

Bremen



Virtual Reality & Physically-Based Simulation Interaction Metaphors

G. Zachmann

University of Bremen, Germany

cgvr.cs.uni-bremen.de



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



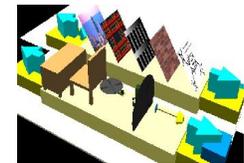
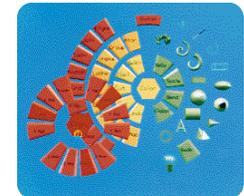
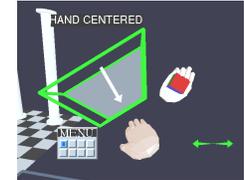
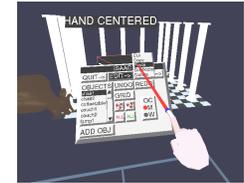
How to interact with Virtual Environments (VEs)?

- *Universal Interaction Tasks (UITs) in VEs* [Bowman]:
 1. Navigation = change 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)
- *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)

Digression: Classification of Widgets for 3D UIs

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

Menu Selection	
Temporary Option Menus	
	<i>Rotary Tool Chooser</i>
	<i>Menu Ball</i>
	<i>Command & 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



The Design of User Interfaces

- 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., motion 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, ...)
 - Pro: very flexible, reconfigurable, "anything goes"
 - Con: can be difficult to use because of lack of force feedback

- Goals (in particular in VR):

- Intuitive / natural interaction (**usability**)
 - By definition: easy to learn
 - Adjust to the users expertise (**expert vs. novice**)
- Efficient interaction (**user performance**)
 - Precision, speed, productivity of the users

*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]

- 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

- Is basically a simple classification problem:
 - Given: a **flex vector** $x \in \mathbb{R}^n$, $n \approx 20$ = joint angles
 - Wanted: gesture $G(x) \in \{ \text{"Fist"}, \text{"Hitch-hike"}, \dots \}$
- Wanted: an algorithm that is ...
 - .. user **in**dependent
 - .. robust (> 99%)
 - .. Fast

An Extremely Simple Gesture Recognition Algorithm

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

- Discretize flex vector

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

0 = flex is "somewhere in the middle"

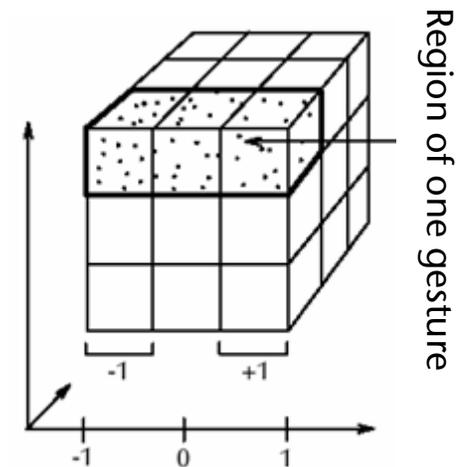
- Gesture = region of d-dimensional parameter cube
- Represent each region/gesture by a discrete vector:

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

- Gesture i is recognized iff

$$f' \cdot g_i = |g_i|$$

- Condition for this to work: regions of different gestures must not overlap

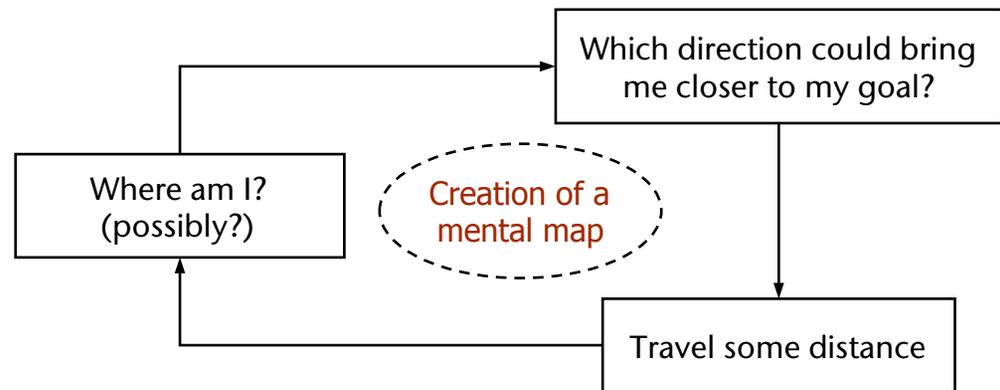


- Implementation details:
 - Do automatic calibration at runtime to fill the range [0,1]:
 - Maintain a running min/max and map it to [0,1]
 - Over time, shrink min/max gradually (robustness against outliers)
 - Ignore transitory gestures
- Dynamic gestures =
 1. Sequence of static gestures (e.g., sign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

- Problem: how should a virtual environment be constructed such that *wayfinding can be trained effectively*?
- How do people find their way:
 - Natural hints/clues
 - Signs (man-made)
- A simple user model for way finding:



- In VEs, there can be 2 kinds of navigation [sic] 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



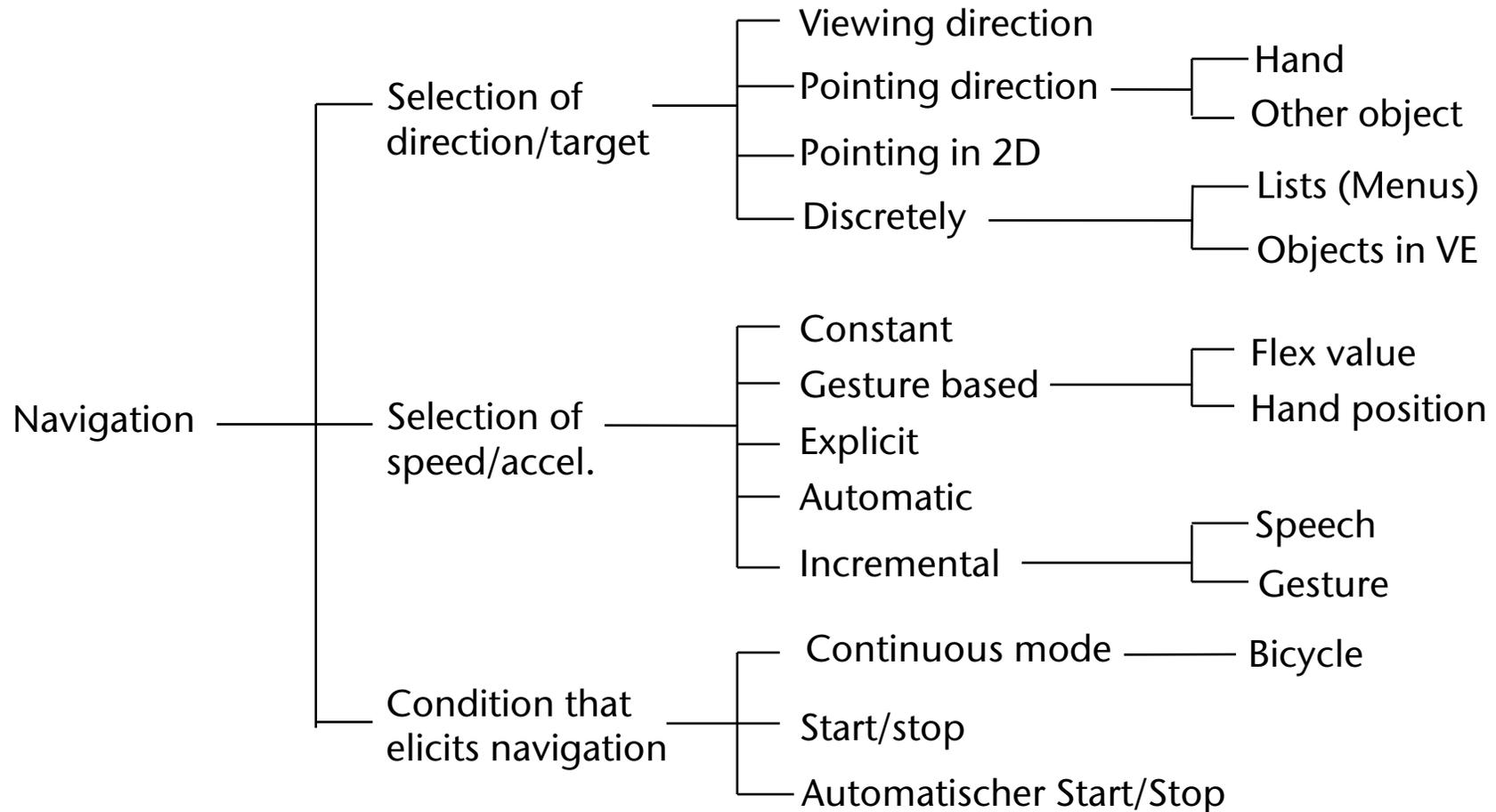
<http://www.spiegel.de/wissenschaft/technik/0,1518,739416,00.html>

Kerstin Schill, Uni Bremen

Techniques for Navigation in VEs

- Real user navigation, e.g., walking, turning head, ...
- *Point-and-fly* (especially in Caves and HMDs)
- *Scene-in-hand*
- *World-in-Miniature*
- Orbital mode
- And some more ...

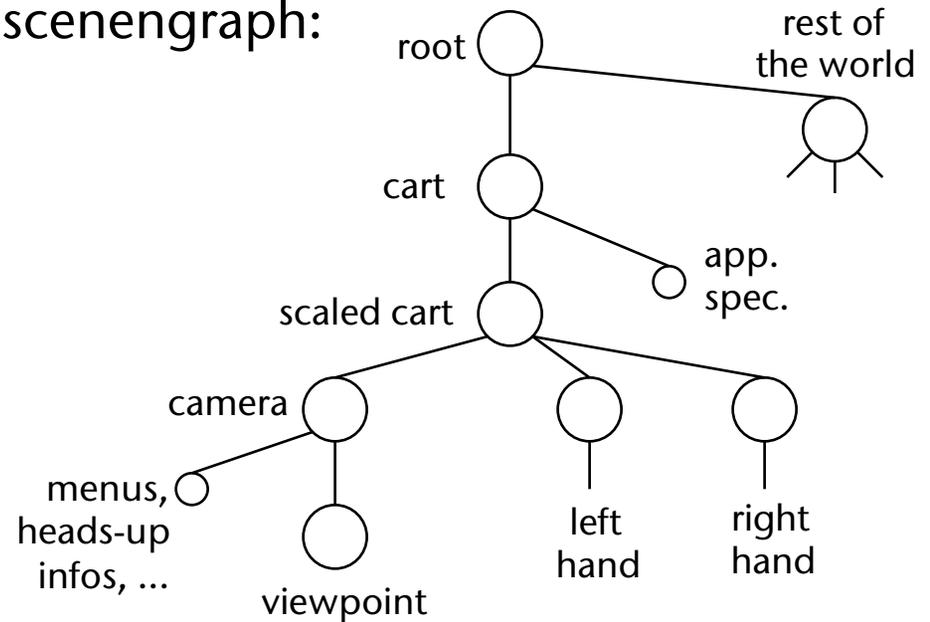
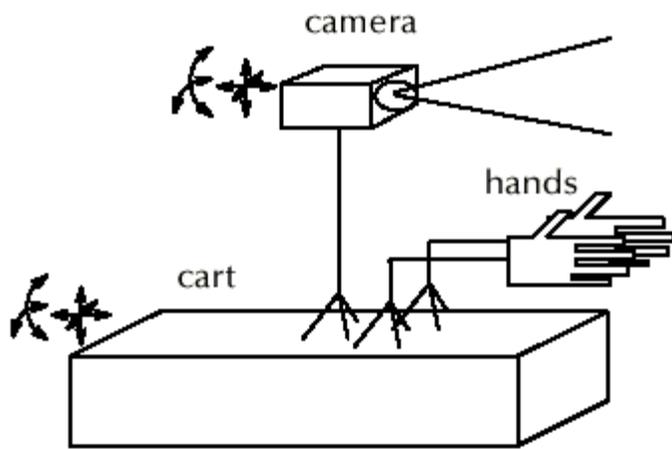
A Taxonomy for this Interaction Task



- Taxonomies are a way to explore (exhaustively, if possible) the *design space*!

Representation of the User

- User = Head, Hand, perhaps whole body (avatar)
- Metaphor "flying carpet":
 - User → camera
 - Camera is placed on a carpet / cart / wagon
- Representation as a (part of) scenengraph:



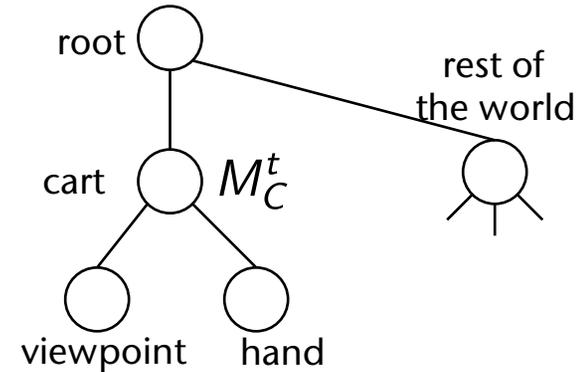
The Point-and-Fly Metaphor

- Kontrollierende Sensoren:
 - Head sensor → viewpoint
 - Hand sensor → moves cart:

$$M_C^t = M_C^{t-1} \cdot s \cdot \mathbf{t}$$

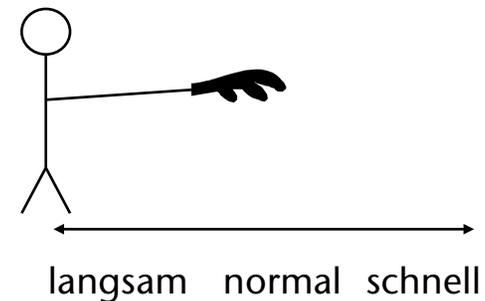
s = speed,

\mathbf{t} = translation vector = 3rd column of hand tracking sensor

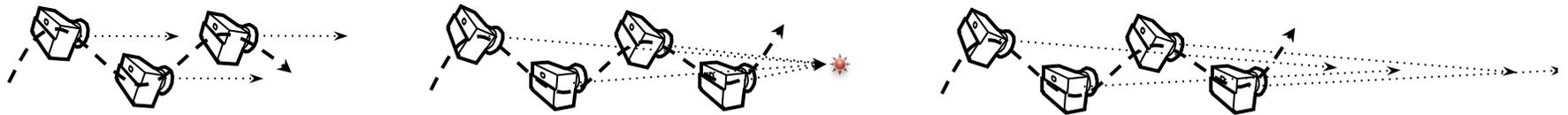


- Generalization: use graphical objects instead of sensor to derive translation direction

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



- 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 it earlier ;-)

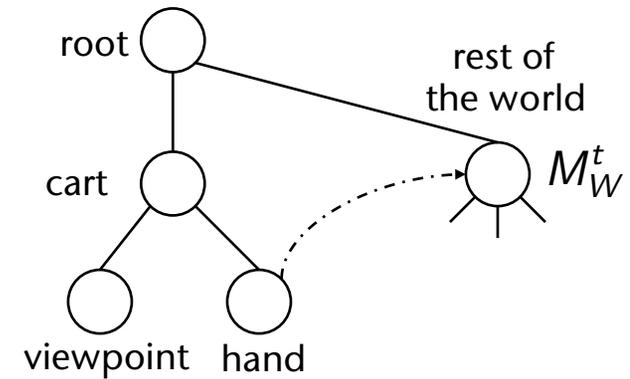


- Resultats:
 - Only vertical oscillation helps increase presence
 - Users prefer slight oscillation over no oscillation
 - Short "travel distances" can be estimated more precisely (~ factor 2)

■ *Scene-in-hand:*

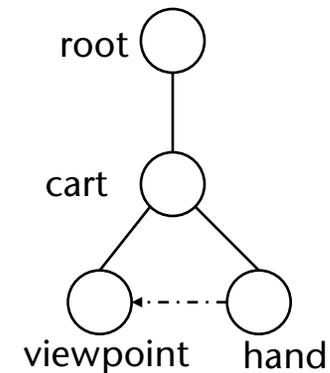
- "Grabbing the air" technique
- Cart remains stationary, scene gets rotated by hand sensor about a specific point in space
- The transformation:

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



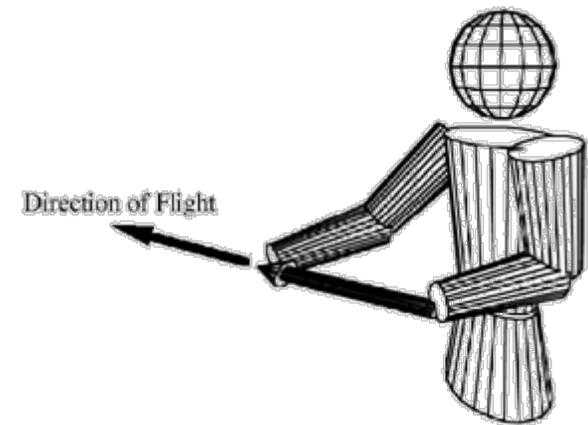
■ *Eyeball-in-hand:*

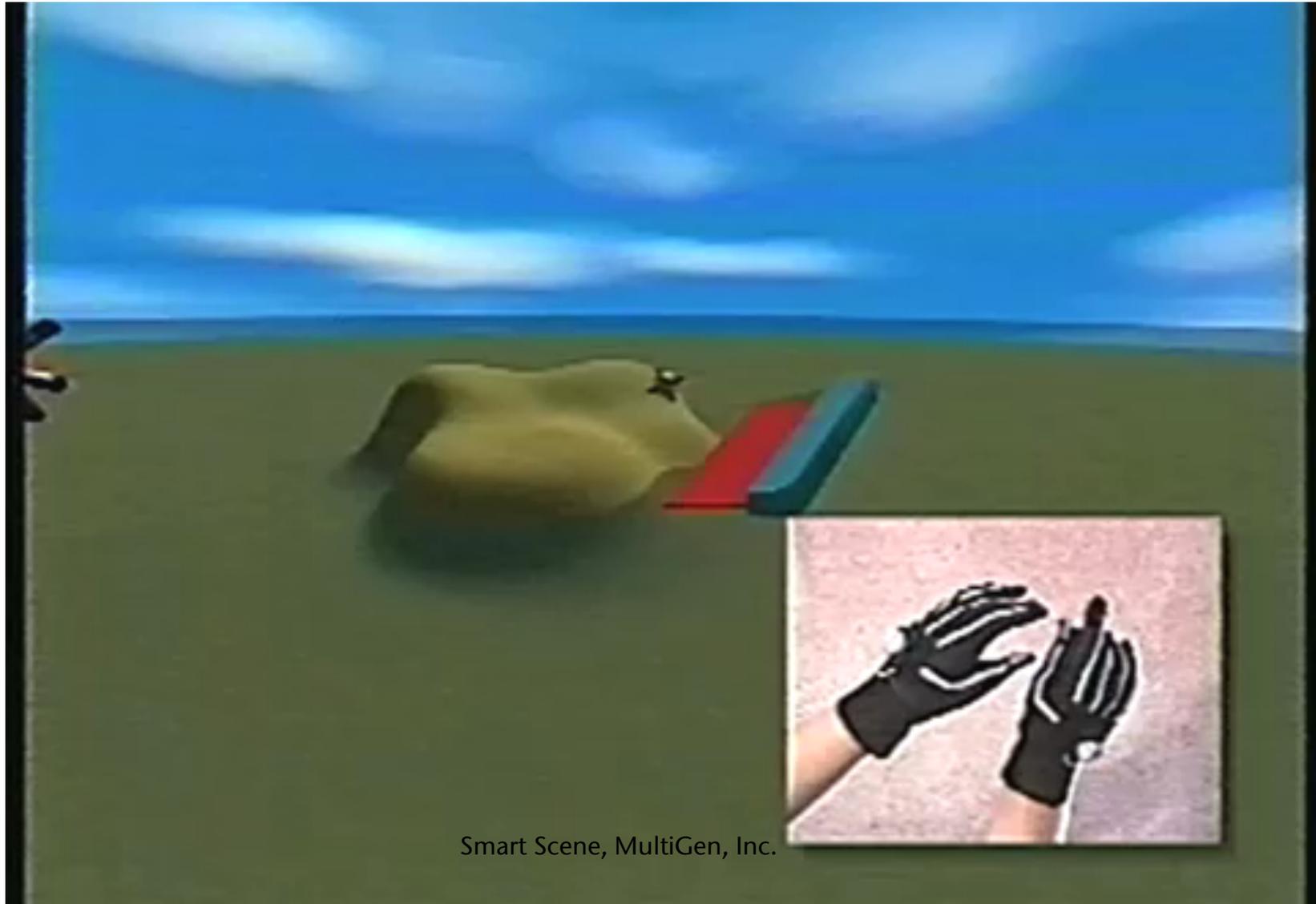
- Viewpoint is controlled directly by hand
- Can be absolute or relative (accumulating) mode



Two-Handed Navigation (with Pinch Gloves)

- Question: how to navigate with both hands?
(increase input bandwidth)
- Idea: only use 2 points and 1-2 triggers (→ pinch gloves)
- Idea: use "*scene-in-hand*"
 - 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





Navigation Without Hands

- 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 trigger 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 = e.g., leaning angle



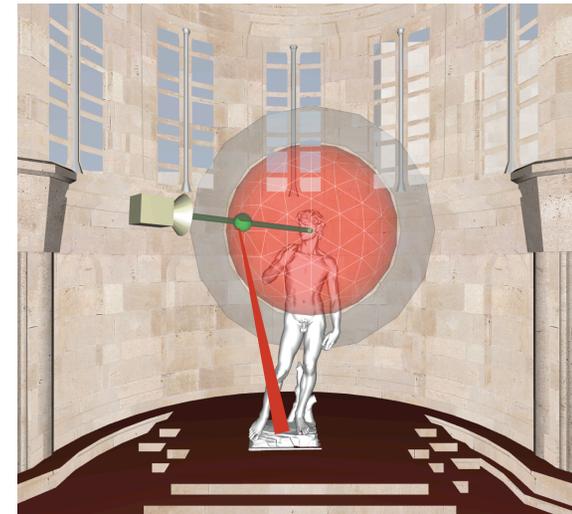
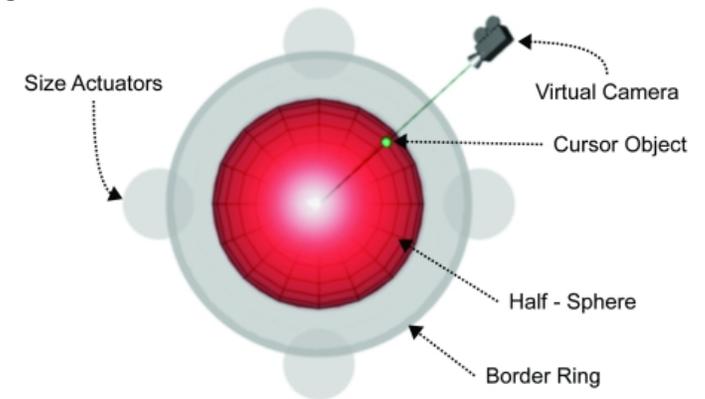
Exploration of VEs using a Magic Mirror

- 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 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 1 class



- Metaphor for defining the viewpoint directly
- Input device: *wand* with wheels and buttons
- **Decomposition** of the task:
 1. Define center of the sphere
 - Will be the new **center of interest (COI)**
 - E.g. by ray casting: shoot ray into scene, intersection point = new COI
 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)
 5. Switch to next phase using a button



Navidget for Immersive Virtual Environments

Sebastian Knödel, Martin Hachet
iparla.labri.fr