Virtual Reality for Virtual Prototyping

Dr. Gabriel Zachmann
Institute for Computer Science II
University Bonn
Römerstr. 164, 53117 Bonn
zach@cs.uni-bonn.de
http://web.cs.uni-bonn.de/~zach/

Overview

I. Introduction to Virtual Reality
II. Introduction to Virtual Prototyping
III. Algorithms / Techniques / Issues

Part I
Quick Introduction to Virtual Reality

1. What is it?
2. Devices
3. Software System Design

What is it?

It is VR, when …

1. Real-time rendering,
2. Interaction in 3D in real-time,
3. Simulation in real-time,
4. Intuitive input devices (> 2D),
5. Stimulation of as many senses as possible,
6. Immersion and/or presence.

VR is *not*

- Cyberspace
- Any 3D computer graphics system with > 10 fps
- VRML


Art + Com

Devices

- Categories:
  - Output devices
  - Input devices
- Output devices:
  - Immersive displays
  - Haptic & Force feedback
  - Spatial audio
- Input devices:
  - Tracking devices
  - Glove
  - Other input devices

Immersive Displays

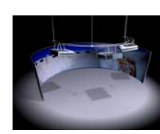
- Head-mounted display (HMD):
  - Relatively inexpensive, good immersion
  - Small field-of-view, low resolution


Sony

- Cave:
  - Non-invasive, pretty good immersion, high resolution
  - Low contrast, expensive,


Aalborg, DK

- Variants:
  - Workbench, Powerwall, Holobench, …
  - Curved screen projections


IMK FhG, DE

Characteristics

- The following table shows "rules of thumb" for several properties of the displays:

Display	Resolution	# users	Cost
HMD	Low	1	Low
Cave	High	4	High
Powerwall/ Curved screen	High	5–30	Medium

Haptic Feedback Devices

- Needed to render:
  - Contacts / Forces
  - Surface (haptic) texture
  - Guide user's hand

CyberGrasp

Boeing

Cybernet SpacePen

Tracking Devices

- Optical:
  - Fast, precise, can handle lots of markers, non-intrusive
  - Expensive, not always real-time, line-of-sight problem

- Inertial + ultra-sound:
  - Precise, no distortion
  - Mid-range price

Intersense

- Electro-magnetic:
  - Still most inexpensive, no line-of-sight problem
  - Distortion, intrusion

Polhemus Fastrak

Other Input Devices

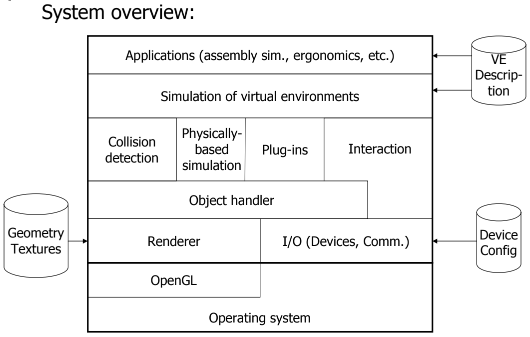
- Dataglove
  - Finger tracking, 17 or 23 sensors
  - Very intrusive, not very precise
  - No alternatives (yet)

- Spacemouse
  - 6 DOF desktop device

- Wand, flying joystick, …
  - 6 DOF tracked + buttons
  - For pointing and clicking

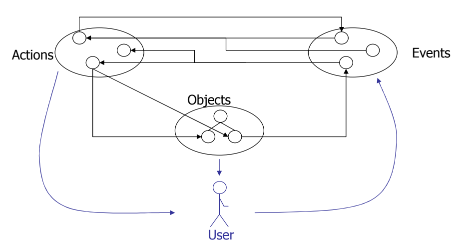
Software System Design

System overview:



Authoring Virtual Environments

- "Event-Action" or "Fields-and-Routes" concepts:



- Specification either with ASCII file, or with GUI.
- Only "middleware" for industrial applications!

VR Systems

- Commercial:
  - *VirtualDesign II* from VRCom (http://www.vrcom.de/)
    lots of functionality for several applications in VP;
  - Multigen-Paradigm (http://www.multigen-paradigm.com/)
    tendency towards military apps & 3D GIS (e.g., training);
  - *VisMockup* from EDS (http://www.eds.com/products/plm/teamcenter/vis/mockup/)
    not really VR, good integration with CAD infrastructure;
  - *WorldToolkit* & WorldUp from Sense8/EAI (http://www.sense8.com/)
    development library, many platforms;
  - Division/PTC (http://www.ptc.com/products/division/mockup.htm);
  - *Opus* Realizer from Opticore (http://www.opticore.se/)
    high-quality VR visualization (virtual showroom);
  - *InsideReality* from Schlumberger (http://www.sis.slb.com/content/software/virtual/)
    for oil & gas appls and geosciences;

- Academic / Non-commercial:
  - *Avalon* from ZGDV/IGD Darmstadt
    (http://www.igd.fhg.de/~avalon/)
  - *DIVE* from SICS (http://www.sics.se/dce/dive/)
    research system for distributed collaborative systems,
    little support for immersion, limited VR functionality;
  - *Alice* from VirginiaTech & CMU (http://alice.cs.cmu.edu/)
    browser plug-in, Windows-only, Python-scripting;
  - *VR Juggler* from Iowa State (http://www.vrjuggler.vrac.iastate.edu/)
    cross-platform, library with basic VR functionality;
  - *NPSNET* (http://www.npsnet.org/~npsnet/v/)
    programming toolkit for large-scale distributed battle sim
  - *Maverik* from U of Manchester (http://aig.cs.man.ac.uk/maverik/)
    toolkit providing some VR functionality

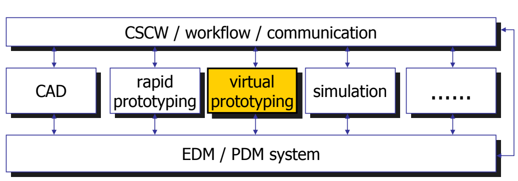
Part II
Introduction to Virtual Prototyping

1. Definitions
2. Applications
3. How to Build Your Own Lab

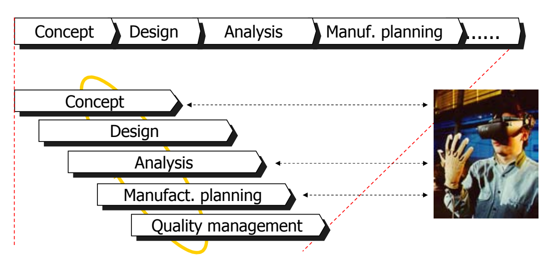
Definitions

- *Virtual Prototyping* (VP) =
  application of VR to simulate physical prototypes using product and process data, trying to emulate all characteristics of the physical prototype relevant to the application area as closely as possible.
- *Digital Mock-Up* (DMU) =
  all kinds of computer simulations of some aspect of a product; humans are not necessarily involved in the simulation.
- *Rapid Prototyping* (RP) =
  automatically construct physical models from CAD data. ("3D printing")

Where does VP fit in the IT infrastructure?

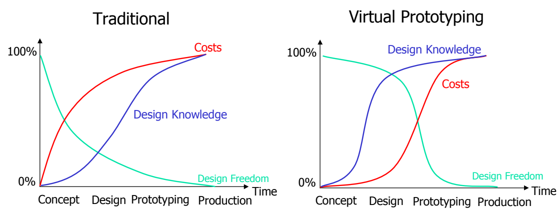


VP helps to implement Concurrent Engineering

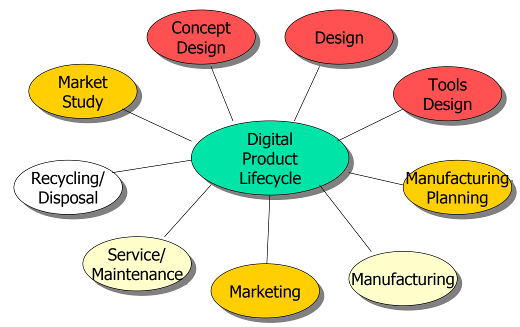


Front Loading by VP

Design freedom / Costs / Design Knowledge
vs.
Product development time



Application Areas



Styling Review

- Presentation: Powerwall
- Teams discuss style, possible changes, etc.
- High demands on rendering:
  - Huge polygon counts
  - Lacquer, gloss, glass, mirrors
  - Material properties should be physically correct
- Well established in today's design process in automotive industry


DaimlerChrysler VRC

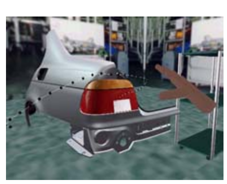

Volkswagen, FhG-IGD

Concept Design

- Idea:
  Roughly sketch design (e.g., car body) in VR
- More practical:
  Import concept from CAD/Softimage/AW, do only small "what-if" changes in VR

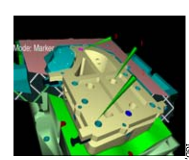
Assembly Simulation

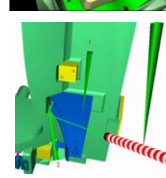
- Analysis:
  - Can it be assembled?
  - Can it be serviced/maintained?
  - What is the physical stress on the worker?
  - Document problems/suggestions
- Path generation
- Tap into knowledge of experienced workers & engineers
- Very high demands on VR:
  - Physically-based simulation
  - Lots of functionality
  - Needs natural hand interaction



Tools Design Review

- Reduction of error probability:
  - Error in design of punching machine can costs millions; possibly a whole part of the assembly line must be redesigned!
- Analysis:
  - Tears
  - Disposal of remainders
  - Safety for worker
- Advantages:
  - 1:1 rendering
  - Efficient viewing interaction
  - Cuts in real-time


IGD, BMW



Ergonomics of Customers

- Humans are subject of investigation
- Disadvantages of CAD tools:
  - Man-in-the-loop
  - Difficult user interface
  - No immersion


IGD / BMW


VW, VRCom

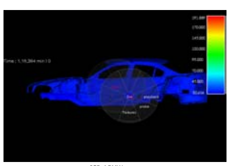
Interior

- Analysis:
  - General impression?
  - Character?
  - Space?
- High demands on rendering:
  - Correct lighting simulation
  - Correct optical material properties
  - Good tone mapping of display
  - Large plygon counts

IGD / BMW

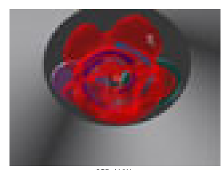
Immersive Scientific Visualization

- Possible advantage:
  - Immersion helps to better understand the huge amounts of data
- Scenarios:
  - Cooling process in lacquering the body
  - Virtual wind tunnel
  - Crash simulation visualization

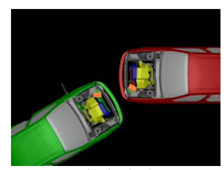


IGD / BMW

IGD / VW



IGD / VW



National Crash Analysis Center

Showroom

- Idea: no real cars at dealers any more; instead: show car on Powerwall
- Advantages:
  - Can have more models "on display"
  - Customer can customize car with "his" favoured combination of colors, accessories, variants – *and* see it immediately
- Demands: similar to Styling review & Interior

Marketing

- Make product known through "cool" technology/games
- Problem with VR: throughput

IGD / UBS

Factory Planning

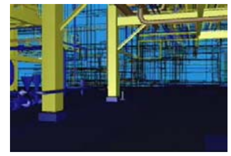
- Visualize plans / factory layouts created by commercial desktop systems
- Advantage:
  - Easier to spot problems
  - Interactive modification

Audi

RWTH Aachen, SMS Demag

Walk-Throughs

- Immersion really helps, even when just Powerwall
- Sells much better to public and top executives

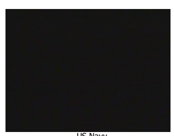


Training

- Learning by doing (in VR)
- Advantages:
  - Available as early as virtual prototype
  - Flexible configuration
  - No danger for trainees (or patients)
  - Easier tranfer to real world than with conventional training methods

real

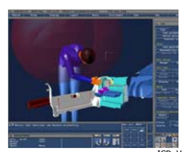
simulated

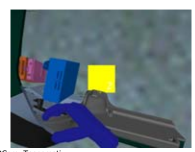


US Navy

Front-end for CAD systems

- Benefits:
  - Integration into IT infrastructure
  - Intuitive and immsersive UI for CAD
  - Get lots of features from CAD into VR
- Example Robcad/Man & *VirtualDesignII*:
  - Specification of Robcad/Man poses in VR
  - Playback paths from Robcad in VR
  - Online ergonomic analysis of worker pose





IGD, VRCom, Tecnomatix

How to Build an Industrial VR Lab

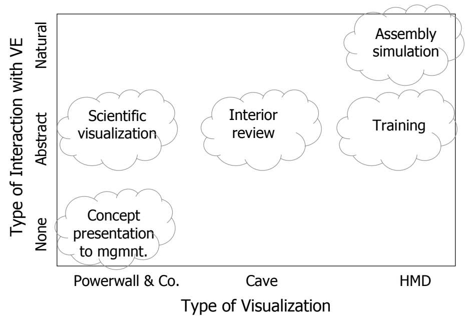
1. Identify applications, tasks, needs, limitations:
   - What will it be used for?
   - Can the task be done with conventional CAD?
   - How would VR be better? (faster, better, cheaper)
   - Perform feasibility study!
   - Try to calculate the ROI.
   - What will it *not* be able to do? (render 1,000,000 pgons with 30 fps, track the complete body, build cars, …)
   - Don't oversell it!
2. Operation:
   - Who will run the lab? (designated person?)
   - Will it be a in-house service or self-service facility?

---

1. Identify usage / hardware needed:
   - How often will the lab be used? By how many people?
   - What display is needed? (Powerwall, Cave, HMD, …)
   - One large central facility, or many distributed sites?
   - Requirements of tracking (accuracy, line-of-sight, sample rate)?
   - What computers? (PC? SGI? HP? Sun?)
   - What's the budget?
2. Identify software:
   - Is there commerical VR software that can do it?
   - If not: who can build it? Will the development fit with the company's product schedule / business plans?
   - Other tools needed? (converters, simplifier, radiosity, etc.)

Decision aid to choose hardware

- Try to position your application(s) in this chart:

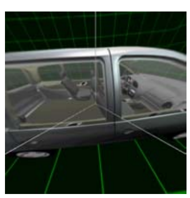


Part III
Algorithms / Techniques / Issues

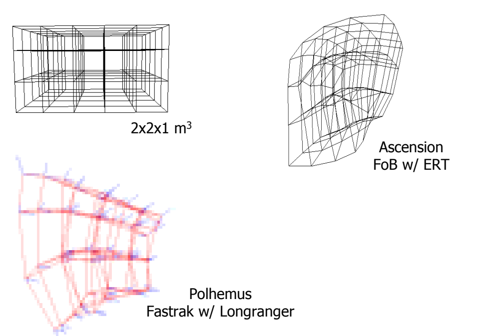
1. Tracking correction
2. Collision detection & Force feedback
3. Beyound Phong

Correcting Tracking Errors

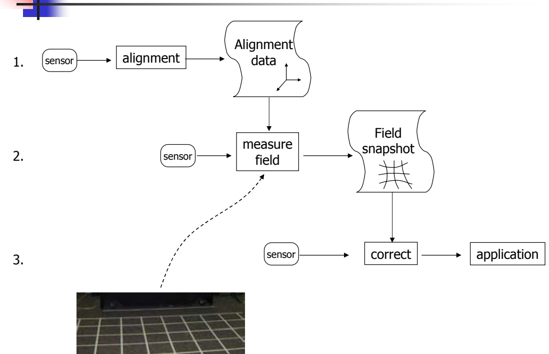
- Problem: wrong tracking leads to
  - Distortion of images in Cave & Co (stationary displays)
  - Mismatch between visual and proprioceptive feedback (HMD)
- Most serios error sources:
  - Lag (leads to other problems, too)
  - Distortion of electro-magnetic field



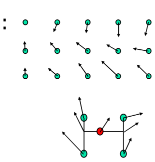
Typical Distortion with Electro magn. Tracking



General correction procedure



Two simple correction algorithms



- Lookup table + trilinear interpolation:
  - Resample field snapshot
  - Do trilinear interpol at run-time to get estimate of error
  - Subtract estimated error
- Hardy's Multiquadric:
  - Compute interpolating function

$$f(\mathbf{P}) = \sum \mathbf{A}_i \omega_i(\mathbf{P}) \quad , \quad \mathbf{A}_i \in \mathbb{R}^3$$

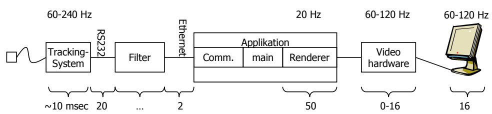
$$\omega_i(\mathbf{P}) = \sqrt{(\mathbf{P} - \mathbf{P}_i)^2 + R^2}$$

  - At run-time evaluate

$$f(\mathbf{P})$$

Fighting latency

- Latency pipeline:

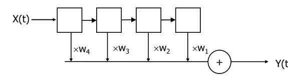


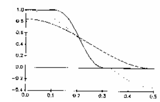
- Techniques to reduce lag:
  - Clever communication between device & app
  - Predictive filtering
  - Rendering with levels-of-detail
  - Etc.

Two simple Filtering Techniques

- Finite impulse response filter (FIR):

$$y_t = \sum_{i=-k}^{k} w_i x_{t+i}$$

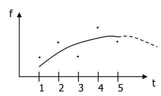




Don't choose all weights equal.
With 3 weights, choose ¼ , ½ , ¼ .

- Fitting a polynomial:

$$f(i) = a_0 + a_1 i + \ldots + a_n i^n$$



using current history of sample solve

$$\mathbf{A}^T\mathbf{A}\mathbf{a} = \mathbf{A}^T\mathbf{f} \quad , \quad A_{ij} = i^j \quad , \quad \mathbf{f} = (f(1), f(2), \ldots f(n))$$

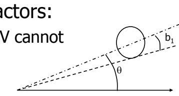
evaluate f "in the future".
Precompute LU decomposition of **A**.
Fast enough for small degrees.

- Kalman filter:
  - Optimal for linear systems (user motion is not)
  - Non-trivial to implement
  - Not necessarily easier to adjust or better results

Level-of-Detail selection

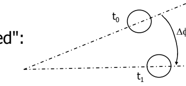
- Choose level based on human factors:
  - Details of objects at periphery of FOV cannot be seen:

$$k_1 = \begin{cases} e^{-(\theta - b_1)/c_1} & , \theta > b_1 \\ 1 & , \text{sonst} \end{cases}$$



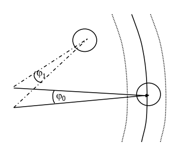
  - Fast moving objects appear "blurred":

$$k_2 = e^{-(\Delta\varphi - b_2)/c_2}$$



  - Objects outside the focus, too:

$$k_3 = e^{-(|\varphi_0 - \varphi_1| - b_3)/c_3}$$



- Bestimmung des LODs:
  1. $k = \min\{k_i\} \cdot k_0$, oder $k = \prod k_i \cdot k_0$
  2. $r_{\min} = 1/k$
  3. Select level l such that all pgons are larger than $r_{\min}$

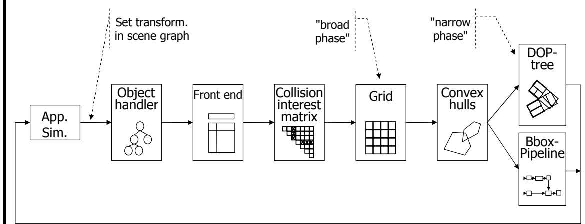
- Predicitve LOD selection:
  - Otherwise: sudden "jerks" in framerate
  - Optimization problem:
    maximize Σ_S Benefit(Obj,Level)
    under the constraint Σ_S Cost(O,L,R) $\leq$ max. frame time
  - Compute good suboptimal solution incrementally

- Recent work: View-dependent triangulation of NURBS on-the-fly
  - Sew adjacent patches together across trimming curve
  - Calculate max allowed error for each patch
  - Current patch triangulation error < max error?
  - If not: refine triangulation or make coarser
  - Any trimming loops appearing in current frame?
  - If so: create new triangulation for the patch
  - Performance:
    ca. 1,500 patches with 10 fps avg and 2 pixels error

Collision Detection

- Base technology:
  - Physically-based simulation
  - Natural object interaction (grasping)
  - Tolerance verification

Collision Detection Pipeline

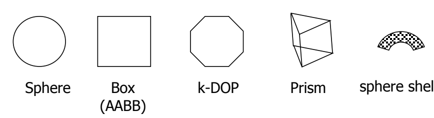


Hierarchical collision detection

- Build hierarchy of BVs
- Traverse 2 BV hierarchies simultaneously:

  <u>traverse(A, B)</u>
  if (A,B) do not overlap → return
  if A is leaf && B is leaf → check polygons enclosed
  forall children A_i , forall children B_j of B:
  traverse(A_i , B_j)

- Differences among hierarchical algorithms:
  - Type of BV

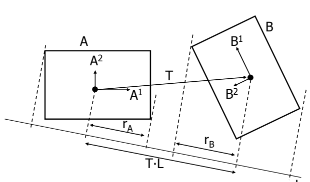


  - Construction of the hierarchy

Two popular BVs

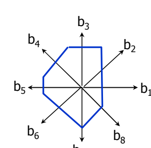
- OBB (oriented bounding box):
  - Separating axis test:

    $|T \cdot L| < r_A + r_B \Rightarrow$ A,B do not overlap
  - Suffices to check exactly 15 axes!



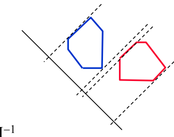
- k-DOP:

  DOP $D = \bigcap_{i=1}^{k} H_i \quad , \quad H_i : \mathbf{B}_i \cdot \mathbf{x} - d_i \le 0$



  - Representation:

    $D = (d_1, \ldots, d_k) \in \mathrm{R}^k$

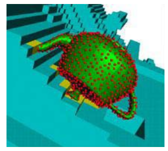


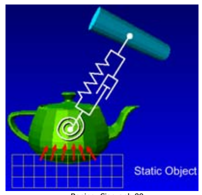
  - Overlap test: check k/2 intervals
  - Transformation of "tumbled" DOPs:

$$d_i' = \mathbf{B}_i \begin{pmatrix} \mathbf{b}_{j_1^i} \\ \mathbf{b}_{j_2^i} \\ \mathbf{b}_{j_3^i} \end{pmatrix}^{-1} \begin{pmatrix} d_{j_1^i} \\ d_{j_2^i} \\ d_{j_3^i} \end{pmatrix} + \mathbf{B}_i \mathbf{o} \quad , \quad \mathbf{b}_j = \mathbf{B}_i \mathbf{M}^{-1}$$

Haptic Rendering

- Simple algorithm:
  - Represent objects as voxels and point clouds
  - Calculate force on each point
  - Calculate total force on object
  - Calculate force on haptic device (spring-and-damper model)

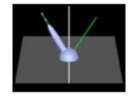


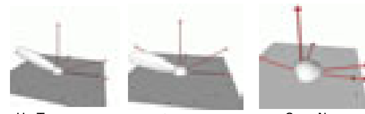


Boeing, Siggraph 99

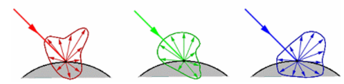
Beyound Gouraud & Phong

- Real-world materials do not behave like Phong
- More complicated lighting models:



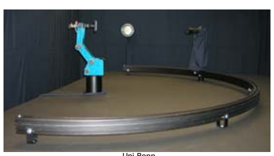


- Many real-world materials are still more complicated:

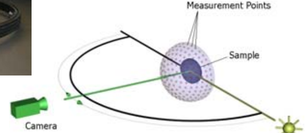


BRDF / BTF

- Better to measure optical material properties:
  - Take sample of material, take "standard" light source
  - BRDF: measure incoming light per viewing/lighting direction
  - BTF: take photo (= texture) per viewing/lighting direction



Uni Bonn



- Comparison:
  BTF rendering vs. simple texture

Uni Bonn

- Challenges:
  - Data size / compression (BTF = *x* GB)
  - Fast rendering
  - When BRDF / when BTF?

Challenges / Trends

- Force feedback in complex scenes and large volume
- Un-tethered devices
- Deformable objects (plastic parts, hoses, …)
- Rendering of complex optical material properties
- Installation of VR at SMEs (e.g., suppliers)

References

- Kay Stanney (ed.):
  Handbook of Virtual Environments.
  Lawrence Erlbaum Associates, 2002.
- Singhal & Zyda:
  Networked Virtual Environments.
  Addison-Wesley, 1999.
- Most other VR books are old …

The End.