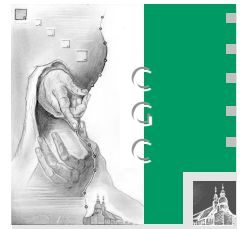


Virtuelle Realität
Interaktionsmetaphern

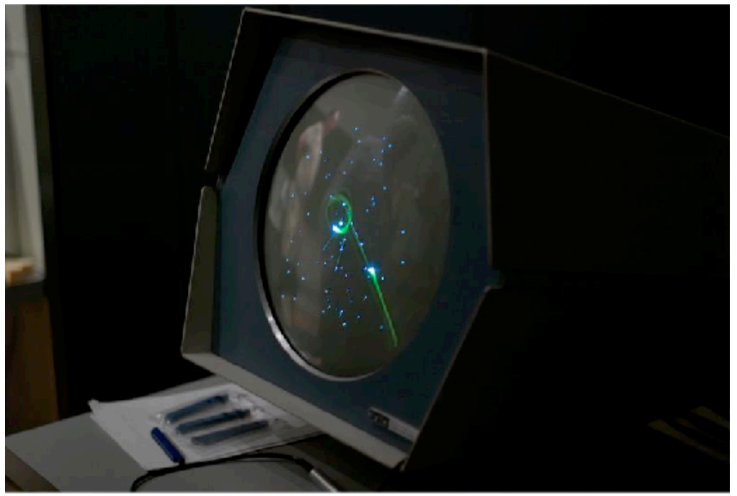
G. Zachmann

Clausthal University, Germany

cg.in.tu-clausthal.de

Historie

- Das erste Computer-Spiel (vermutlich):
  - Spacewars, 1961, MIT
  - Damit auch die ersten Interaktionsgeräte und -metaphern
  - Zwei Spieler, zwei Spaceships ("wedge" und "needle"), feuern Tropedos



Wie interagiert man mit VEs?

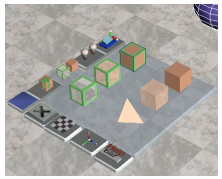
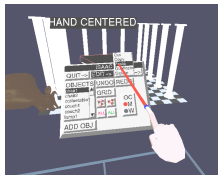
- Grundlegende Aufgaben (= *Universal Interaction Tasks* [Bowman]):
  - Navigation (Viewpoint ändern)
  - Selektion
  - Objekte greifen, bewegen, manipulieren
  - Geometrie modellieren und modifizieren (selten)
  - *System control* (Menüs, *Widgets*, *Slider*, Zahlen eingeben, etc.)
- Elementare Interaktionsbausteine (*BITs = basic interaction tasks* [Foley /]):
  - Selektion (Objekte, Menüs, ..)
  - Positionierung (inkl. Orientierung) oder Manipulation
  - Quantifizierung
  - Texteingabe, Spracheingabe

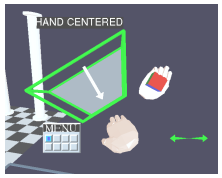
- Zwei grobe Richtungen:
  - Natürliche Interaktion
    - Versuche, die Realität und die Interaktion damit möglichst genau abzubilden
  - "Magische" Interaktion
    - Gib den Usern neue Möglichkeiten
    - Herausforderung dabei: den kognitiven Overhead dabei so klein wie möglich zu halten, so dass der User nicht von seiner Aufgabe abgelenkt wird!
- Hilfsmittel:
  - Direkte *User-Aktion* (Körperbewegung, Geste, ..)
    - Gut wenn intuitiv, Möglichkeiten beschränkt
  - Physikalische Geräte (z.B. Taste, Lenkrad)
    - Haptisches Feedback für präzisere Kontrolle
    - Evtl. schwer zu (er-)finden
  - Virtuelle Geräte (z.B. Menü, "*anything goes*")
    - Flexibel, rekonfigurierbar
    - Nicht leicht/präzise zu bedienen

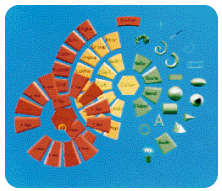
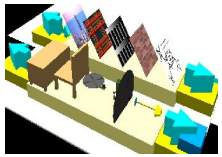
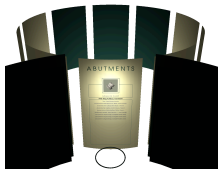
Classification of 3D-Widgets

- **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 Valuators**
  - **Continuous Valuators**
  - **Special Value Input**
  - **Menu Selection**
  - **Containers**

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




- Ziele:
  - Intuitive / natürliche Interaktion (usability)
    - Leicht zu erlernen
    - Passt sich dem Benutzer an (expert vs. novice)
  - Effiziente Interaktion (user performance)
    - Genauigkeit, Geschwindigkeit, Produktivität des Users
- Probleme:
  - Keine Constraints
  - Insbesondere: fehlendes haptisches Feedback
  - Effiziente Interaktion mit Objekten außerhalb der Reichweite
  - Tracker-Rauschen / -Ungenauigkeit
  - Ermüdung
  - Fehlende 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.
[Bran Ferran]

Gestenerkennung

- Klassifikationsproblem:
  - Gegeben: Flex-Vektor $x \in \mathbb{R}^n, \quad n \approx 20$
  - Gesucht: Geste $G(x) \in \{\text{“Faust“, “Hitch-hike“ ,...}\}$
- Gesucht: Algo, der ..
  - .. benutzer**un**abhängig ist
  - .. robust ist (> 99%)
  - .. schnell ist
  - .. möglichst nur 1x trainiert werden muss (besser: 0x)

Einfache Gestenerkennung

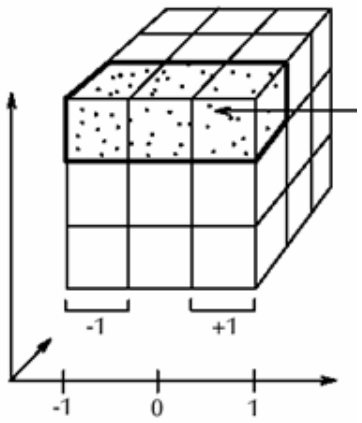
- NN gut, falls viele Gesten, oder Flex-Werte im "Inneren"
- Falls wenige Gesten und alle "am Rand":
  - Diskretisiere Flex-Vektor $f \in [0,1]^d \rightarrow f' \in \{-1,0,+1\}^d$

    0 = Flex-Wert ist weder nah bei 0, noch nah bei 1
  - Bilde Randregionen im d-dimensionalen diskreten Würfel $\{-1,0,+1\}^d$
  - Wähle für jede Randregion (= Geste) einen Repräsentanten $g \in \{-1,0,+1\}^d$

    0 = *don't care*
  - Geste *i* ist erkannt, wenn

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

  - Bedingung: die Regionen der verschiedenen Gesten dürfen nicht überlappen



Bereich einer Geste

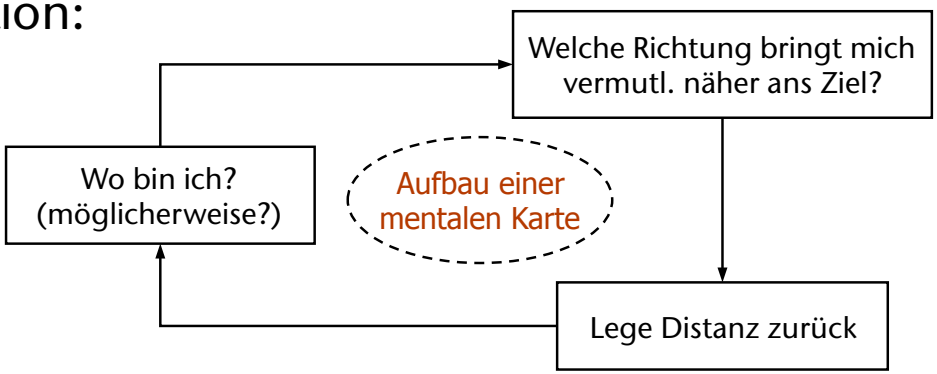
- Implementierungsdetails:
  - Automatische Nachkalibrierung auf [0,1]:
    - Min/Max mitführen und auf [0,1] mappen
    - Min/Max langsam schrumpfen
  - Transitorische Gesten ignorieren
- Dynamische Gesten:
  - Folgen von statischen Gesten (Zeichensprache)
  - Pfad eines Fingers / des Handrückens
  - Nutzen?

Navigation

- *Wayfinding* & *Locomotion*
- *Locomotion / Travel*:
  - Distanz überwinden
  - Maneuvrieren (= Viewpoint setzen, inkl. Orientierung)
  - Technik
- *Wayfinding*:
  - Strategie
  - Wissen

Wayfinding als Aufgabe

- Wie muss die virtuelle Umgebung aussehen, damit *Wayfinding* effektiv trainiert werden kann?
- Hinweise in der Umgebung für *Wayfinding*:
  - Natürliche Hinweise
  - Wegweiser
- User-Modell für Navigation:

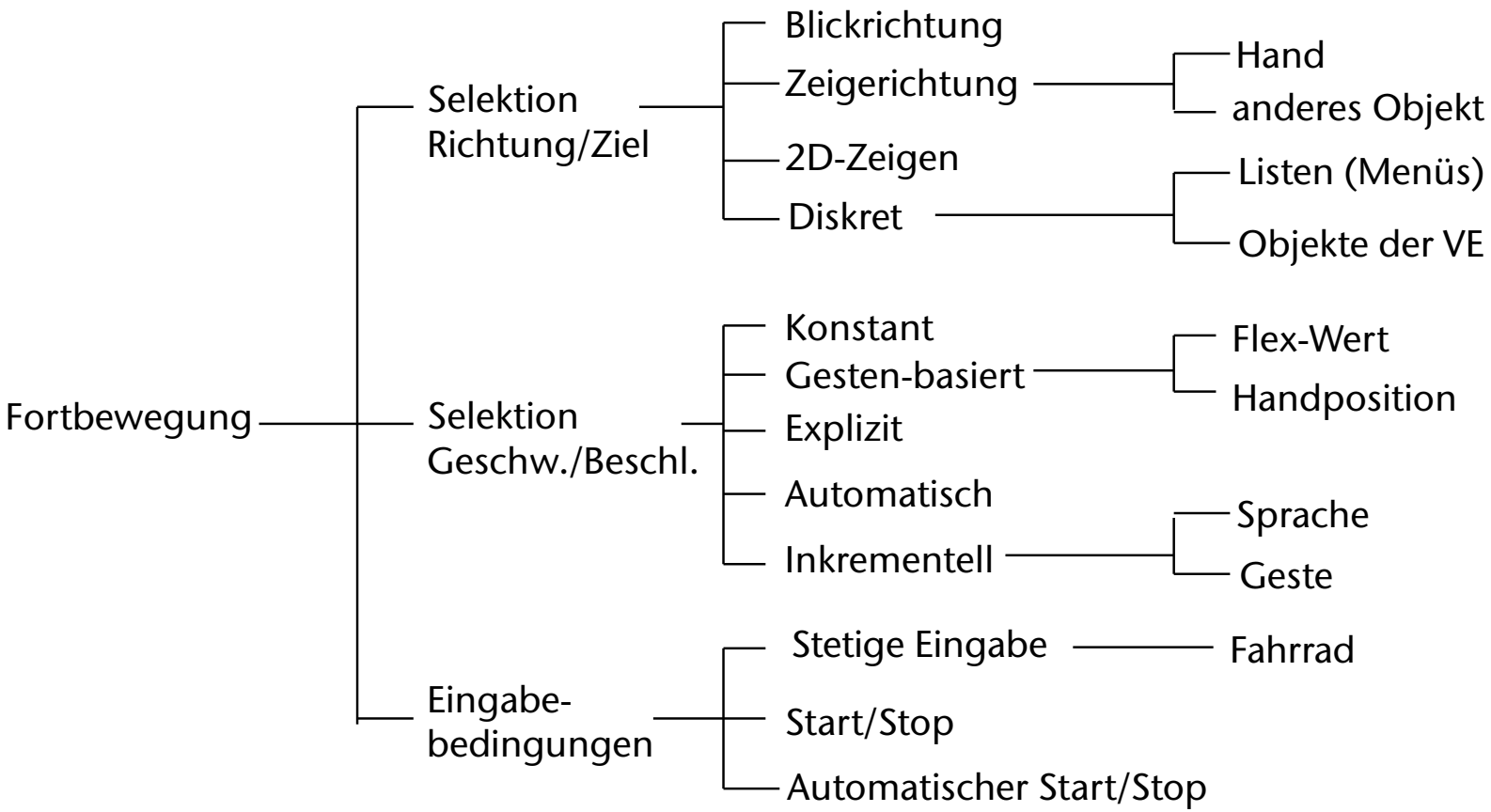


- Navigationshilfsmittel:
  - Steigerung der Performance des Users in der virtuellen Umgebung
  - Steigerung in der realen Welt (= Steigerung des Trainingseffekts)

Navigationstechniken

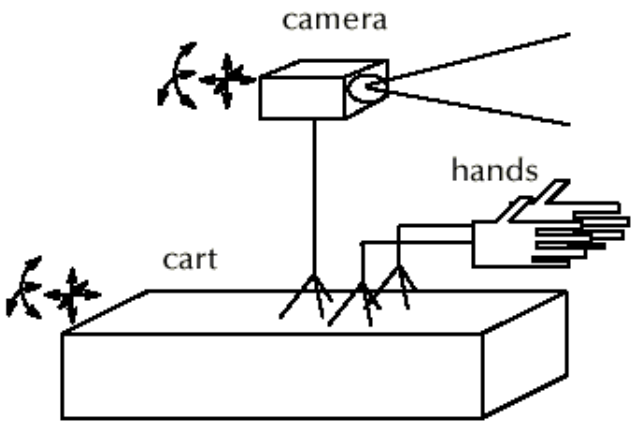
- Real: Laufen, Kopf bewegen
- *Point-and-fly* (*Cave*, HMD)
- In Blickrichtung (Boom)
- Fadenkreuz
- *Scene-in-hand*
- *World-in-Miniature*
- Orbital mode
- Richtige Art hängt stark von der Applikation ab!

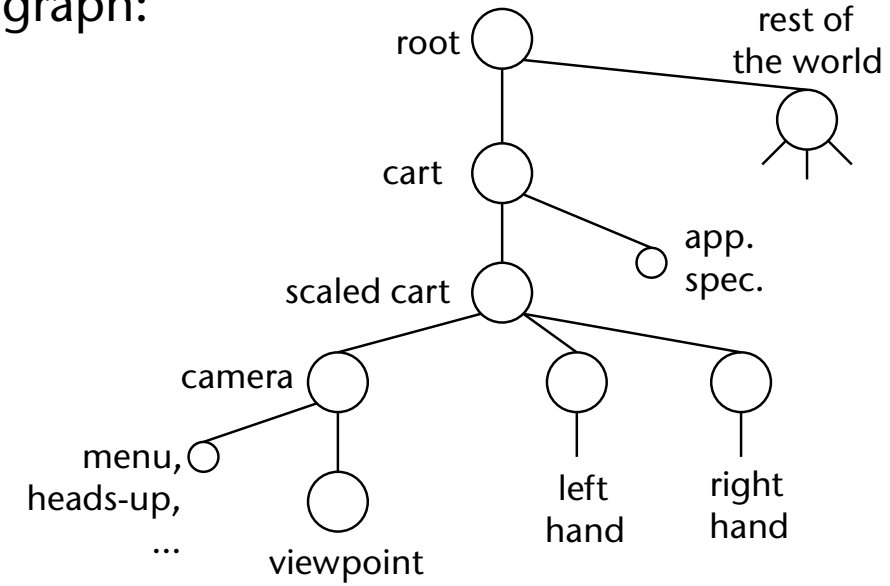
Eine Taxonomie



Repräsentation des Users

- User = Kopf, Hand,
  evtl. ganzer Körper (Avatar)
- Metapher "fliegender Teppich":
  - User → Kamera
  - Kamera steht auf Wagen (Teppich)
- Abbildung in (Teil-)Szenengraph:

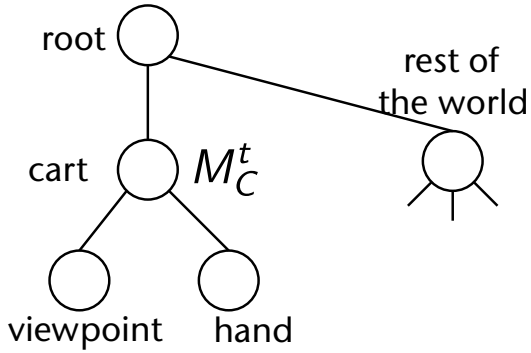




Point-and-Fly

- Kontrollierende Sensoren:
  - Kopfsensor → *Camera*
  - Handsensor → *Cart*

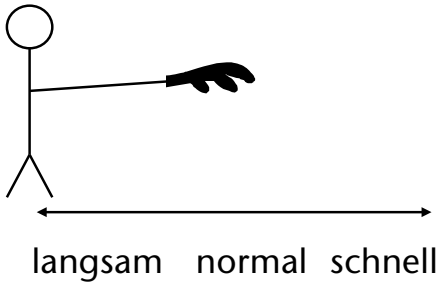
$$M_C^t = M_C^{t-1} \cdot v \cdot \underbrace{T(M_H^z)}$$



Translation, die aus der 3. Spalte der Rot.matrix des Handsensors erzeugt wird

- Verallgemeinerung:
  Graphische Objekte statt Sensoren

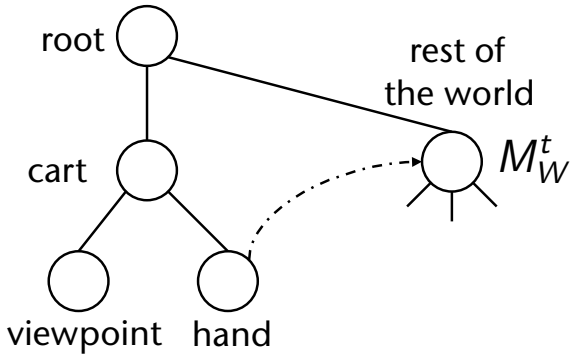
- Spezifikation der Geschwindigkeit:
  - Konstant (z.B. Boom)
  - Daumenkrümmung
  - Abhängig von Entfernung Hand – Brust
  - Manchmal unabhängig von *Framerate*



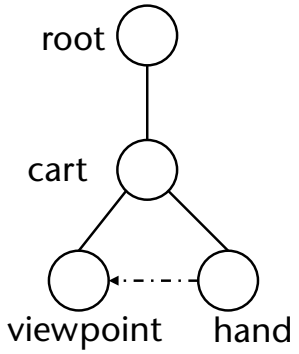
Scene-in-hand, Eyeball-in-hand

- *Scene-in-hand*:
  - *"grabbing the air" technique*
  - *Cart* bleiben stehen, Szene wird rotiert durch Handsensor
  - Die Transformation:

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

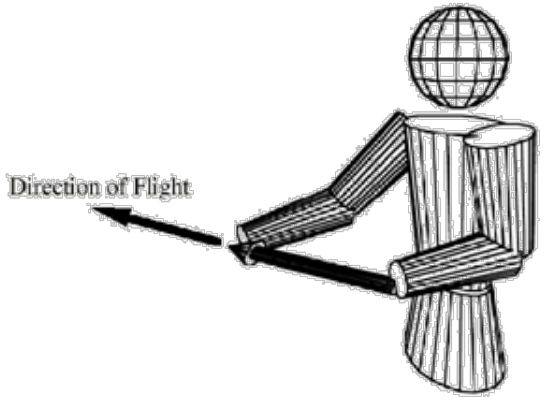


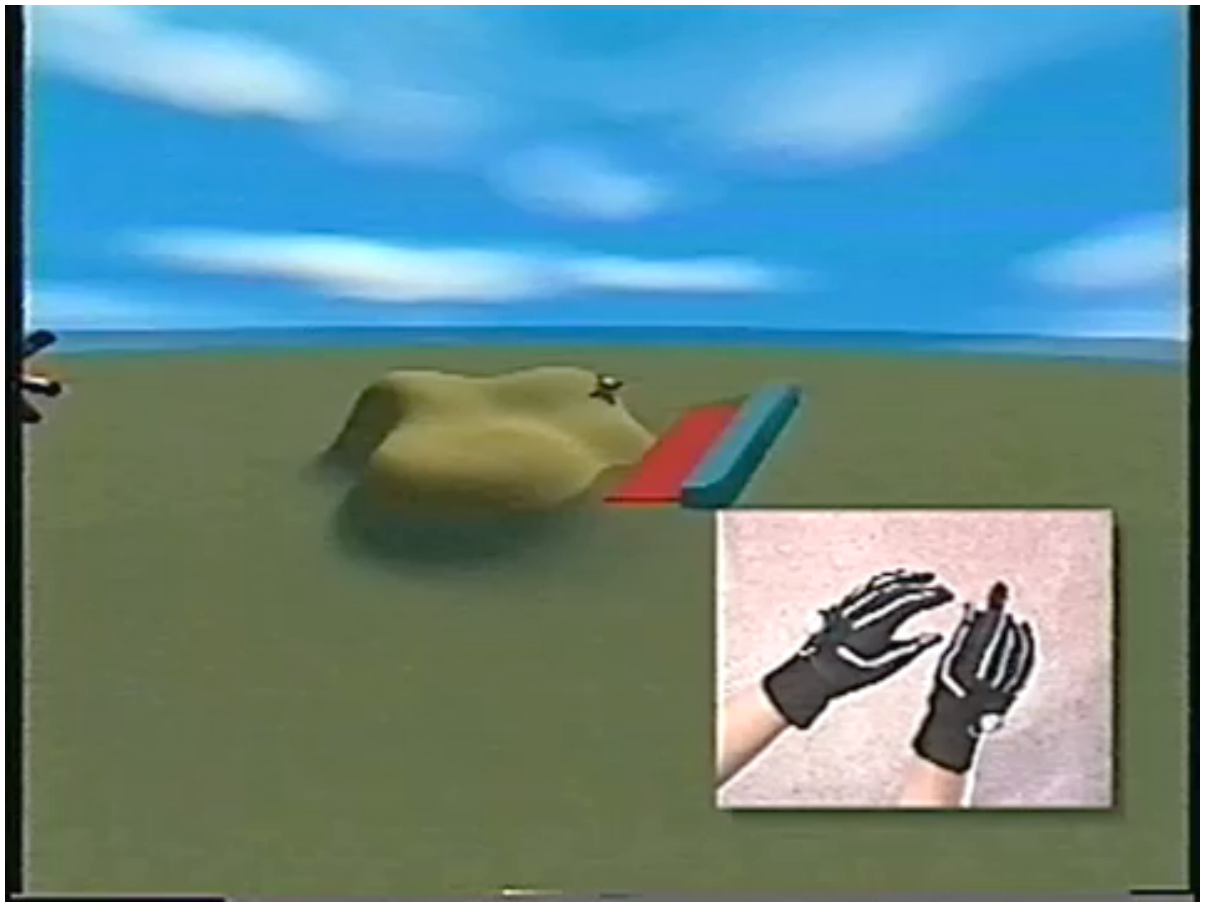
- *Eyeball-in-hand*:
  - Viewpoint wird direkt durch Hand gesteuert
  - Absolut oder relativ (akkumulierend)



Beidhändige Navigation (mit *Pinch Gloves*)

- Fragestellung: wie kann man mit 2 Punkten + 1–2 Trigger navigieren?
- Idee: "*scene-in-hand*"
  - 1 Trigger, 1 Punkt bewegt → Translation der Szene
  - 2 Trigger, 1 Punkt fest, 1 Punkt bewegt → Rotation der Szene
  - 2 Trigger, 2 Punkte bewegt → Skalierung der Szene
- Hat sich trotzdem nicht durchgesetzt
  (vermutlich, weil *Pinch Gloves* sich nicht durchgesetzt haben)
- Zwei-händige Navigation (Variation):
  - Vektor zwischen Händen = Richtung
  - Länge des Vektors = Geschwindigkeit





Smart Scene, MultiGen, Inc.

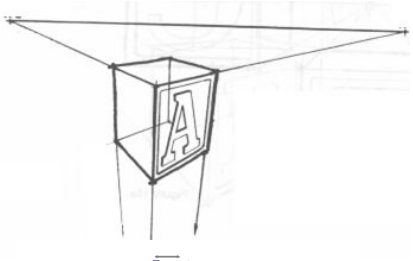
Navigation ohne Hände

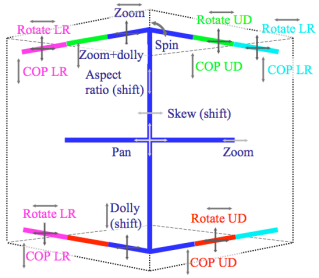
- Idee: projiziere VE verkleinert auf Boden und verwende Füße
- Grobe Navigation:
  Teleportation → User läuft zu Punkt auf Karte und triggert
- Systemkommandos:
  1. Karte einschalten = Blick zu Boden + Trigger
  2. Teleportation = Blick zu Boden + Trigger
  3. Karte ausschalten = Blick nach oben + Trigger
     - Trigger = Sprache oder "Fußgeste"

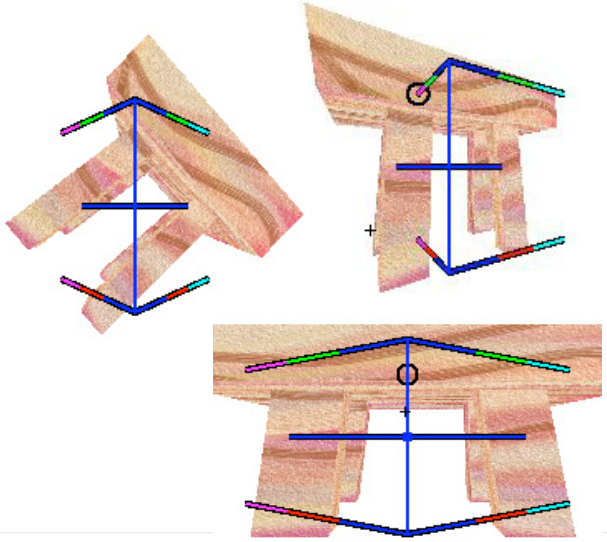
- Feine Navigation: in gewünschte Richtung "lehnen";
  Geschwindigkeit häbgt ab von
  - Neigungswinkel
  - Abstand vom Rand der Cave

Exkurs: das IBar — eine intuitive 2D-Metapher

- Aufgabe: intuitive Metapher zur Manipulation der perspektivischen Projektion
- Beobachtung: Zeichner konstruieren die Projektion durch Fluchtpunkte
- Idee:
  - Manipuliere diese Fluchtpunkte
  - Verwende dazu das Bild der Kanten eines Würfels
- Durch Manipulieren der "Handles" kann man verschiedene Parameter modifizieren:
  - Orientierung, Zoom, Pan, Proj.zentrum

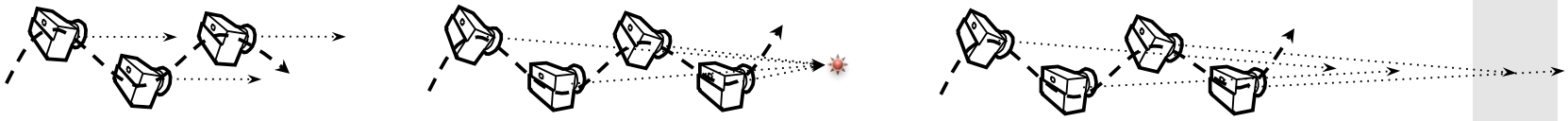






Wahrnehmung der zurückgelegten Distanz in VR [2009]

- Frage: wie gut ist die Präsenz bei Navigation in VR?
- Idee:
  - Oszillation des Viewpoints wie in der Realität nachbilden
  - (Haben die first-person-shooter schon viel früher entdeckt ;-))



- Resultate:
  - Nur eine Oszillation entlang der Hochachse bringt etwas
  - User ziehen leichte Oszillation der Navigation ohne vor
  - Kurze "Reisedistanzen" werden genauer eingeschätzt (ca. Faktor 2)

Exploration von Szenen: der Magic Mirror

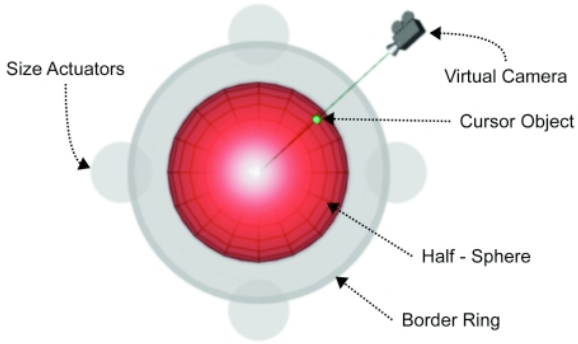
- Aufgabenstellung: zweiten Viewpoint (Bild im Bild) intuitiv verständlich in eine Szene integrieren und manipulierbar machen
- Idee: der Spiegel → "magic mirror"
  - Ein Objekt in der VE dient als Handspiegel
  - Wird immer relativ zur Kamera positioniert (wandert mit)
  - Kann man manipulieren wie jedes andere Objekt auch
- Zusatz-Features (nicht bei realen Spiegeln):
  - Zoomen
  - Vergrößern / verkleinern des Spiegels
  - Clippen von Objekten vor dem Spiegel, die die Sicht versperren
  - "Richtig-herum-Drehen" der Szene im Spiegel
  - Positionieren des Haupt-Viewpoints an der Stelle des gespiegelten VP

- Beispiele:

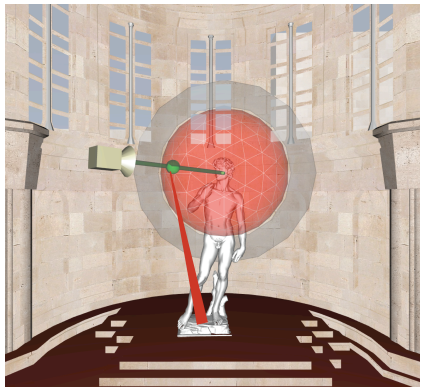
- Implementierung:
  - 2x rendern
  - Erstes Mal nur einen kleinen Viewport mit dem gespiegelten Viewpoint
  - Abspeichern als Textur
  - Zweites Mal main view rendern, ohne Spiegel
  - Dann Spiegel-Objekt drüber rendern ohne Z-Test

Immersive Navidget [2008]

- Metapher zur Definition eines Viewpoints
- Eingabegerät: *Wand* mit Rädchen und Buttons
- Dekomposition:



1. Mittelpunkt einer Kugel festlegen
   - Wird der neue Center of Interest (COI)
   - Z.B. durch Ray-Casting: Schnittpunkt mit Szene = Mittelpunkt
2. Radius der Kugel festlegen = Abstand zum COI
   - Hier mit dem Rädchen am Wand
3. Viewpoint auf der Kugel bestimmen
4. Animation des Viewpoints auf einem Pfad zum neuen Viewpoint
5. Wechsel zwischen den einzelnen Phasen mit dem Button




Navidget for Immersive
Virtual Environments
Sebastian Knödel, Martin Hachet
iparla.labri.fr