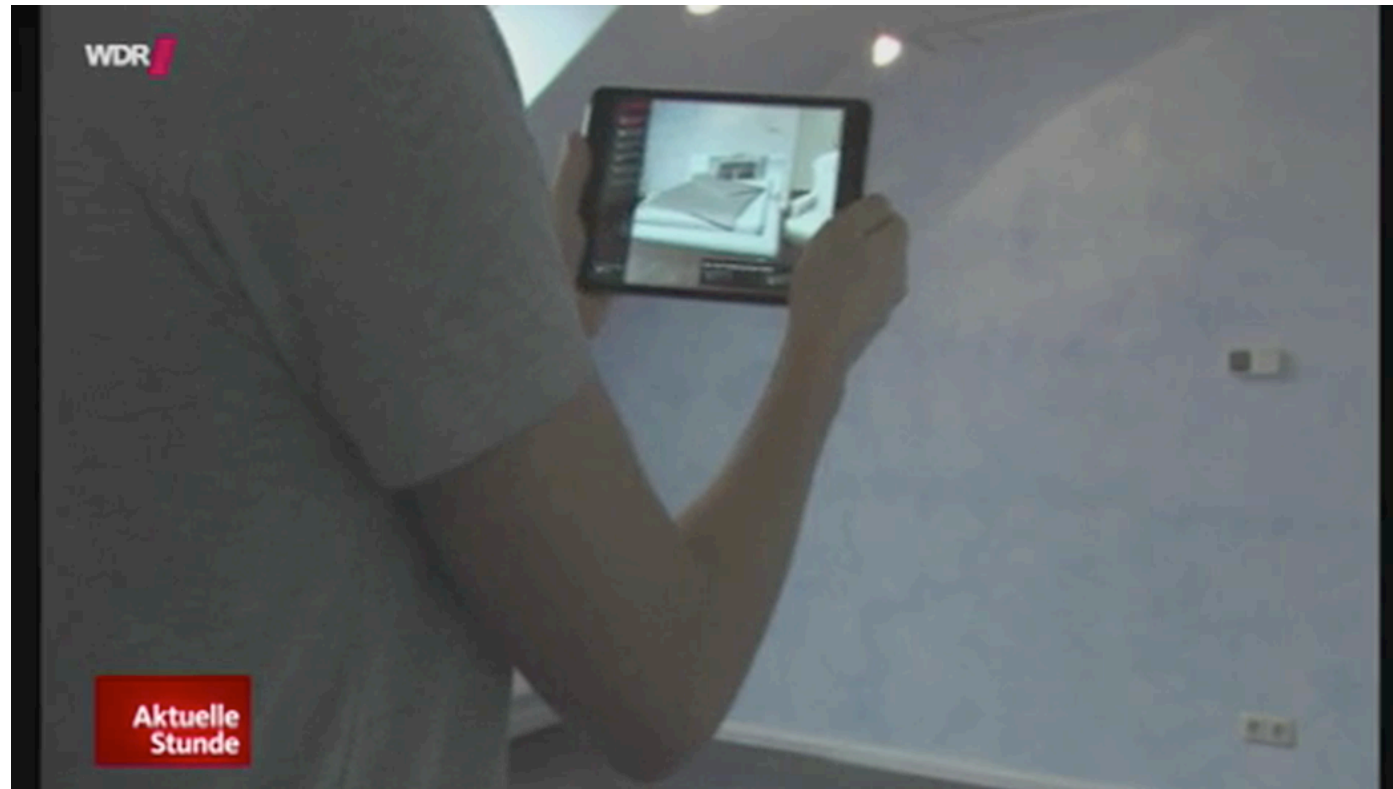
- Multiple Views into scene



- Task:
  - Visualization of volume data (here, CT) on an iPad
  - Intuitive navigation (= specification of the viewpoint)
- Solution: regard the iPad as a "magic lens" into the VE



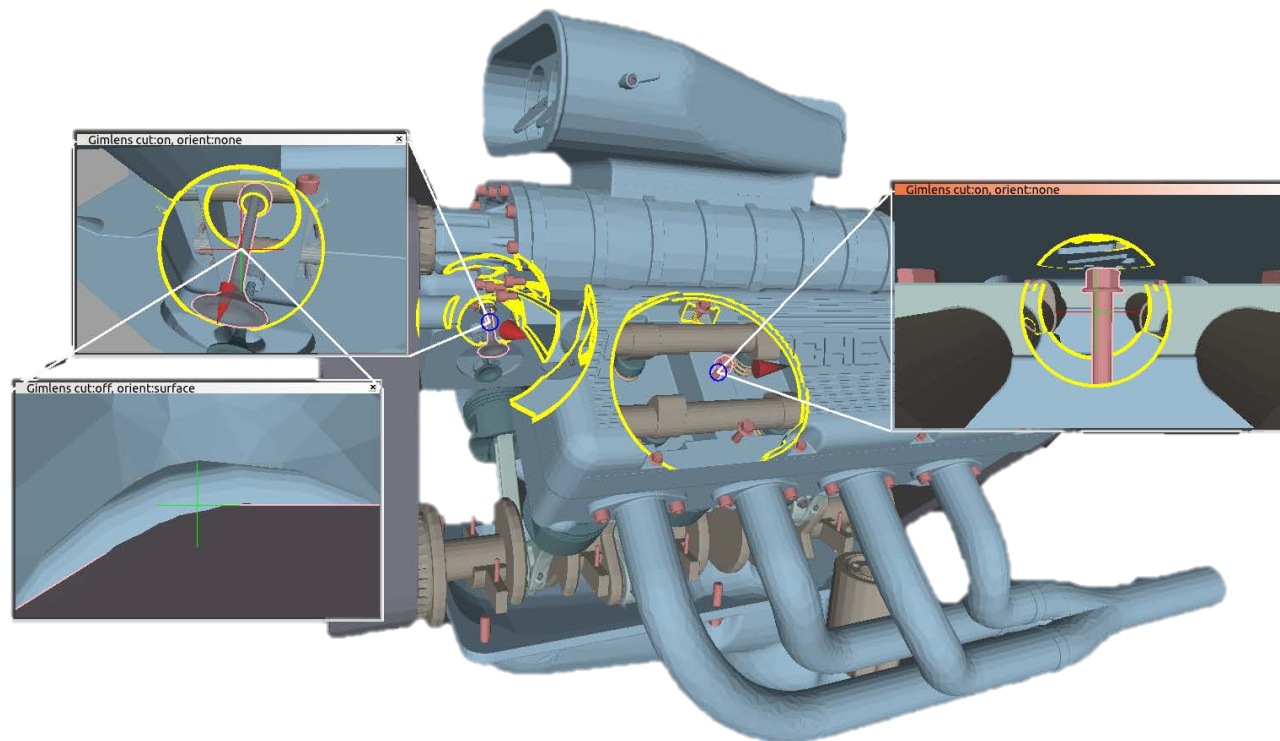
- In a sense, a magic window is an interaction metaphor that works just like some of the augmented reality apps on tablets
  - (You might argue, a magic window implements "augmented virtual reality")

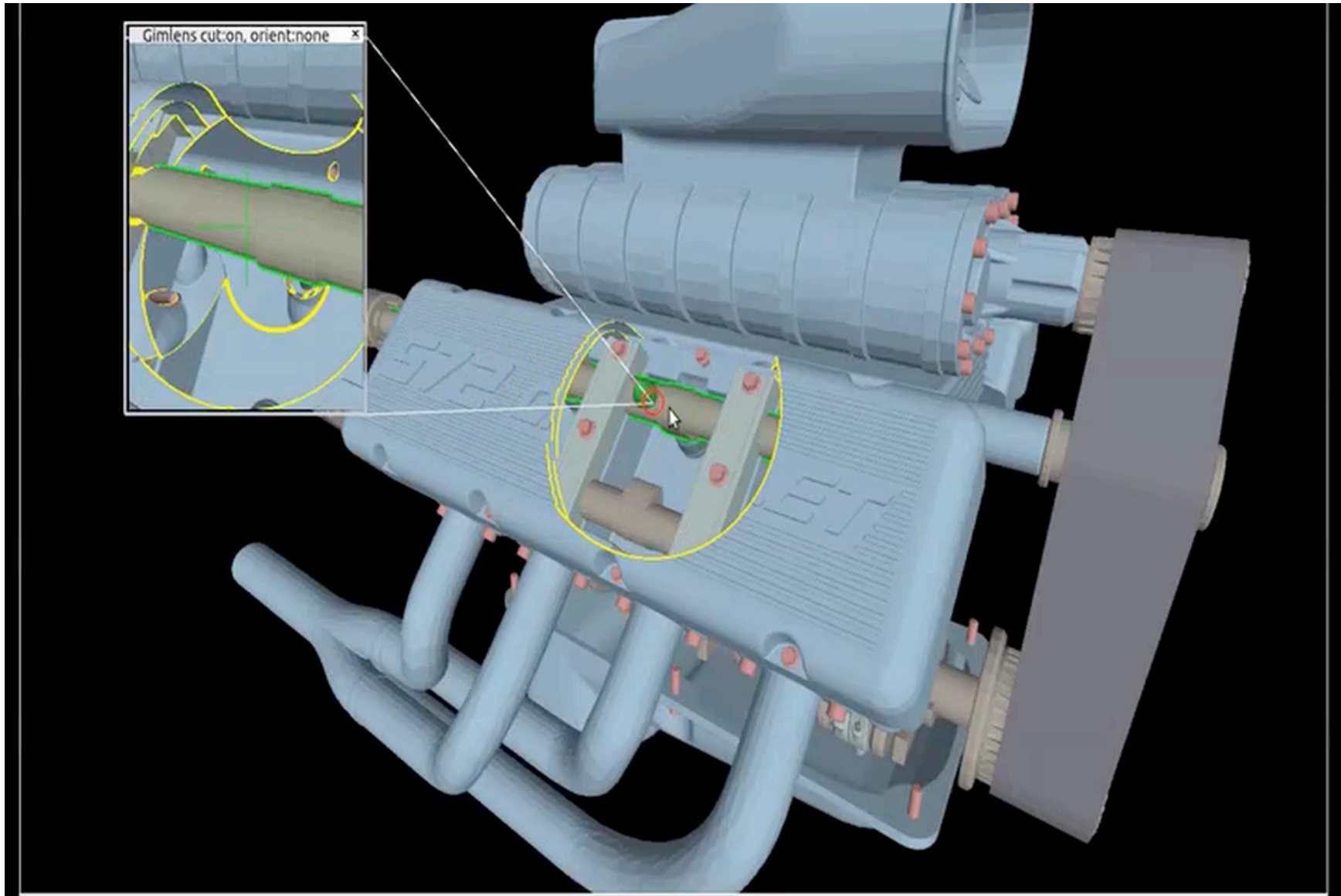


WDR, Aktuelle Stunde, 23. 1. 2015



- *Gimlens* = magic lens to specify cut-aways
- Cut-away = truncated cone
- Positioning by cone-shaped proxies
- Implementation: fragment shader tests fragments against the cone





# Redirected Walking

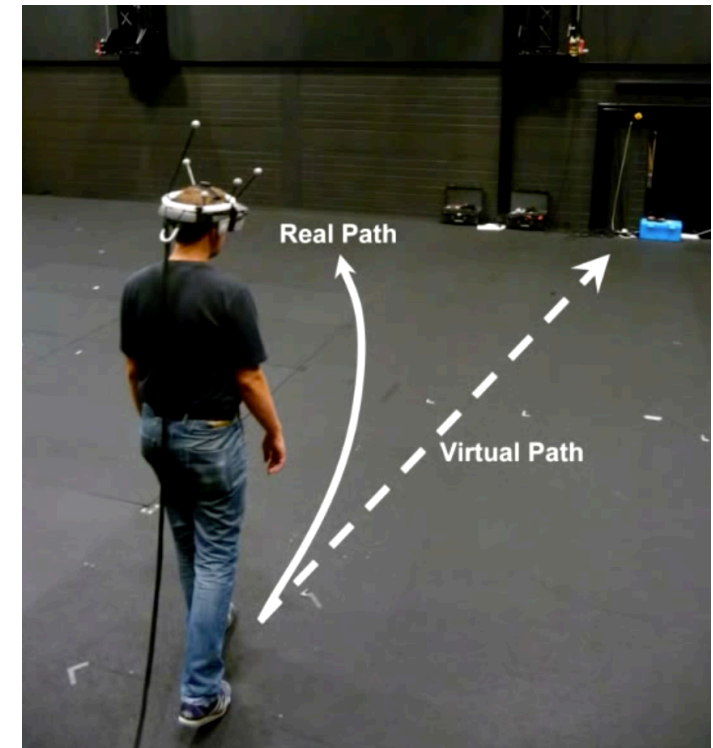
- Walking in real space = very natural, intuitive navigation in VEs
- Problem: virtual space can be much larger than the physical space, in which the user is confined in the real world
- Challenge: how to make large VE accessible without additional navigation metaphors?
- One solution: "fence" in the VE around user before they hit any real obstacles

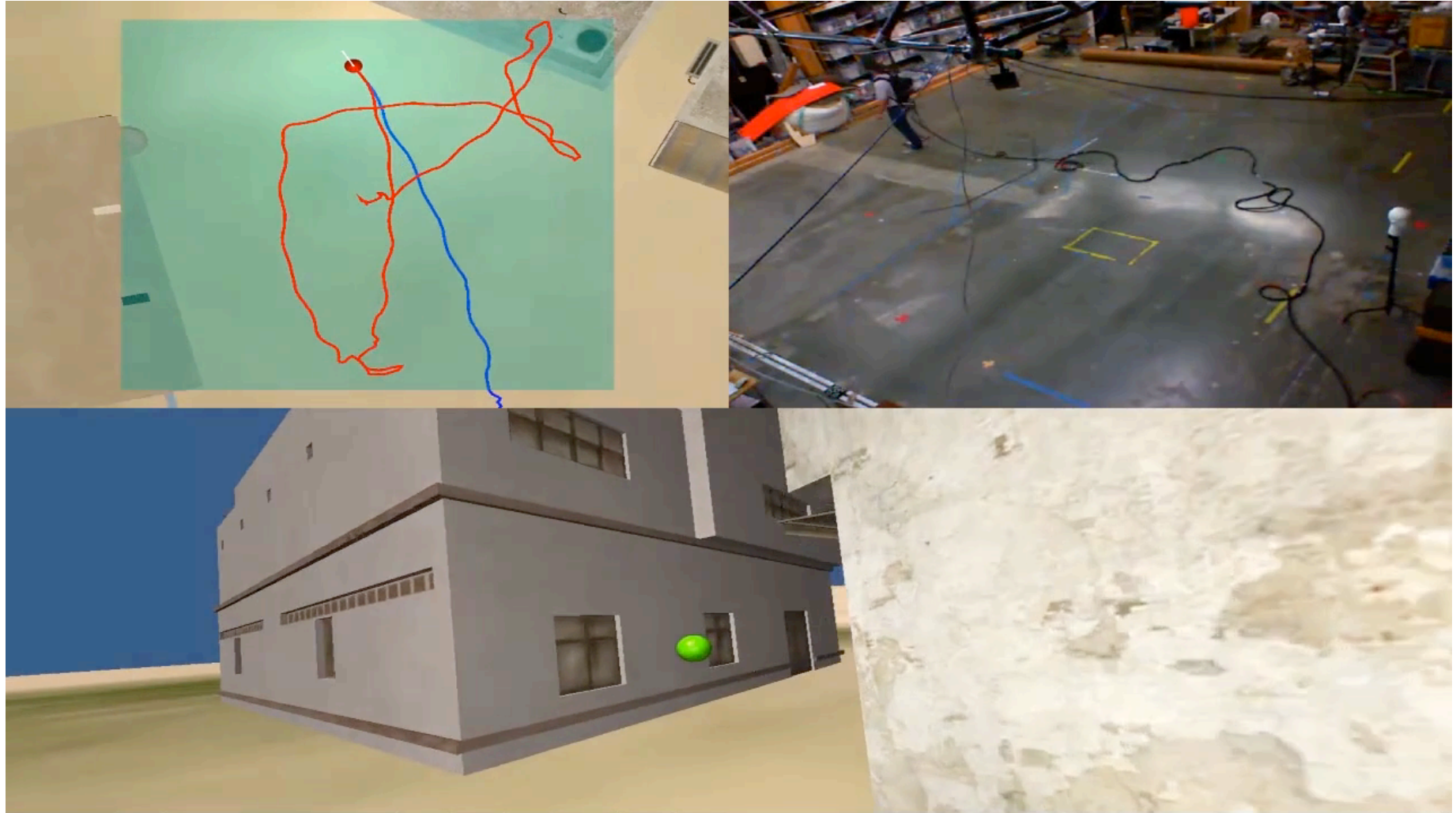


Azmandian, Grechkin, Phan, Bolas, Suma



- Possible solution: "cheat" on user
  - Map *distance* of user's walking in **real** space to **smaller/larger distance** of avatar in virtual space → **translation gain**
    - Somewhat similar to the CD ratio
  - Map *angles* of user's turning in real space to smaller/larger turning angles of avatar in virtual space → **rotation gain**
  - Make mapping or change in position/orientation **unnoticeable** to user:
    1. Choose unnoticeable gains that create only small deviations between virtual/real
    2. Perform rotational/translational "jump" during a saccade (needs gaze tracking in HMD)
    3. Distract user, e.g., by suddenly appearing objects (sign posts, birds, ...)

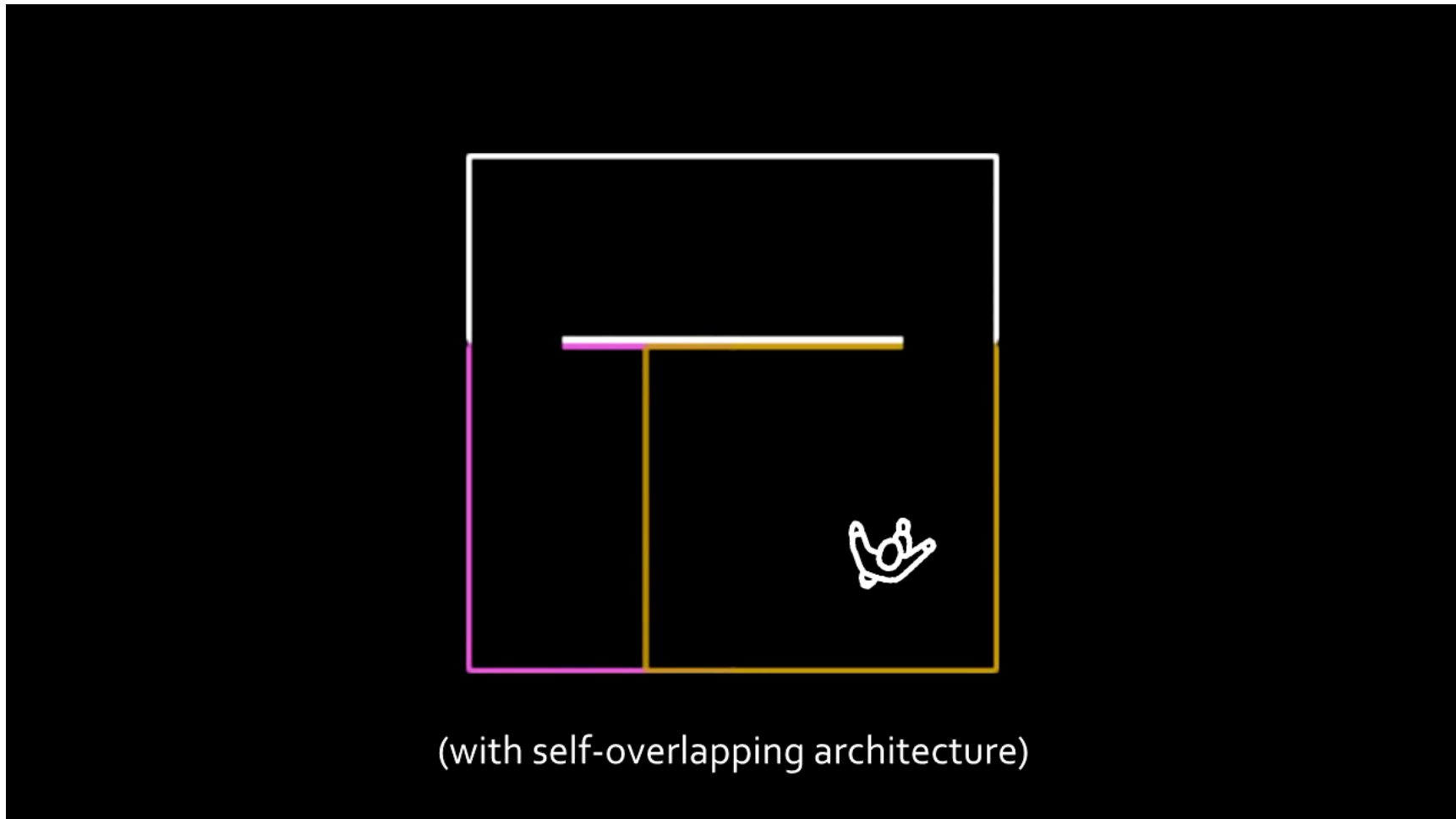




Azmandian, Grechkin, Phan, Bolas, Suma

# Related Technique: Spatial Illusions

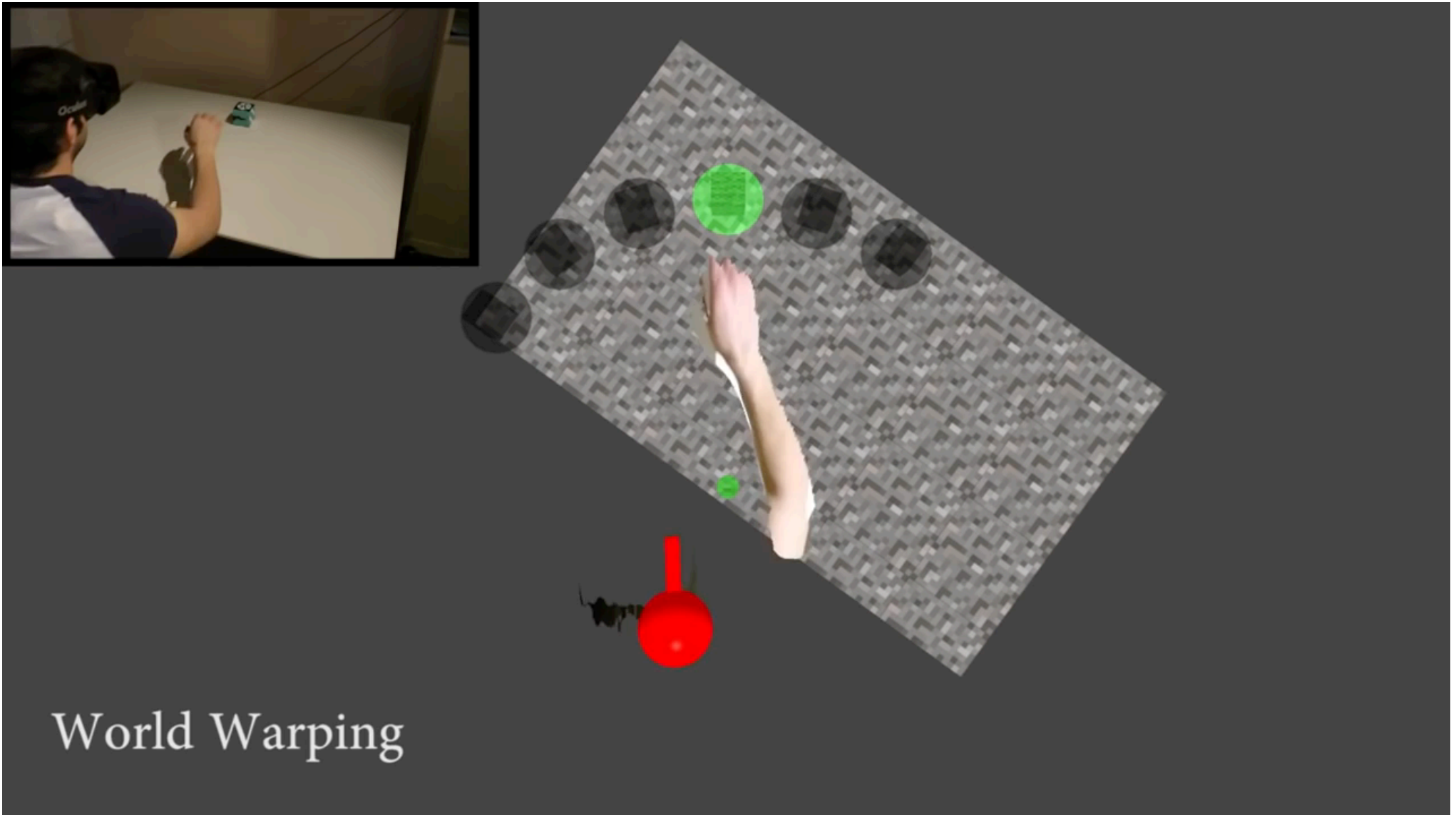
- One possibility: self-overlapping architectural spaces



# Categories of Redirecting Users

- Continuous redirection by translation and rotation gains
- Discrete redirection by making the user reorient through specific events, e.g., barriers or "distractors" (not shown here)
- Redirection by changing the geometry of the VR

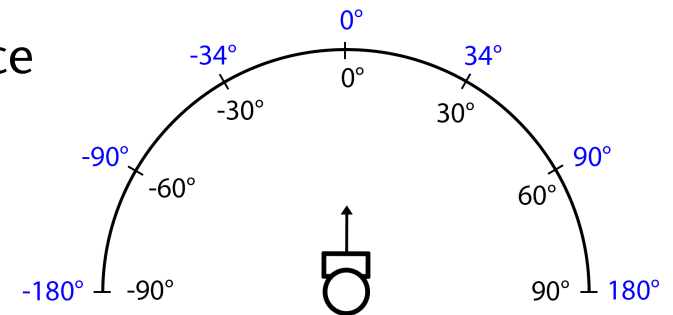
- Goal: provide rich passive haptic feedback with minimal physical haptic props
- Setting: HMD, tracked hand, seated user
- Idea: apply redirection techniques to hand location and user orientation
  - "Body warping" = redirect user's hand + slight avatar deformation
  - "World warping" = rotational gain



- Manipulates C/D ratio for user's head rotation
  - Remember the Go-Go technique for the user's hands?
- Especially useful for seated VR experience
- The technique:
  - User defines preferred forward direction → yaw angle = 0
  - Only yaw angle (rotation about vertical axis) is modified
  - Yaw angle (a.k.a. heading) is multiplied by factor (nonlinear):

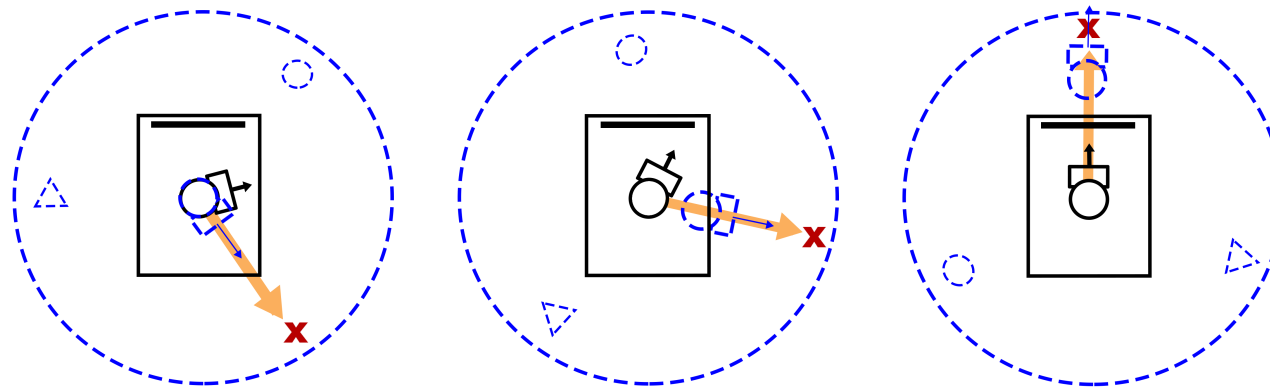
$$\theta_v = (2 - \cos(\theta_r)) \cdot \theta_r$$

where  $\theta_{r/v}$  = yaw angle in physical/virtual space





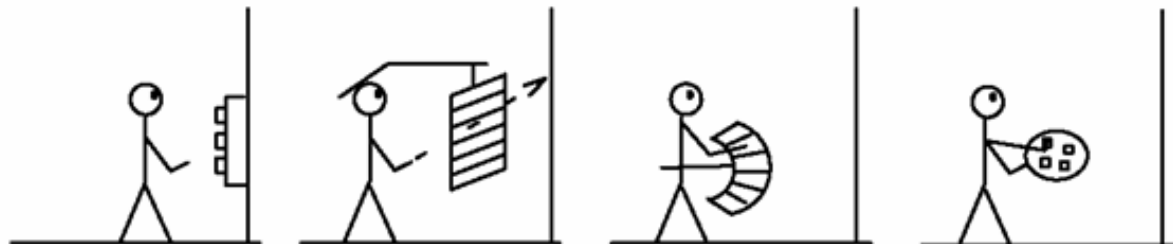
- Amplification with wash-out: gradually realign a user's head orientation with the preferred forward direction as they virtually move (translate) through the VE





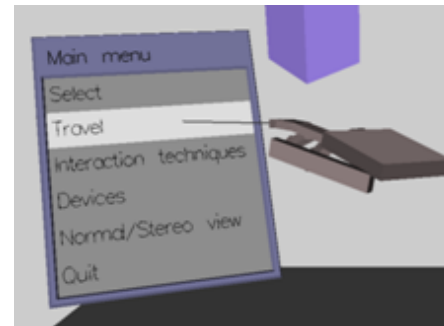
- The 3<sup>rd</sup> big category of interaction tasks in VR:
  - The somewhat unbeloved child in the VR interaction community
  - The general task of these interactions: change the system's state
    - and everything else that doesn't fit anywhere else
- Consequence: a taxonomy is almost impossible
- Typical techniques:
  - Menus
  - Speech recognition
  - A set of gestures (for 1-out-of-n triggers)
  - Physical devices

- Task decomposition:
  1. Bring up menu
  2. Navigate the menu
  3. Select an item
- A possible taxonomy should contain:
  - Input modalities: gestures, speech, buttons, ...
  - Positioning of the menu
  - Selection of items
  - Dimension and shape of the menu
- Examples for positioning the menu:

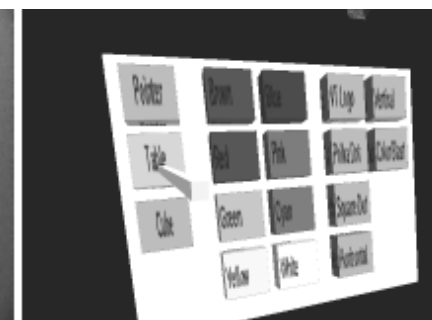
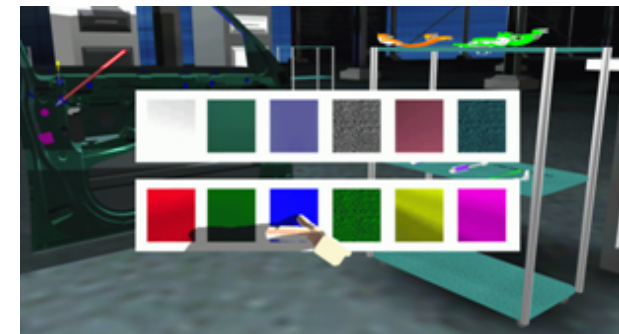
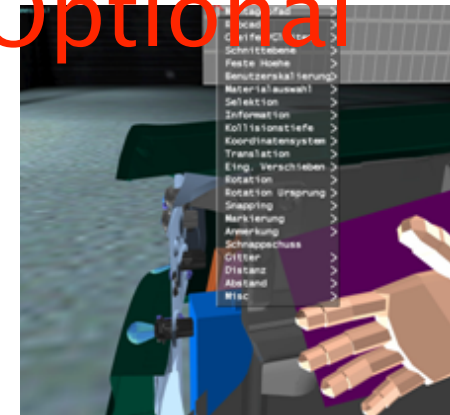


# Examples

- Embedded in 3D or 2D overlay (heads-up)
- Item selection: one of the earlier selection techniques, e.g., ray casting or occlusion technique, or map relative hand motion to "active" menu item
- Positioning:
  - Fixed in 3D,
  - Heads-up (moves with head),
  - Attached to left hand, ...

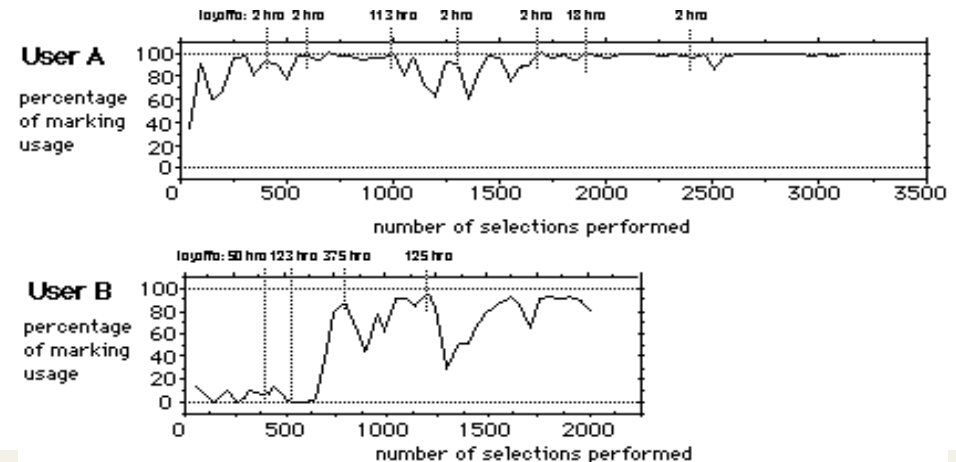
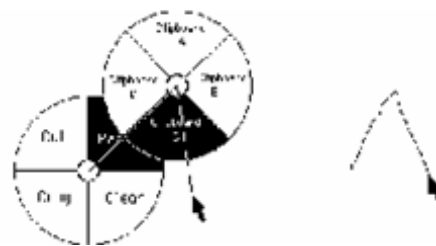
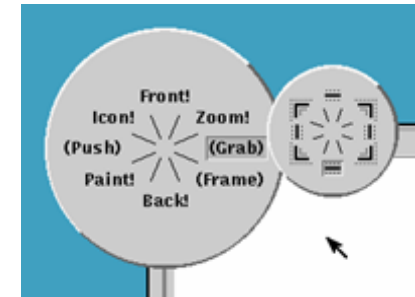


Optional



# 2D Digression: "Marking Menus" (a.k.a Pie Menu) Optional

- Idea: arrange menu items around as center (in a circle, square, ...)
- With menu trigger: position its center at current pointer position
- Advantage:
  - Smooth transition from **novice mode** to **expert mode**
  - Experts can navigate the menu "blindfolded"
- Mouse gestures (marks) are much more efficient than a menu:





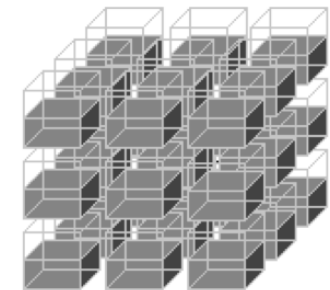
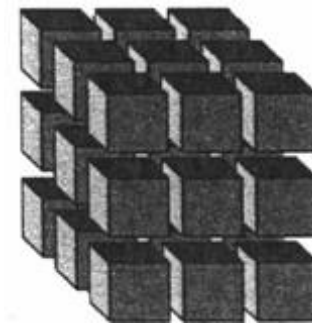
- Video (2D):



[SecondLife]

- In 3D?

- Direct "translation" → "control cube"
- Not too successful
- Can you do it better?



# A Few General Design Principles

Optional



- Two-handed interaction
- Action-at-a-distance
- Proprioception
- Task decomposition
- Dimension decomposition
- Multimodal interaction

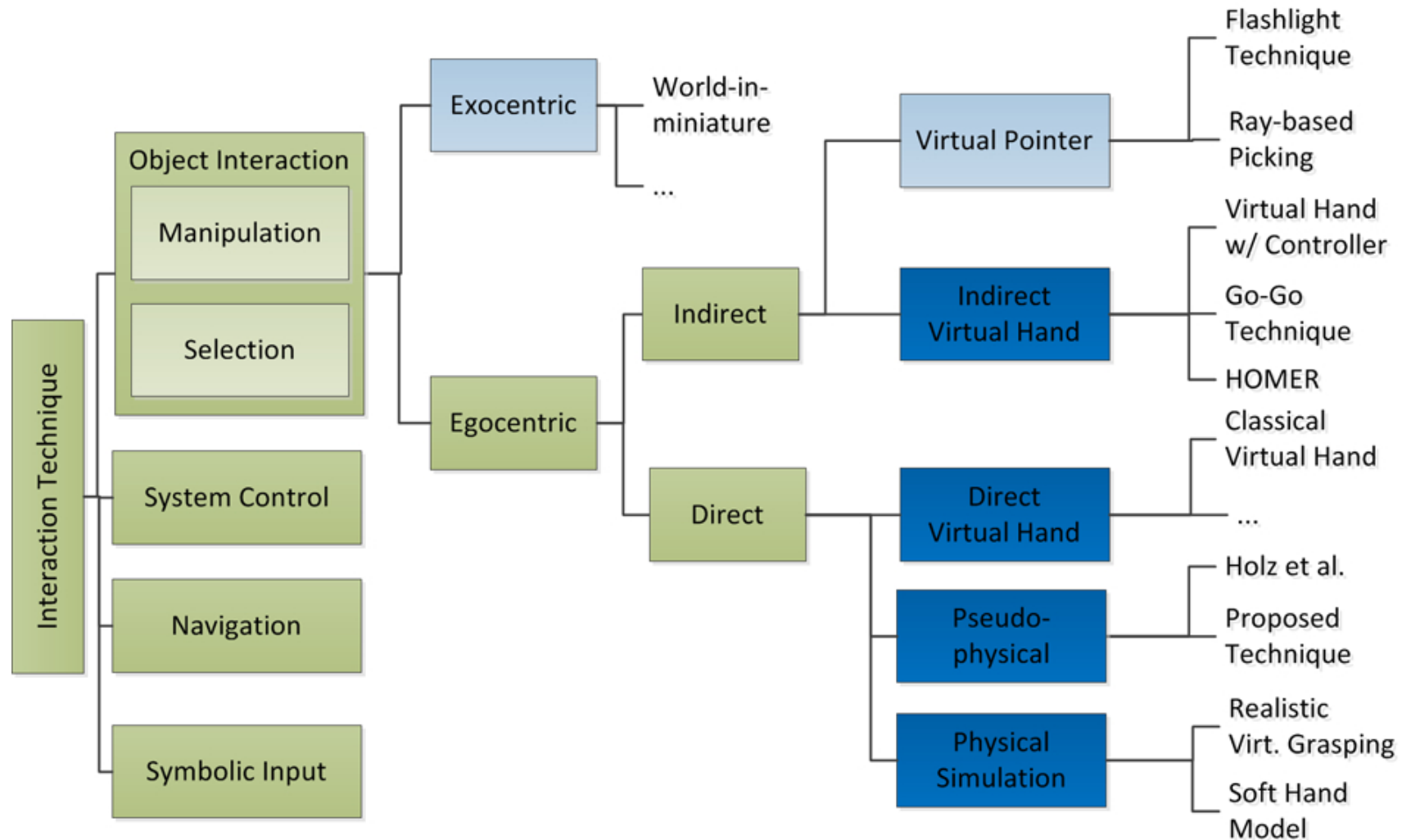


- Users have 2 hands: a **dominant** one (usually right) and a **non-dominant** one (left)
- The roles of each hand:
  - Non-dominant hand = reference coordinate system, positioning of context
  - Dominant hand = fine-skilled motor tasks within that context
- Good metaphors = metaphors that utilize **both hands** within their **respective roles**
- Examples already seen: iSith, balloon, WIM, ...

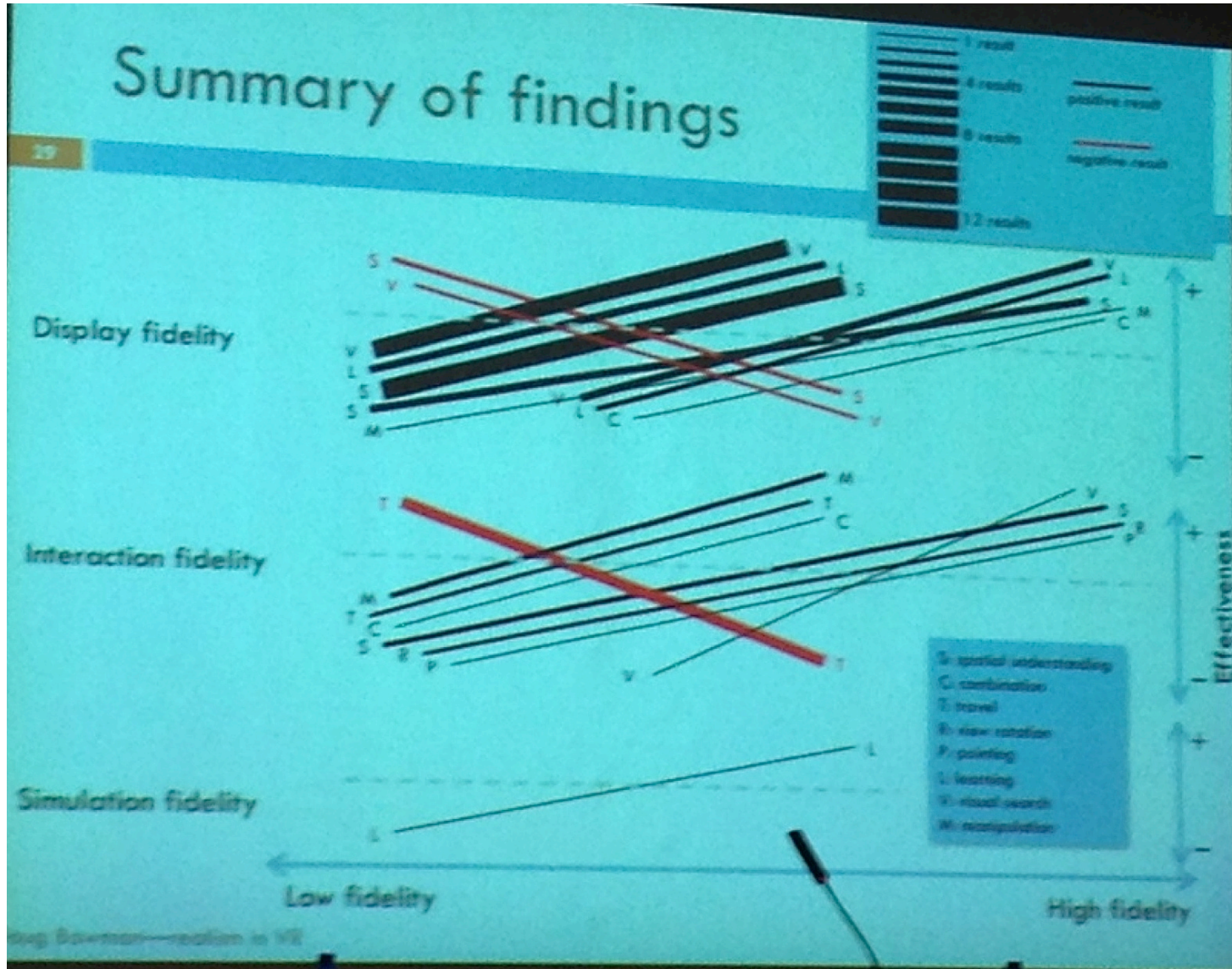
- Single input:
  - Specialized:
    - Single modality that is clearly ideal and sufficient
    - Ex.: Big red emergency button
  - Different but equivalent:
    - User chooses one of several modalities, each gives same result
    - Ex.: trigger an action by menu or voice command
- Sequential input:
  - User uses one modality to trigger next modality
  - Ex.: Push-to-talk
- Simultaneous input:
  - Concurrent:
    - User issues different commands
    - Ex.: point-to-fly while requesting info using voice command
  - Complementary:
    - Combined info from different modalities gives one command
    - Ex.: intersect object by ray while saying "select"
  - Redundant:
    - Different modalities convey same info
    - Ex.: intersect cube with virtual hand while saying "grab cube"

1. **Visibility of system status**: always keep user informed
  2. Users should always have **full control and freedom**: e.g., always provide an emergency exit, undo, and redo
  3. **Recognition rather than recall**: users should not have to remember information from one part of the dialogue to another  
→ always make actions and options visible
  4. Cater to both novice and expert users, e.g., by accelerators that are unobtrusive to the novice user
  5. **Aesthetic and minimalist design**: don't show information that is irrelevant or rarely needed (it competes with the relevant info)
- ...

# A Taxonomy of Object Interaction Techniques



Mathias Moehring: Realistic Interaction with Virtual Objects Within Arm's Reach



Doug Bowman, JVRC'12

- Higher (interaction) fidelity often results in higher effectiveness
- Increasing fidelity does not always improve user effectiveness within a virtual environment (it does not decrease it either)
- Very few cases where higher fidelity is detrimental
  - Travel techniques are one strong case for *less* fidelity
- Best cases for high fidelity:
  - Difficult and complex visuo-spatial tasks
  - Learning / training
  - High-DOF interaction tasks

- Idea: instantiate virtual/abstract interaction metaphors (handles, icons, sliders, ...) by physical objects

- **Defintion Tangible User Interface (TUI):**

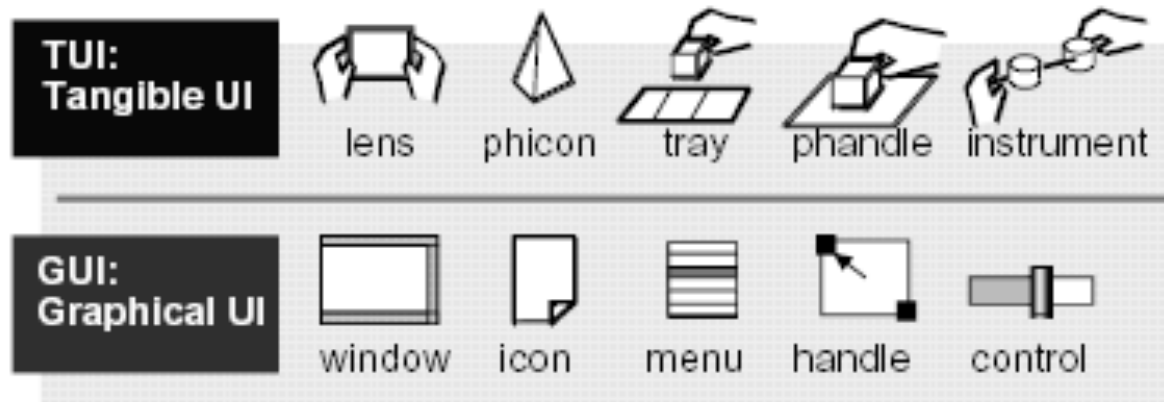
An attempt to give physical form to digital information, making "bits" directly manipulatable and perceptible by people.

Tangible Interfaces will make bits accessible through

- augmented physical surfaces (e.g. walls, desktops, ceilings, windows),
- graspable physical objects (e.g. building blocks, models, instruments),
- ambient physical media (e.g. light, sound, airflow, water-flow, kinetic sculpture).



- Analogies between GUIs and TUIs:



- Examples:

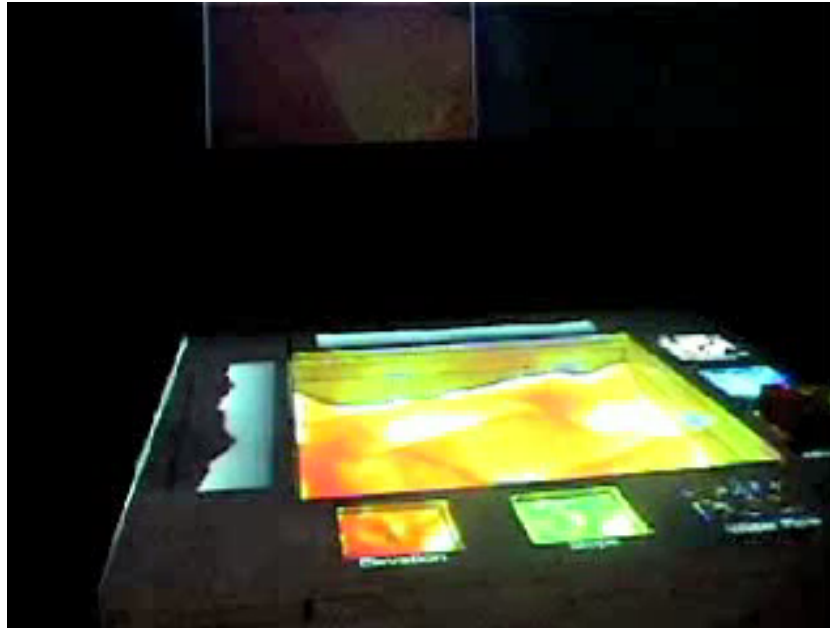


*Tangible Magic Lens*



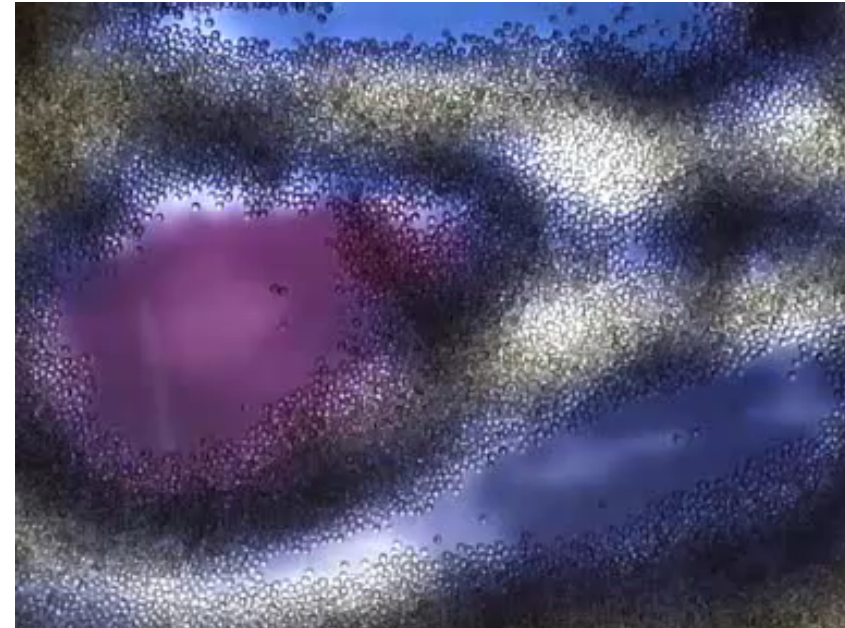
*Tangible Slider*

*Sandscape*  
(sand as terrain)



<http://tangible.media.mit.edu>

*GranulatSynthese*  
(interactive art installation)



[http://imve.informatik.uni-hamburg.de/  
projects/GranulatSynthese](http://imve.informatik.uni-hamburg.de/projects/GranulatSynthese)



*IP Network Design Workbench*  
(use pucks for manipulation  
of nodes and edges)



[http://tangible.media.mit.edu/  
projects/ipnet\\_workbench](http://tangible.media.mit.edu/projects/ipnet_workbench)