

MMVR '09 Workshop – January 19, 2009
 Simulator Construction Without the Agonizing Pain:

Simulation Virtual Environment Development

Eric Acosta

National Capital Area Medical Simulation Center
 Uniformed Services University
<http://simcen.usuhs.edu>

1

Virtual Reality-based Simulation



(Video)

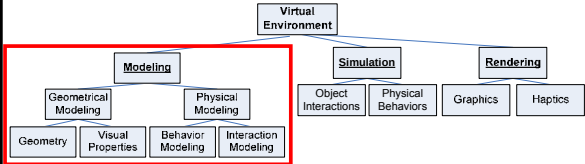
2

Presentation Outline

- Elements of a virtual environment (VE)
 - Modeling, simulation, and rendering
- Simulation VE development
 - Mesh- and Voxel-based haptic interactions
- VE Development for multi-core CPUs
 - Pipeline VE architecture

3

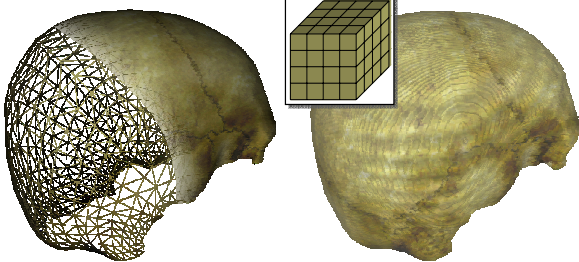
Elements of a Virtual Environment (VE)



4

VE Geometrical Modeling

- Model geometrical representations



Polygonal Mesh (surface) Voxels (volumetric)

5

VE Geometrical Modeling

- Model visual properties (appearance)



Color Material Texture Map

Color Material Mesh Texture Mapped Mesh

6

VE Physical Modeling

- Model object interactions
- Example:
 1. Calculate force ($F = kd$)
 2. Apply forces

Rigid Hand Model

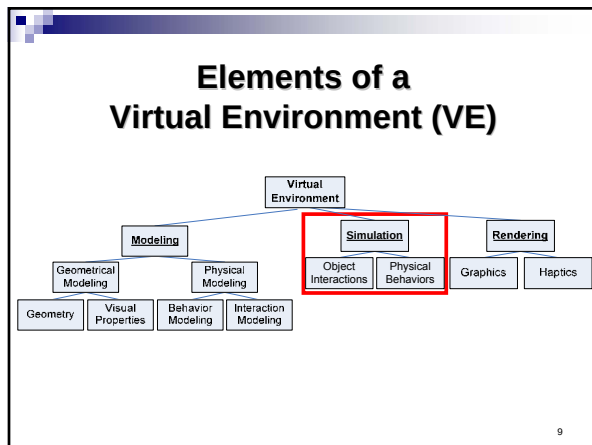
Mass-Spring Model

VE Physical Modeling

- Model object behavior
 - Physical equations → behavior

Mass-Spring (video)

Rigid Bodies (video)



Object Interaction Simulation

- Simulate modeled VE interactions
- Steps:
 - Collision Detection
 - Collision Response
 - Physical behavior update

Object Interaction Simulation

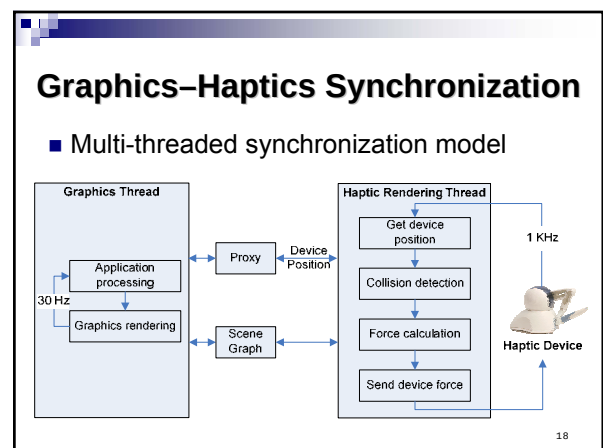
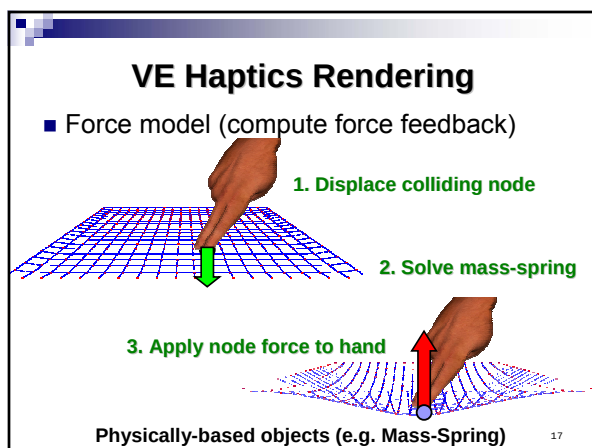
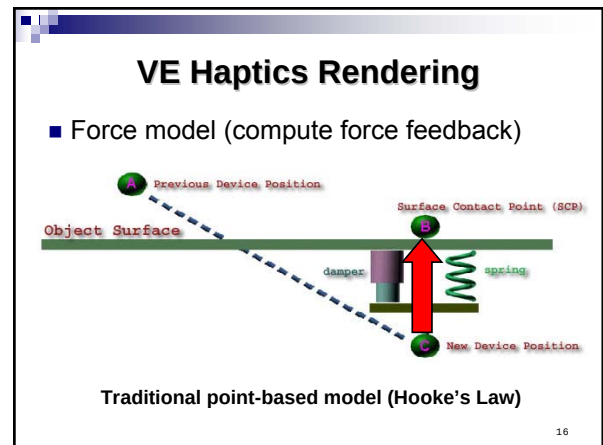
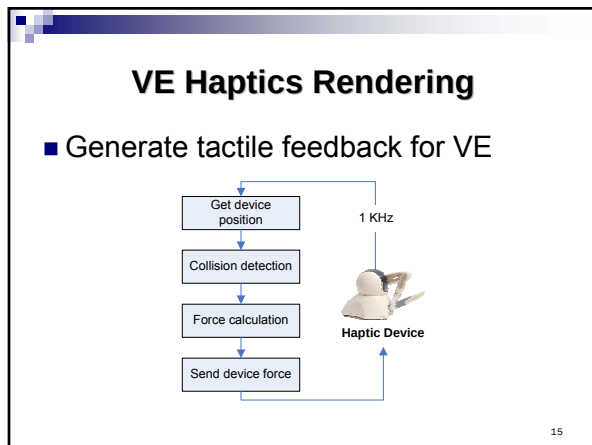
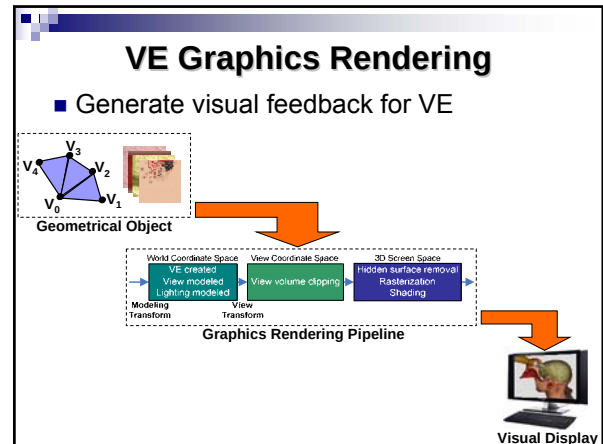
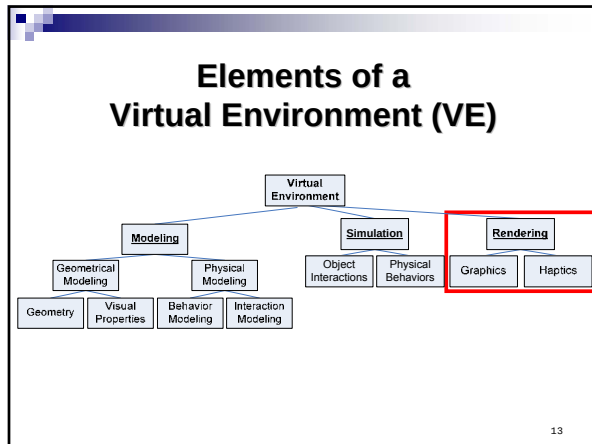
- Collision detection

1. Broad Phase

2. Narrow Phase

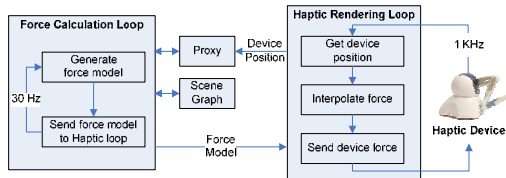
Object Interaction Simulation

- Collision response (compute physical model)
 - Interpret collision detection results
 - Compute modeled interaction
- Behavior update (compute physical model)
 - Compute underlying physical equations
 - Update object (e.g. transforms/geometry)



Graphics-Haptics Synchronization

- Multi-rate haptics synchronization model



19

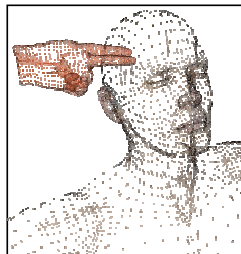
Presentation Outline

- Elements of a virtual environment (VE)
- **Simulation VE development**
 - Mesh- and Voxel-based haptic interactions
- VE Development for multi-core CPUs

20

Mesh-based Haptic Interactions

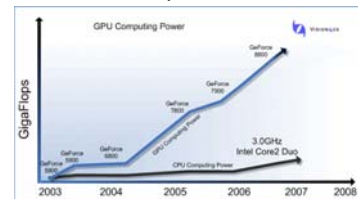
- Collision detection
 - When / where contacting
- Where (compute force)
 - Surface point tracking



21

Collision Detection

- CPU intensive
- Can we free up CPU?
- GPU performance
 - Rapidly increasing (> CPU)
 - Trend: GPU → computational resource

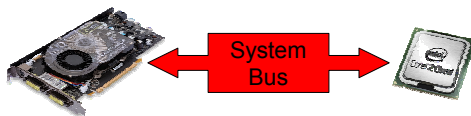


Source:
Vision4CE

22

Collision Detection

- GPU read-backs too slow!!!



- GPU/CPU hybrid approach
 - GPU (broad phase)
 - CPU (narrow phase)

23

Collision Detection (Broad Phase)

- Bounding volume tree (commonly used)
 - Update/rebuild tree (e.g. deformations)
 - Large number of bounding volume tests
- Quick-Cullide GPU-based algorithm [6]
 - No topology assumed
 - Linear performance
 - Minimal GPU read-backs

24

Quick-Cullide [6]

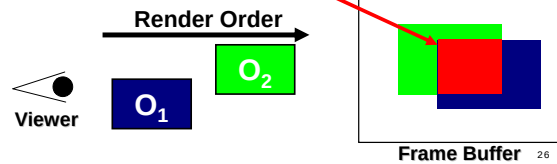
- GPU-based algorithm
 - Render objects
 - Visibility testing (GL_NV_occlusion_query [7])
- Generate potentially colliding set (PCS)
 - Test in narrow phase

25

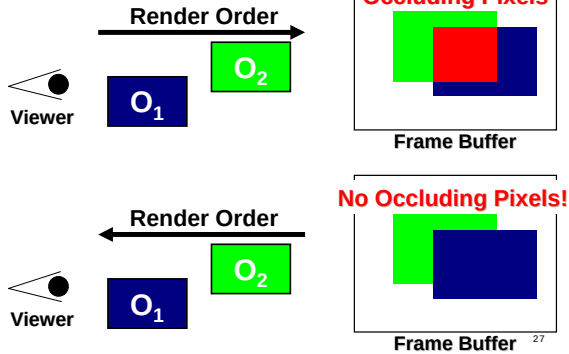
Quick-Cullide Visibility/Occlusion Testing

- GL_NV_occlusion_query [7]
 - Number pixels pass depth test
 - Depth test → GL_EQUAL (behind/occluded)

**Number Pixels > 0
Occlusion → Possible Collision**



Visibility/Occlusion Testing



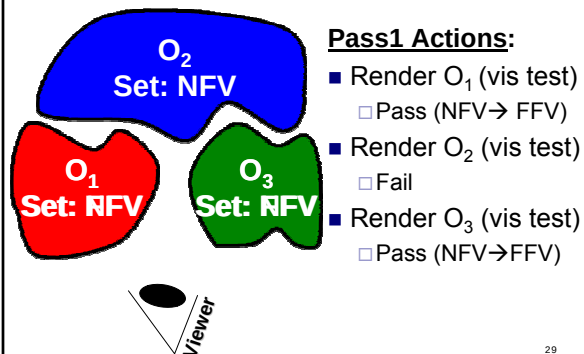
27

Quick-Cullide

- Rendering order matters → 2 Passes
 - $O_i, i = 1 \dots n$
 - $O_i, i = n \dots 1$ (reverse rendering order)
- Objects sets (classification)
 - FFV : fully visible pass 1
 - SFV : fully visible pass 2
 - NFV : not fully visible pass 1 or 2
 - **BFV : fully visible pass 1&2 (remove/prune)**

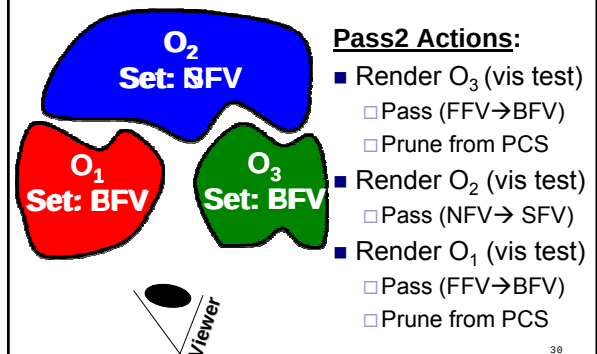
28

Quick-Cullide Example: Pass 1



29

Quick-Cullide Example: Pass 2



30

Quick-Cullide Example: Pass 1 (Iteration 2)



Pass1 Actions:

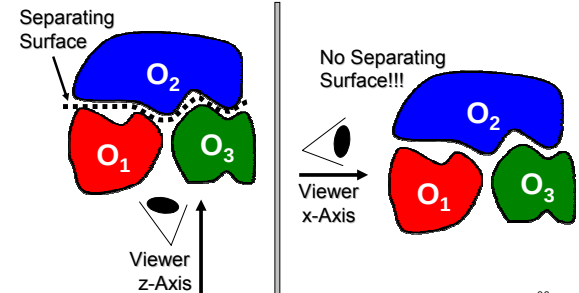
- Render O_2 (vis test)
 - Pass (SFV $\rightarrow$ BFV)
 - Prune from PCS



31

Quick-Cullide

- Best culling $\rightarrow$ Render along 3 axis!!!



32

Quick-Cullide

- Drawback
 - Render up to 2 times/axis (6 times)
 - Run until PCS empty or not changed
 - Collision detection at image resolution
- Potential gotcha
 - Not rendered $\rightarrow$ Not colliding (false negative?)
 - Example: plane $\perp$ camera
 - Tip: Use slightly rotated axis

33

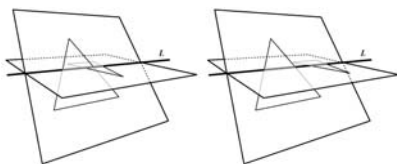
Quick-Cullide

- Performance tips:
 - Test with object bounding boxes first
 - Turn off extras (e.g. textures, lighting, etc.)
 - Orthogonal projection

34

Triangle-Triangle Collision Detection (Narrow Phase)

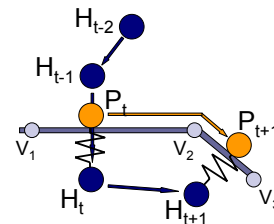
- Triangle-Triangle test
 - Based on (Moller, 1997) [4]
- Source code:
 - http://www.cs.lth.se/home/Tomas_Akenine_Moller/code/



35

Haptic Interactions: Surface Tracking

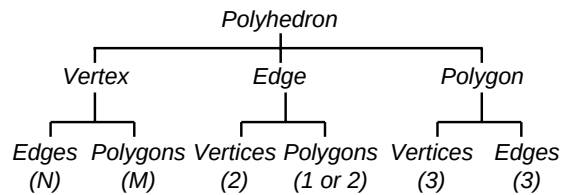
- Point-based force calculation



36

Haptic Interactions: Surface Tracking

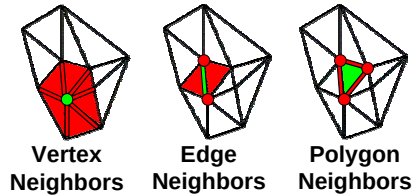
- Triangle-triangle test → deepest vertex
- Local collision detection [8]
 - Neighbor primitive database



37

Haptic Interactions: Surface Tracking

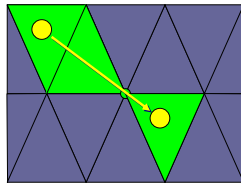
- Converge on closest primitive [8]
 - Iterative search → only look at neighbors
 - Go to primitive shortest distance



38

Haptic Interactions: Surface Tracking

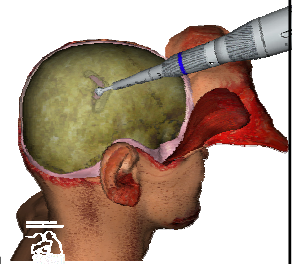
- Only consider neighbors
- Performance → polygon count independent



39

Voxel-based Interactions Overview

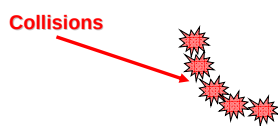
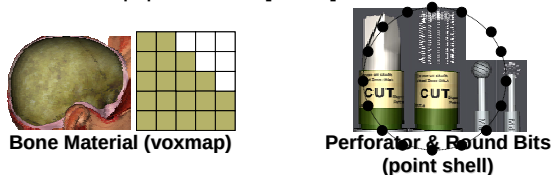
- Collision detection
 - Voxmap-pointshell
- Tool interactions
 - Haptic
 - Bone drilling
- Volume rendering
 - GPU → Mesh/volume visual blending



40

Voxel-based Collision Detection

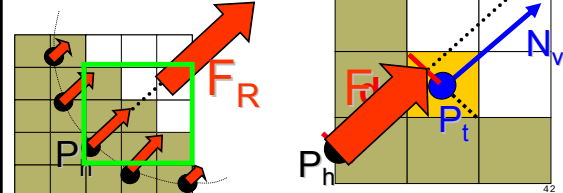
- Voxmap-pointshell [9, 10]



41

Voxel-based Haptic Interactions [10]

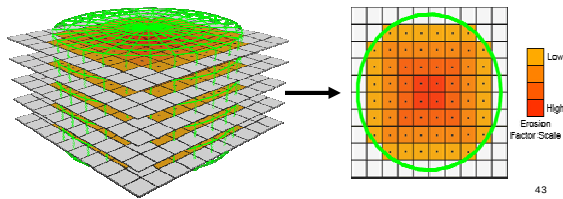
- Plane perpendicular to surface normal
- Force normal to surface
 - Voxel point on plane (P_t)
 - $d = (P_t - P_h) \cdot N_v$
 - $F = N_v k d$



42

Voxel-based Bone Drilling [10]

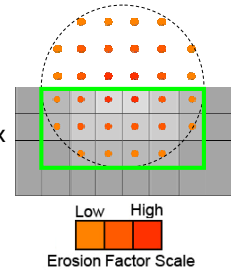
- Erosion points
 - Voxelize primitive shape
 - Surface & interior points
 - Pre-computed erosion factor



43

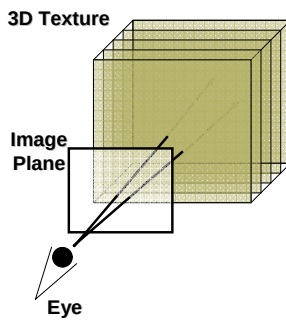
Voxel-based Bone Drilling [10]

- Collision detection
- Bone erosion
 - $B'_v = B_v(1 - ES)$
 - If $B_v < \text{threshold}$, then $B_v = 0$
- Track modified bounding box
 - Update gradients within box



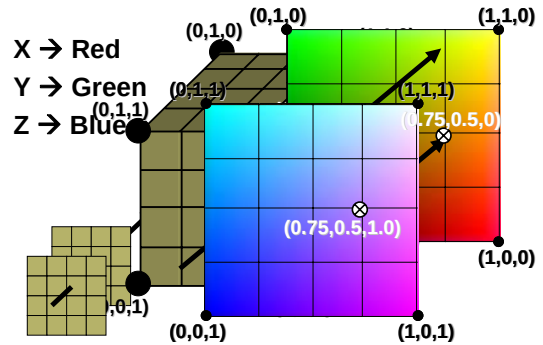
44

Volume Raycasting [8]

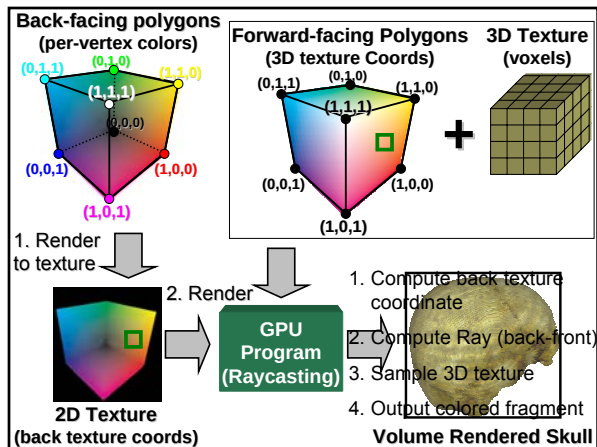


45

GPU-based Ray-casting



46



Volume + Mesh VE Blending

- Does not blend well with mesh rendering!



48

Volume + Mesh VE Blending

- Common Solution → Marching Cubes



Image from
Sewell et al. [2]

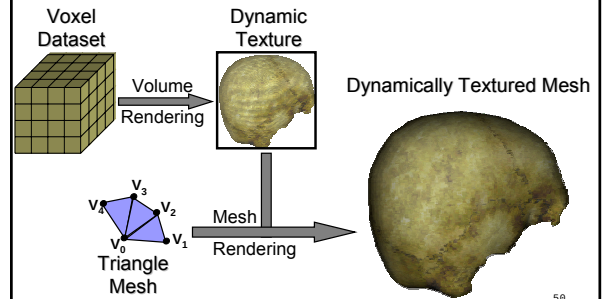


Image from
Eriksson et al. [3]

49

Volume + Mesh VE Blending

- Our Solution → Dynamic Texture Mapping



50

Visual Comparison



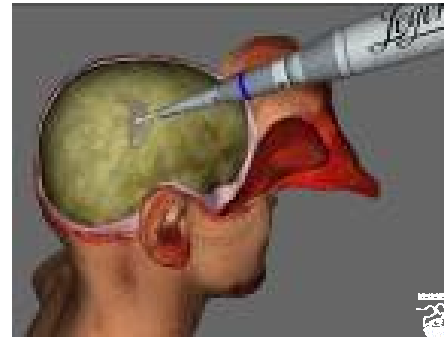
Volume Rendered Skull



Dynamically Textured Skull

51

Bone Drilling + Dynamic Texture



(video)

52

Presentation Outline

- Elements of a virtual environment (VE)
- Simulation VE development
- VE Development for multi-core CPUs**
 - Pipeline VE architecture

53

VE Development for Multi-core CPU Systems

- Major VE Goal:
 - Real-time simulation / visualization
- Limited CPU resources
 - Traditional bottleneck → VE limitations
- Multi-core CPUs
 - Increase computational resources
 - Becoming standard

54

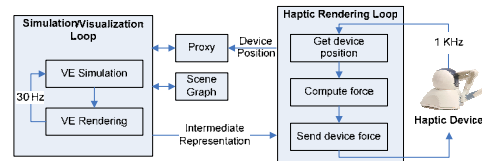
Multi-core CPUs

- Q: How to get the most out of each core?
- A: Multi-threading (parallel work)
- Q: Workload distribution over threads / cores?

55

Common Multi-threading Approaches

- Application specific
- Task-specific threads (static partition)
- Single-threaded simulation/visualization loop



Single-thread for simulation & visualization

56

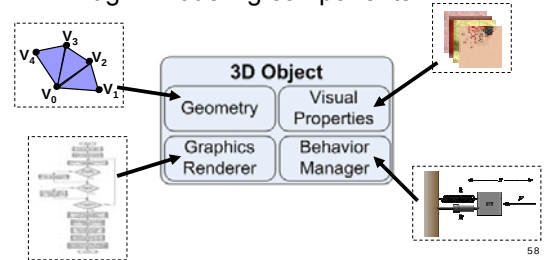
Pipeline VE Architecture

- Multi-core enabled API
- Distributes VE workload across all cores
- No assumptions about problem domain
- Hides parallel processing from developer!

57

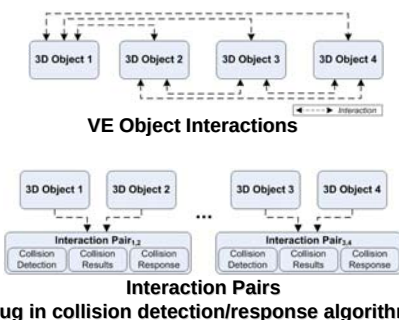
Pipeline VE Architecture: Modeling VE

- Modularized 3D Object
 - Plug in modeling components



58

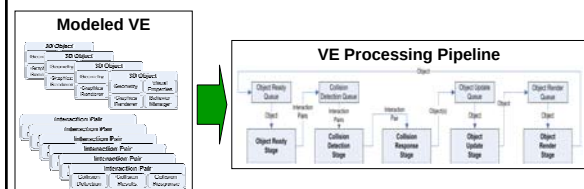
Pipeline VE Architecture: Modeling VE Interactions



59

VE Pipeline

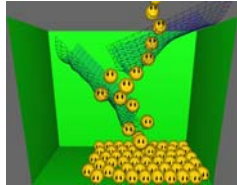
- VE partition (objects, interaction pairs)
 - Dynamic workload distribution to cores



Multi-threaded Pipeline Processes Modeled VE

60

VE Pipeline Parallelism: (Benchmark VE)



Mass-Spring Particles
(video)



Intel Thread Profiler Shows
Very Good CPU Core
Utilization

61

Benchmark VE Performance

Test System 1:

- Dual 1.86GHz Intel Xeon quad-core processors (8 cores)
- 4 GB RAM
- ATI ES1000 (16 MB) graphics
- Windows 2003 Server

VE Speedup

Thread Efficiency

62

Head Trauma VE Performance

Test System 2:

- 3.0 GHz Intel Core2 Extreme quad-core processor
- 2 GB RAM
- 2 – NVidia GeForce 8800 GTX (768 MB) (in SLI mode)
- Windows XP

VE Speedup

Thread Efficiency

63

VE Pipeline Architecture

- Plug-in design → easy integration
- Developers focus on modeling VE
- Dynamic workload distribution to cores
- Develop once → automatic scale performance based on CPU cores

64

Come by and see us!

- Poster presentation (Wednesday)
 - "A Multi-Core CPU Pipeline Architecture for Virtual Environments"
- Well (Tuesday/Wednesday afternoon)
 - Head trauma simulator demo

65

References

1. M.C. Cavusoglu and F. Tendick. "Multirate simulation for high fidelity haptic interaction with deformable objects in virtual environments." *ICRA*, pp. 2458-2465, 2000.
2. C. Sewell, D. Morris, N. Blevins, F. Barbagli, K. Salisbury. "Evaluating Drilling and Suctioning Technique in a Mastoidectomy Simulator." *MMVR '07*.
3. M. Eriksson, H. Flemmer and J. Wikander. "A Haptic and Virtual Reality Skull Bone Surgery Simulator." *World Haptics*, mar 2005.
4. T. Moller. "A Fast Triangle-Triangle Intersection Test." *Journal of Graphics Tools*, 2(2), 1997.
5. C. Basdogan and M. A. Srinivasan. "Efficient point-based rendering techniques for haptic display of virtual objects." *Presence*, 8, pp 477-491, 1999.
6. N. K. Govindaraju, M. C. Lin, and D. Manocha. "Quick-cullide: Fast inter- and intra-object collision culling using graphics hardware." *Virtual Reality*, pp 59-66, 2006.

66

References

7. http://oss.sgi.com/projects/ogl-sample/registry/NV/occlusion_query.txt
8. http://www.daimi.au.dk/~trier/?page_id=98
9. M. Renz, C. Preusche, M. Potke, H. Kriegel, and G. Hirzinger. "Stable haptic interaction with virtual environments using an adapted voxmap pointshell algorithm. *Eurohaptics*, pp. 149–154, 2001.
10. Acosta and A. Liu. "Real-time volumetric haptic and visual burrhole simulation." *IEEE Virtual Reality*, pp. 247–250, 2007.