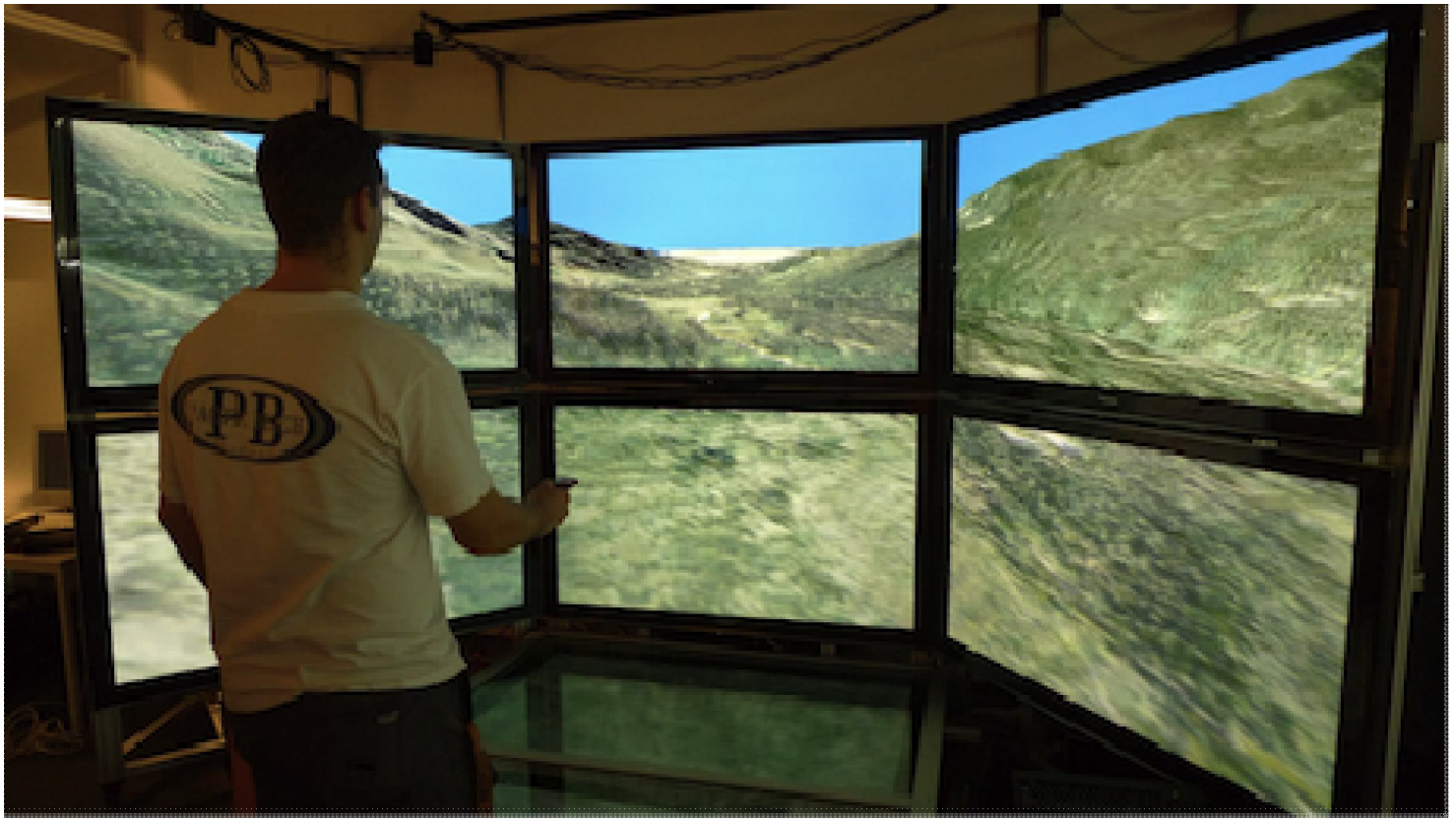
Selection and Zooming using Android Phone in a 3D Virtual Reality



Yanko Sabev

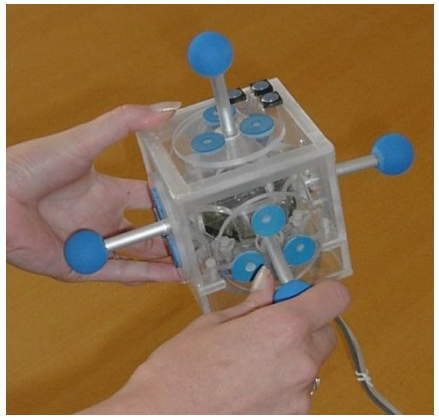
Director: Prof. Gudrun Klinker (Ph.D.)

Supervisors: Amal Benzina

Technische Universität München

Introduction

- Human-Computer Interaction through time



The Cubic SpaceMouse and the YoYo
from Froehlichs “The Quest for Intuitive 3D-Input Devices”

5DT Data Glove Ultra

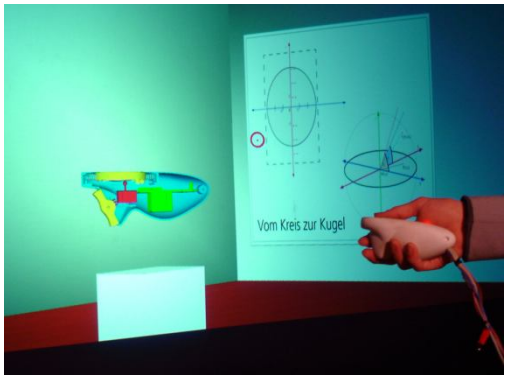
Apple iPhone 4

Kiliks et al. “two-4-six”

Buffalos GyroMouse

Most Relevant Works

- „Phone Based Motion Control Travel Technique in VR“ – FAR 2010
  - Android phone for traveling in a VR
  - Uses orientation sensor and touch screen
- „two – 4 – six“ -Kilik, Blach, Fröhlich, 2006
  - Handheld device for 3D-Presentations
  - Uses Gyroscope and Touch pad



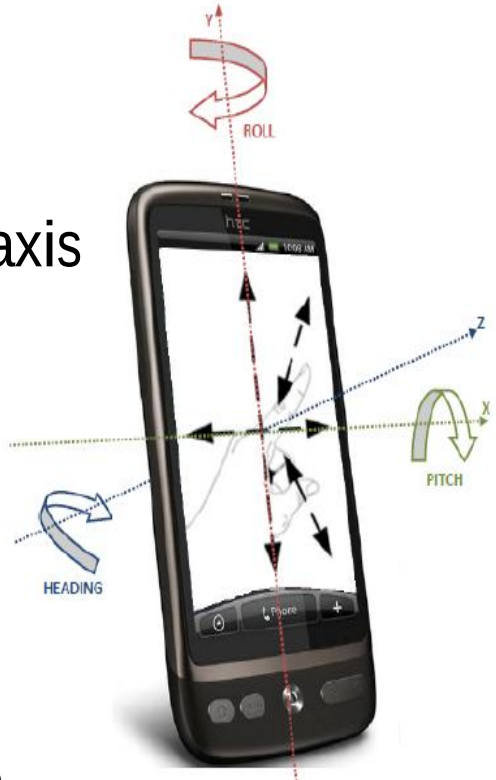
Selection and Zooming with Mobile Phone

- Input Device:
  - Android phone to control a 3D cursor
- Used integrated sensors:
  - Orientation sensor
  - Gyroscope
  - Touch screen

Android Sensors

- **Orientation sensor**
  - Software implementation; uses Accelerometer data
  - rotation angles around the 3 axes in degrees
  - Used value – Pitch, positive values when the z-axis moves **toward** the y-axis

- **Gyroscope**
  - rate of rotation around the 3 axes in radians/second
  - Used value – angular speed around the z-axis, positive in the counter-clockwise direction



"Benzina et al. 2011"

Metaphors

- Mapping Phone -> VE
  - Orientation Metaphor:
    - Heading -> Horizontal
    - Pitch -> Vertical
    - Touch along Y -> Depth
  - Touch Metaphor:
    - Touch along X -> Horizontal
    - Touch along Y -> Vertical
    - Pitch -> Depth

Android data - Pitch

- Filters:
  - Average filters to smooth the data
  - Pitch: last three values
  - Heading: last five values
- Pitch (Vertical coordinate):
  - Calculate the difference between the current and the previous value and subtract it from the current vertical coordinate

Android data – Heading

- Heading (Horizontal coordinate):
  - Calculate the time difference between value changes and turn it from nanoseconds to seconds
  - Multiply the sensor value with the time difference
  - Substract the averaged new value from the current horizontal coordinate
- Sending data:
  - At every sensor change
  - At every finger move on the touch screen

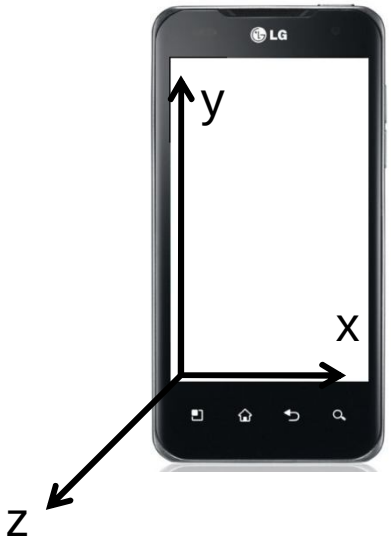
Android data – Touch screen

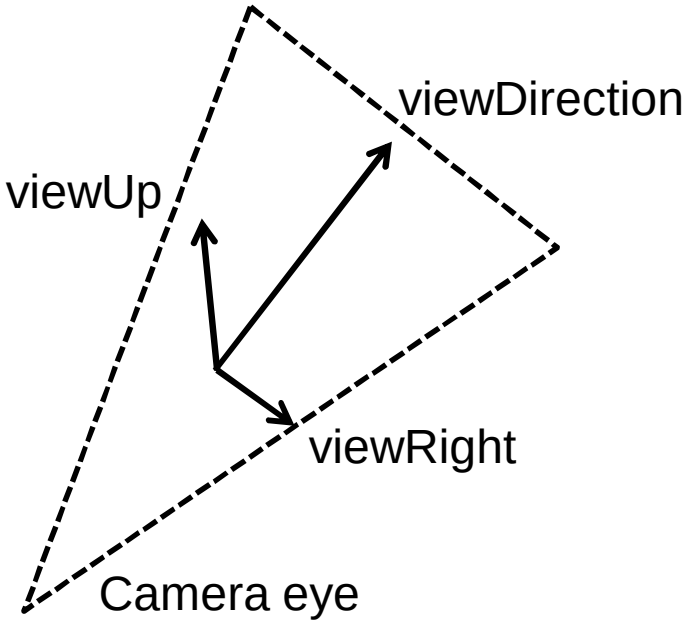
- Precondition for sensor listening and data sending: finger touch

- Finger displacement:
  - On finger down set the start point and assign the current horizontal, vertical and depth values to the touch values
  - On finger move calculate the displacement from the start point and add it to the start values assigned at finger down

Virtual World Camera coordinate system (CS)

- Mapping Android CS to Camera CS:
  - 1:1 mapping:
    - X -> viewRight
    - Y -> viewUp
    - Z ->viewDirection





Cursors coordinates in the Virtual World

- Initial position:
  - 500m in front of the camera
- Coordinates calculation:
  - Add to the start position the product of the android coordinates and the accordant camera view vectors

```
cursorPosition.x() = cursorStartPosition.x() +
        cursorX*m_vViewRight.x() + cursorY*m_vViewUp.x() + cursorZ*m_vViewDirection.x();
cursorPosition.y() = cursorStartPosition.y() +
        cursorX*m_vViewRight.y() + cursorY*m_vViewUp.y() + cursorZ*m_vViewDirection.y();
cursorPosition.z() = cursorStartPosition.z() +
        cursorX*m_vViewRight.z() + cursorY*m_vViewUp.z() + cursorZ*m_vViewDirection.z();
```

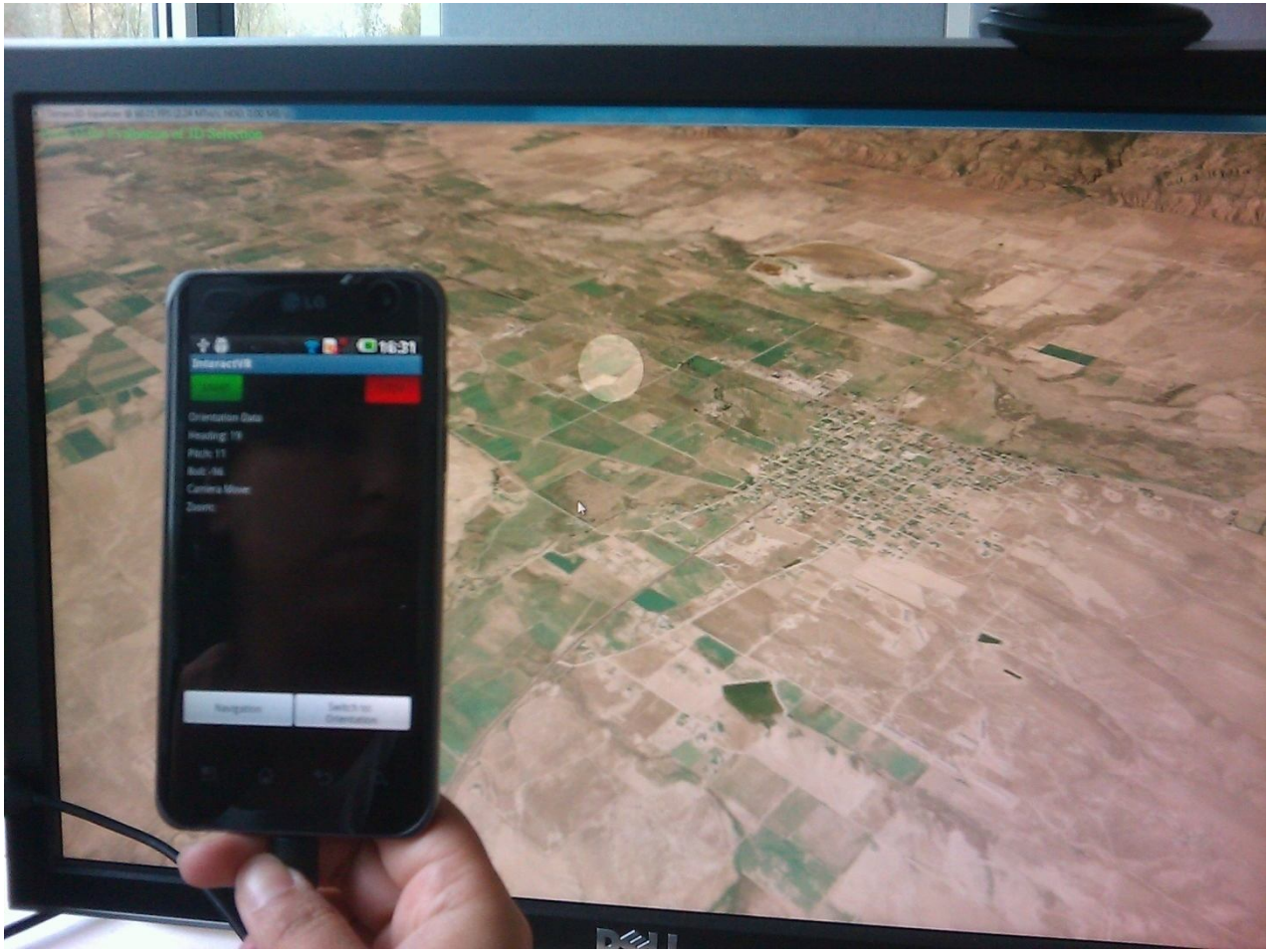
Position and Rate Control

- Position control:
  - Below a threshold
  - Android sensors/touch values are mapped to cursor coordinates
- Rate control:
  - Switch above a threshold
  - Android sensors/touch values rate change are mapped to the cursor velocity

Selection mode

- Select/Deselect:
  - double tab on the touch screen
  - Color change of the cursor as visual feedback
- Bring Point:
  - Ray direction calculation: Camera to Cursor
  - Move – using the Pitch of the Android device to change the camera position
- Zoom – using the Touch Screen of the Android device:
  - Finger slide on the touch screen is mapped to a scale factor for the frustums clipping planes

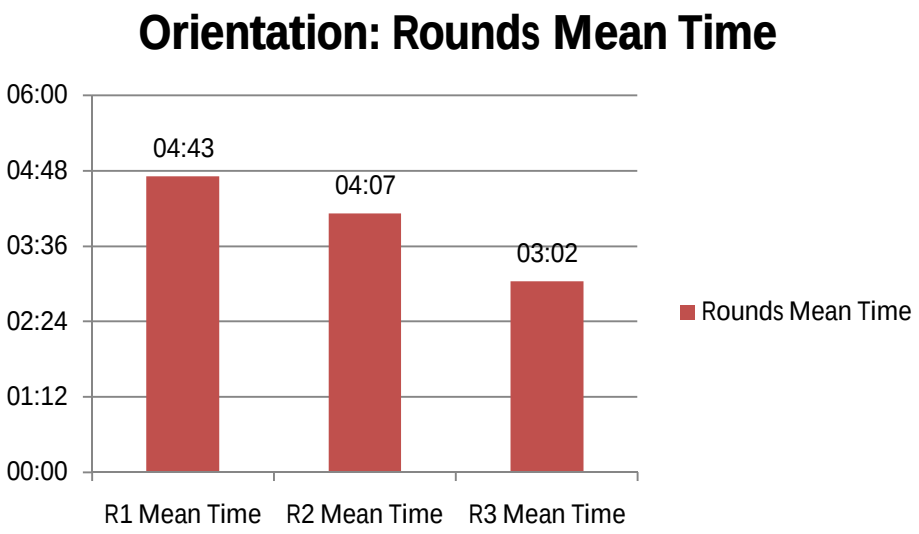
UI and Cursor Design

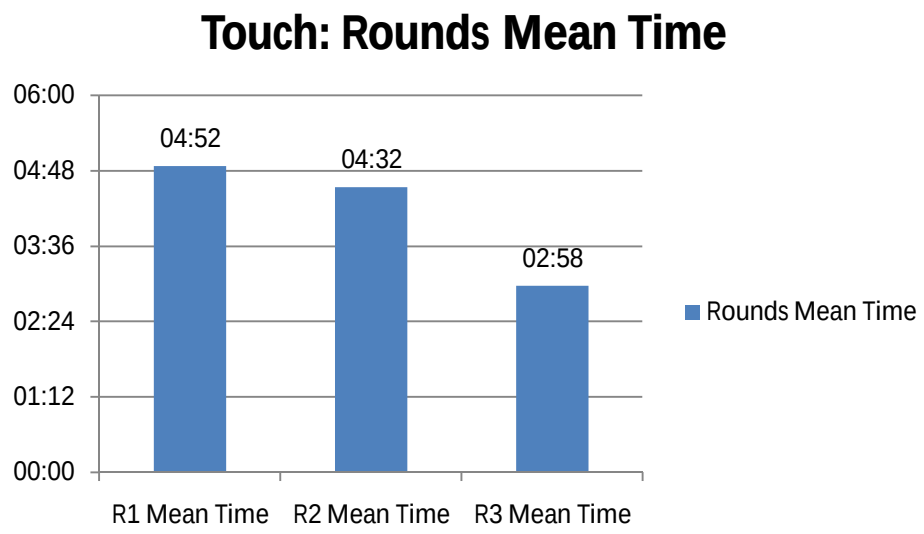


Evaluation

- Hardware:
  - 50‘‘ 3D Monitor, 3D Glasses, LG Optimus 2X P990
- Virtual Environment:
  - Terrain3D
- Test scenario:
  - Use only one metaphor
  - 10 spheres with randomized appereance and constant position
  - color change for intersection feedback
  - score points depending on the distance to the sphere center
  - Time counter
  - 3 rounds
- Test Users :
  - 5 for each metaphor (=10), Age: 18-30, different gender

Test results

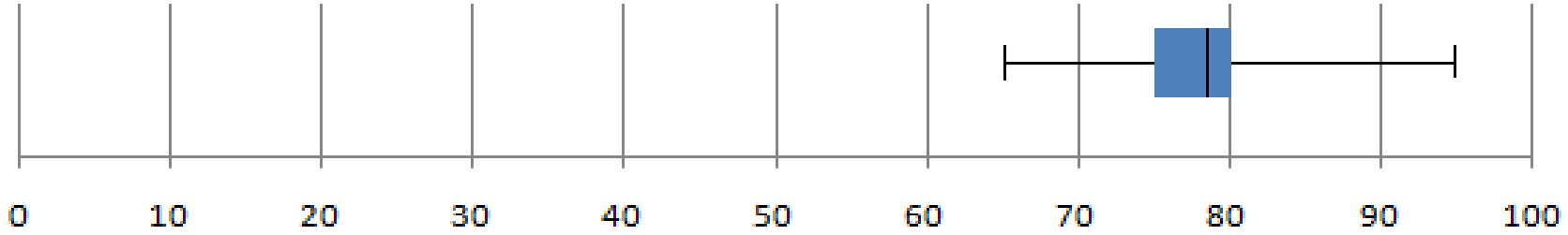




SUS

- System Usability Scale:
  - Orientation (70,5)

  - Touch (78,5)



Comments

- Overall comments:
  - Occasional lack of smoothnes in the O-Metaphor
  - Unintuitive depth movement towards the center
  - Cursors depth not clear enough
- „I think it‘s easy to use. But I can imagine for those people who don‘t have a lot [...]knowledge about 3D can be difficult. (But I guess that‘s not the target group)...“ (Touch metaphor user)

Conclusion

- People liked the Touch metaphor better and found it more easy to use and learn
- With the time as quality measure I was not able to find a relevant difference
- Both metaphor have a good learning effect

Issues

- Sensors
  - Value jumps
- Cursor
  - form - sphere is maybe not the best choice for accurate selection
  - Better depth perception of the cursor
    - it appears as a circle not as a sphere
    - Depth movement of the cursor unintuitive toward the center

Future Work

- Better filtering of the sensor data for more smoothnes in the Orientation metaphor
- Finding more suitible form for the cursor (e.g. 3D arrow)
- Adding of shades to the cursor for better depth appereance
- Making the depth movement more intuitive (e.g. use the Camera-Cursor vector instead of the viewDirection vector)

Demo