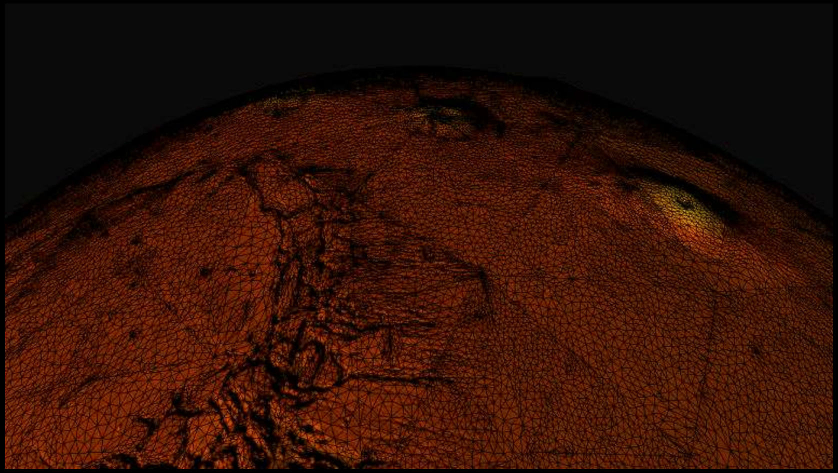
# Multiresolution graphics on commodity graphics platforms



EUROGRAPHICS Italy 2003

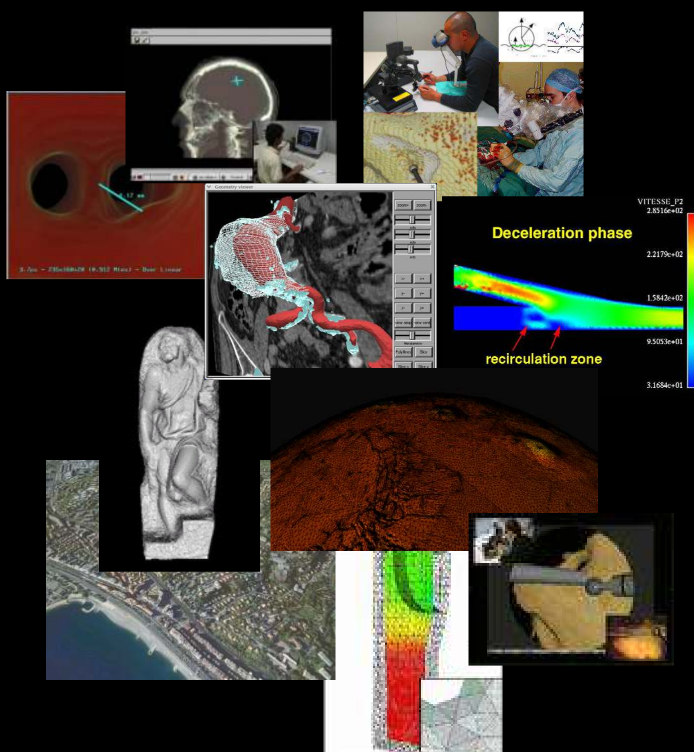
Enrico Gobbetti

CRS4 - Visual Computing Group
Italy

# Today's plan

- Who we are
  - CRS4 / ViC
- Short introduction to multiresolution graphics on commodity graphics platforms
  - Context, motivation, characterization of type of solution
- Two recent application examples
  - Large scale terrain rendering (IEEE Visualization 2003)
  - Global illumination simulation (Eurographics 2003)

# CRS4 - Center for Research, Development, and Advanced Studies in Sardinia




POLARIS
Edificio 1
C.P. 25
09010 PULA (CA)
Italy

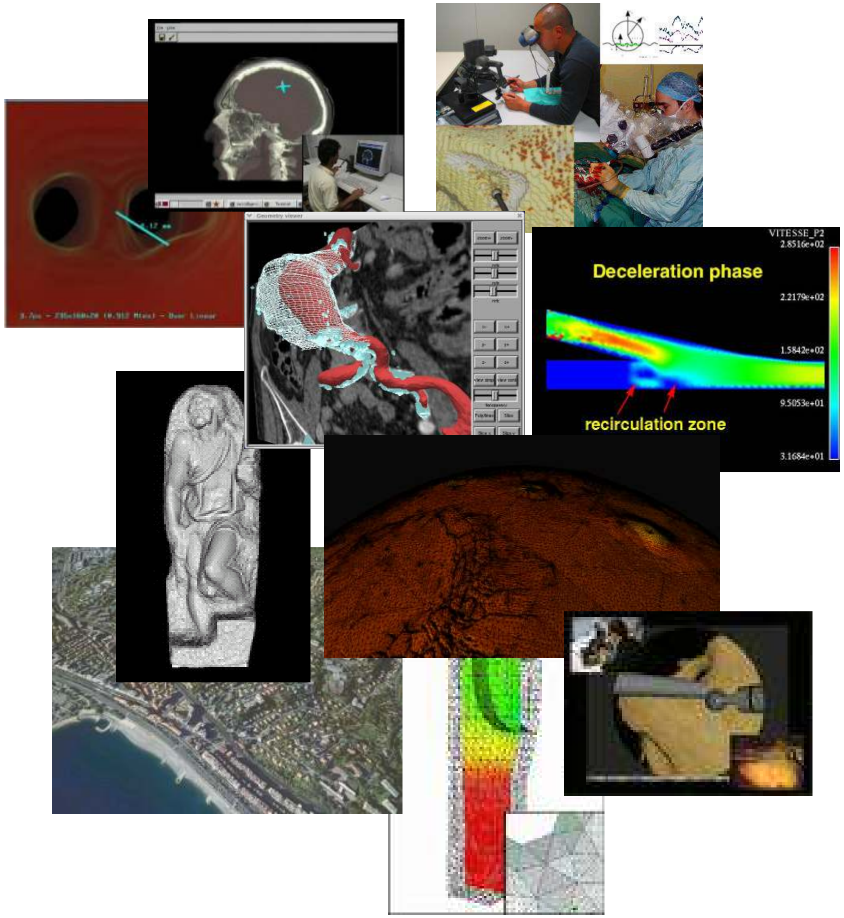
http://www.crs4.it/

# Who we are

- Non-profit consortium of private and public entities
  - C21(RAS), IBM-Italy, STM, Tiscali, Saras, U.of Cagliari and U. of Sassari
  - Established in 1991 in Cagliari (Sardinia, Italy)
- Facts (2002)
  - Resarch staff of 6 research directors, 80 researchers, 8 sysadm
  - Funding: 2.9M from contract research (~50% of turnover)
    - 50% MIUR, 20% EU, 30% Industry

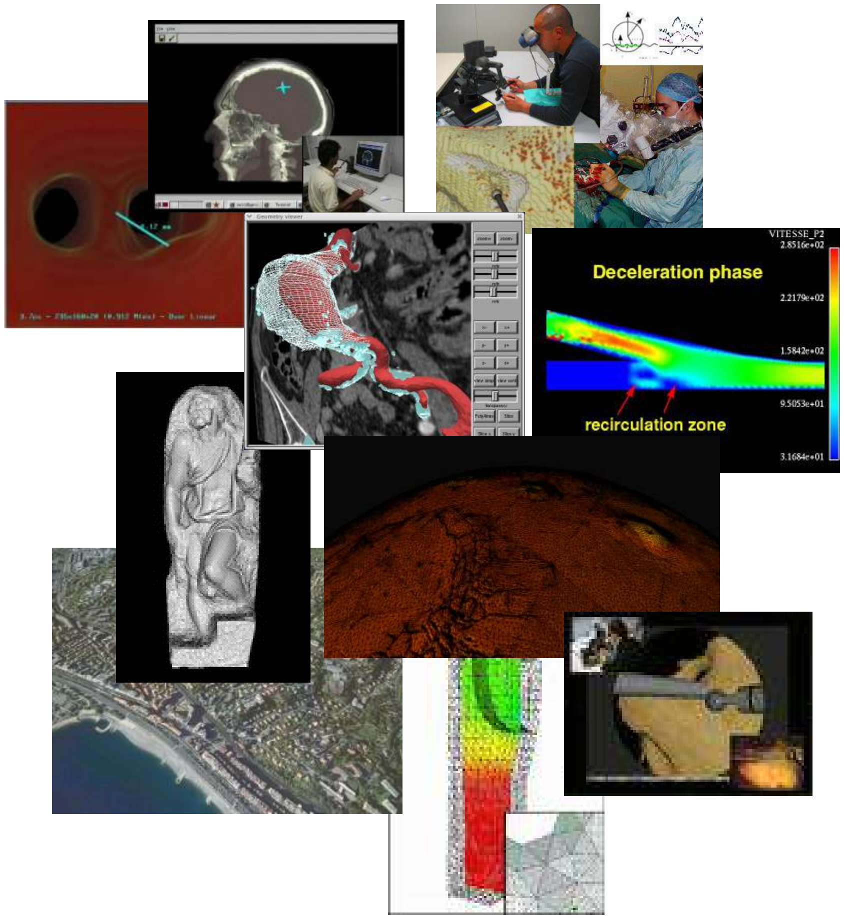

# Enabling technologies and thematic areas

- CRS4 key strengths include:
  - High Performance Computing and Networks
  - Computational Mathematical Methods
  - Visual Computing
  - Information Systems
- CRS4 primary focus is on solving problems stemming from:
  - Environmental Sciences
  - Life Sciences
  - Energy
  - Information Society

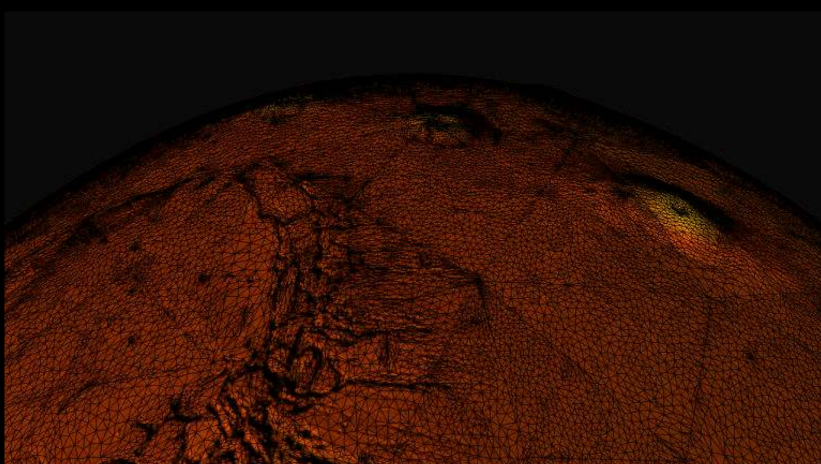


# Visual Computing Group

- 1 director + 6 staff researchers
- Enabling technology RTD
  - Multiresolution modeling
  - Time critical rendering
  - Scientific visualization
  - Haptics
- Applications in all CRS4 thematic areas
  - See http://www.crs4.it/vic/



# Introduction to multiresolution graphics on commodity graphics platforms



EUROGRAPHICS Italy 2003

Enrico Gobbetti

CRS4 - Visual Computing Group
Italy

# The goal

Rapid processing and interactive rendering of complex 3D scenes with high visual and temporal fidelity on a graphics PC platform

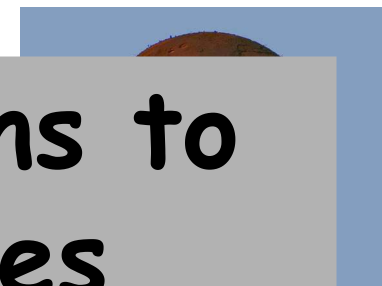
# The goal

Rapid processing and interactive rendering of complex 3D scenes with high visual and temporal fidelity on a graphics PC platform

# Scene Complexity (2003 interactive apps)

*St. Matthew* statue scanning (0.25mm spacing)

~127M vertices



Terr

~

**Hundreds of millions to billions of samples**

Isosurface (Bone, *Visible Female* dataset)

~250M vertices

CAD Models (UNC *DoubleEagle* Tanker)

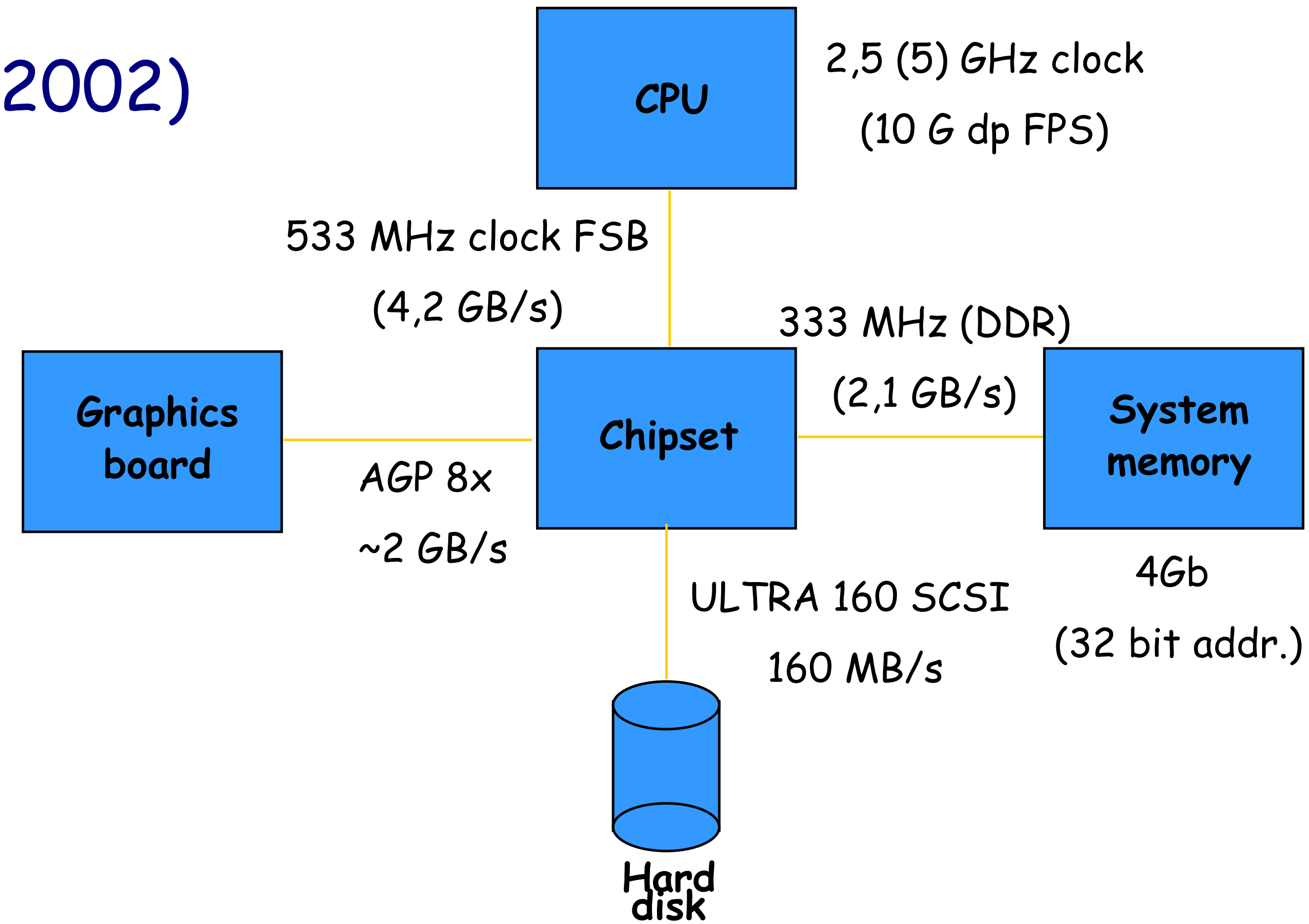
~45M vertices, ~130K objects

# The goal

Rapid processing and interactive rendering of complex 3D scenes with high visual and temporal fidelity on a graphics PC platform

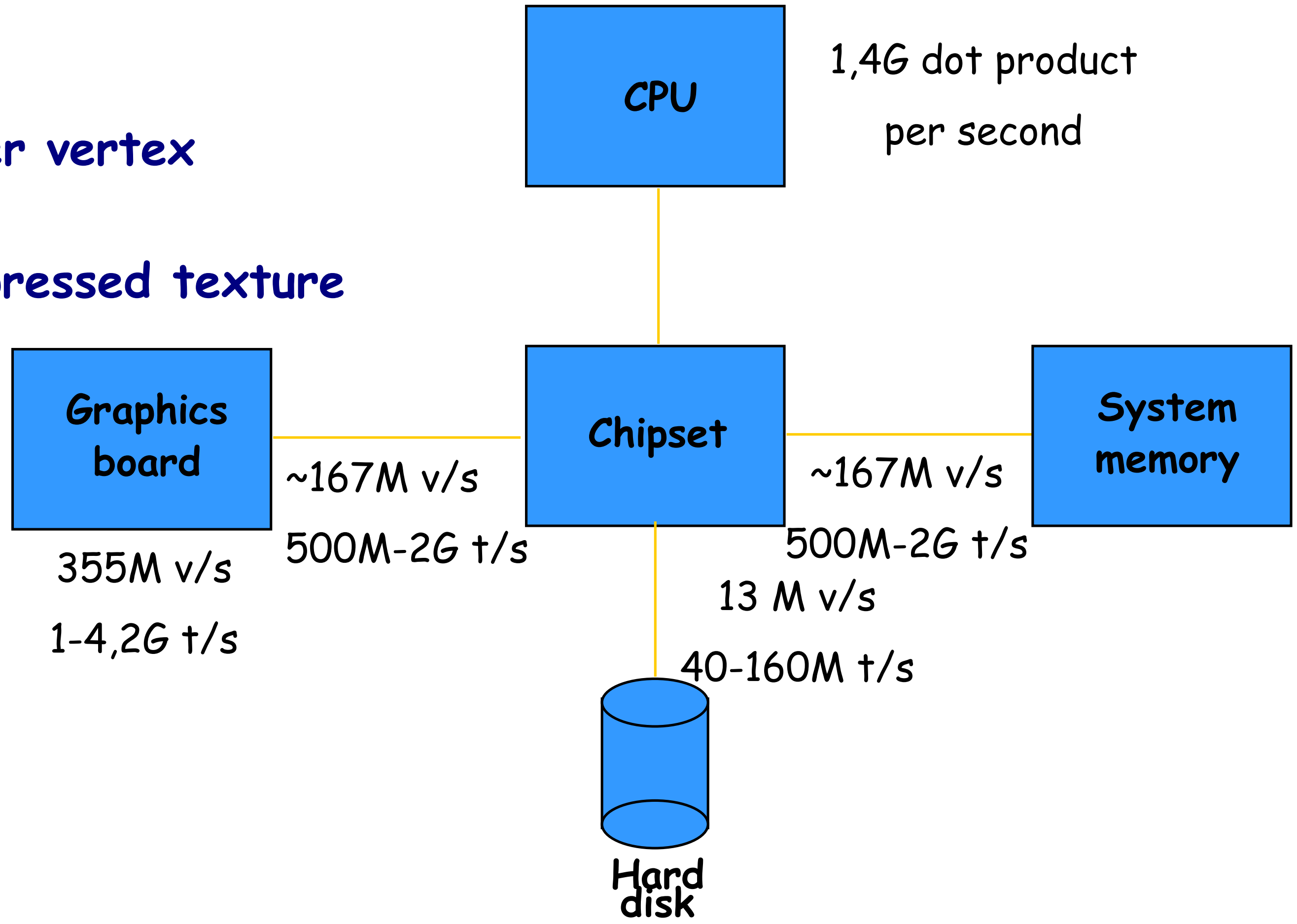
# PC Hardware overview

- High end PC (2002)

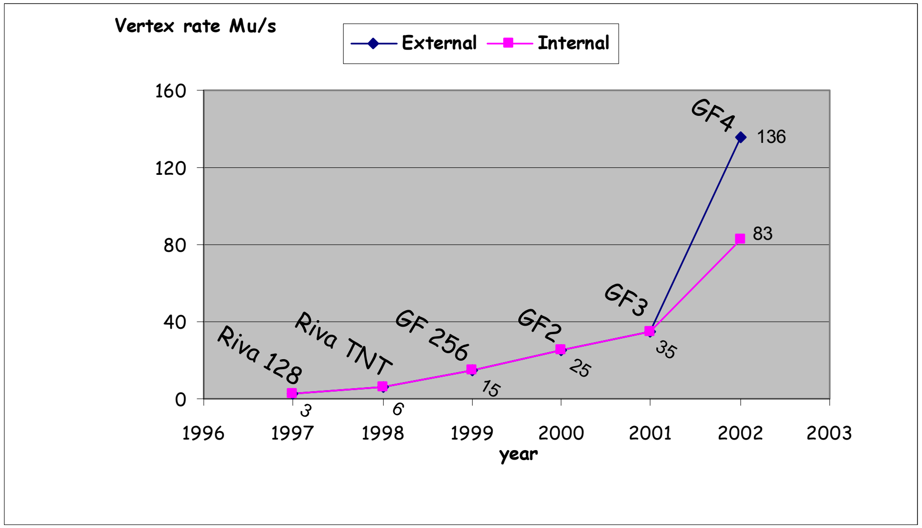


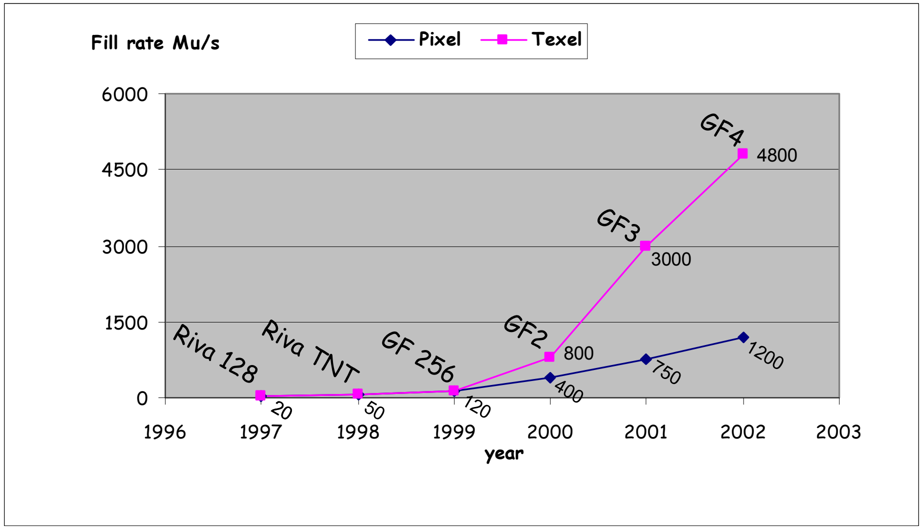
# Data transfer peak rates

**vertex rate with:**
**3 fp coordinates per vertex**
**texel rate with:**
**uncompressed-compressed texture**



# Rendering speed increase (1996-2002)




- Expected high end PC performance at the end of 2004 (educated guesses):
  - ~ 160M vertices/s (external, AGP8x)
  - ~ 340M vertices/s (internal)
  - ~ 2200M pixels/s.

# The goal

Rapid processing and interactive rendering of complex 3D scenes with high visual and temporal fidelity on a graphics PC platform

# Scene preprocessing

- Many types of possible operations
  - Dataset extraction (e.g. isosurfaces, reconstruction from scans, …)
  - Pre-processing to optimize rendering speed
  - Global illumination computation
  - …
- Rapid processing implies scalable algorithms
  - Time is is a soft constraint: we look for max O(N) processing times to ensure scalability
  - Memory is a hard constraint: we require bounded memory requirements for all operations

# Minimum storage requirements

- An “average complex scene” with ~200 M vertices requ
  - ~ al (3f), c
  - ~ mation (3

**4GB main memory limitation imposes out-of-core + compression methods!**

# The goal

Rapid processing and interactive rendering of complex 3D scenes with high visual and temporal fidelity on a graphics PC platform

# Visual Fidelity

- Should match human perceptual capabilities…
  - FOV, depth perception, high dynamic range, CSF, adaptation…
- … but taking into account display device constraints
  - ~1M pixels/frame
  - low dynamic range colors (24bpp, low display luminance scale)

# Temporal Fidelity

- High frame rate
  - 10 fps (minimum to achieve illusion of animation)
  - 60-100 fps ("high speed" simulators)
- Low latency
  - >100 ms: begin degradation of human performance
  - >300 ms: begin cause-effect dissociation

# Memory and time are limited for rendering!

- Brute force point rendering of a 200M samples scene at ~
  - ~
  - ~
  - ~
- One … at we can … ne!

**Timing constraints impose methods for trading rendering quality with speed!**

# Conclusions

- Hardware performance largely insufficient to handle real-world (2003) datasets by brute force methods in the foreseeable future
  - 1-2 orders of magnitude mismatch just for the plain rendering operation
- Rendering optimizations are required to meet performance constraints
  - Occlusion/View Culling => insufficient, since scene complexity from a given viewpoint is potentially unbounded
- Need to trade rendering quality with speed
  - Simplification and multiresolution data representations
  - Error metrics for measuring error degradations
  - Cost models to predict rendering performance
  - Time-critical rendering algorithms for choosing levels of detail

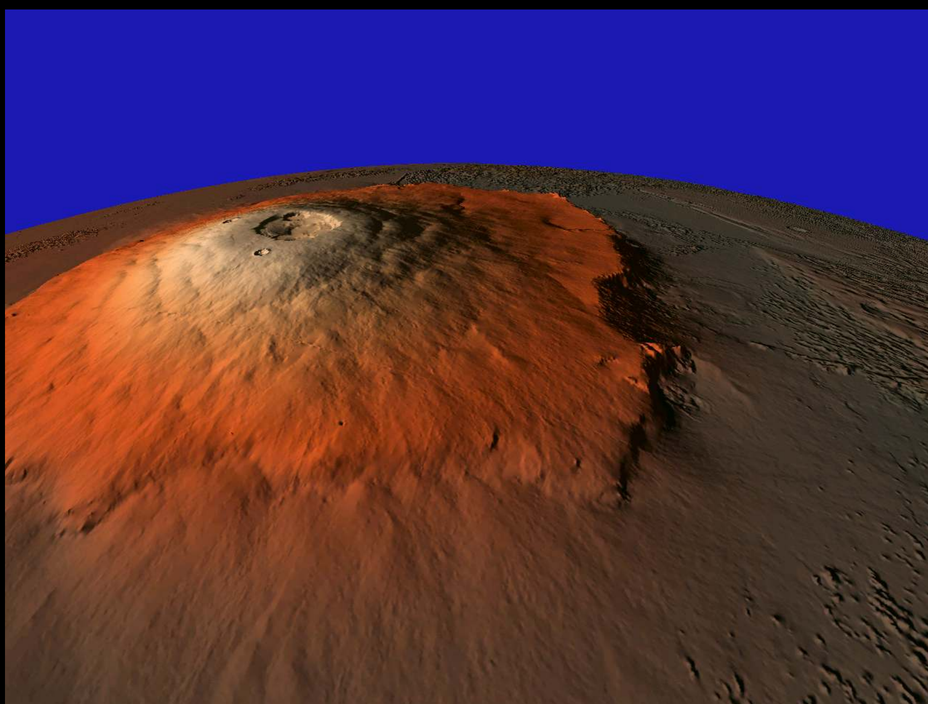
# Hints

- Dataset size potentially exceeds core memory limits
  - ⇒ Use out of-core techniques
  - ⇒ Use compression techniques
  - ⇒ Work around 32 bit architectures limitations (4Gb memory limit)

⇒ Transfer rates (PCI,FSB,AGP) are the limiting factors
  - ⇒ Manage geometry/texture as bandwidth-limited resources
  - ⇒ Use compression techniques

⇒ Internal GPU speed >> CPU/AGP speed
  - ⇒ Favor display lists wrt. immediate mode graphics
  - ⇒ Manage geometry/textures by blocks
  - ⇒ Use programmability features to offload CPU and reduce host-graphics communication needs

# PBDAM: Planet-Sized Batched Dynamic Adaptive Meshes



IEEE VISUALIZATION 2003

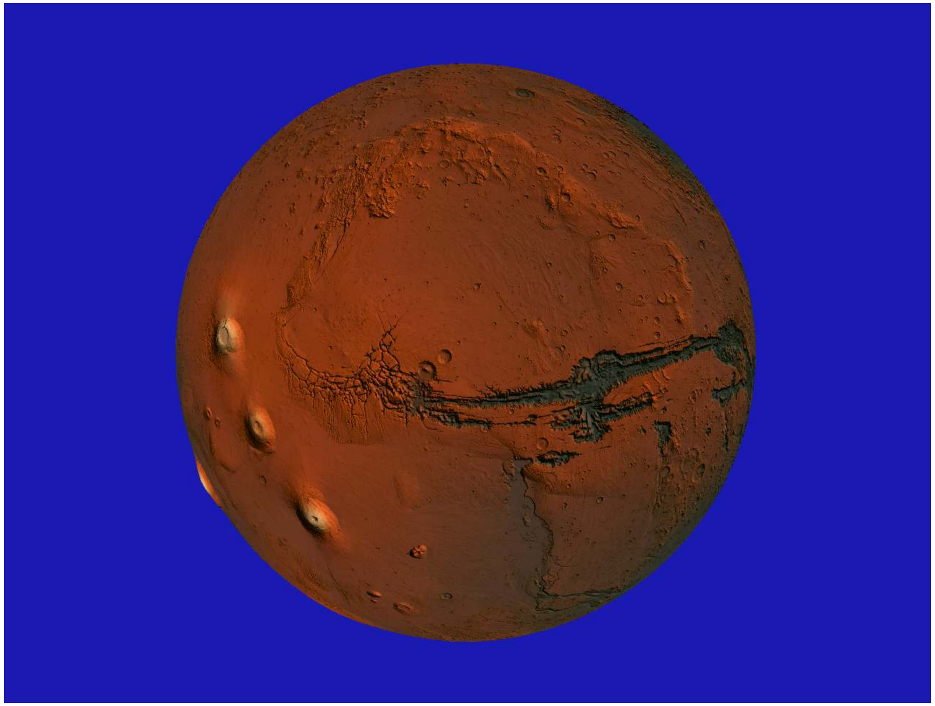
Enrico Gobbetti
Fabio Marton
CRS4 - Visual Computing Group (Italy)

Paolo Cignoni
Fabio Ganovelli
Federico Ponchio
Roberto Scopigno
ISTI-CNR Visual Computing Group (Italy)

*October 22th, 2003*

# The Domain

- Rendering of detailed large scale (full planet) textured terrain datasets at interactive frame rates on PC platforms.



# The Domain

- Rendering of detailed large scale (full planet) textured terrain datasets at interactive frame rates on PC platforms.

| | |
|---|---|
| •Terrain geometry: | NASA MOLA MEGDR 1/128 (1 G samples) |
| •Terrain texture: | Shaded Relief (1.5 G Texels) |
| •Compressed data size: | 5.7 GB |
| •Window size: | 800 x 600 |
| •Screen tolerance: | 1 pixel |

# Previous work *(really short overview)*

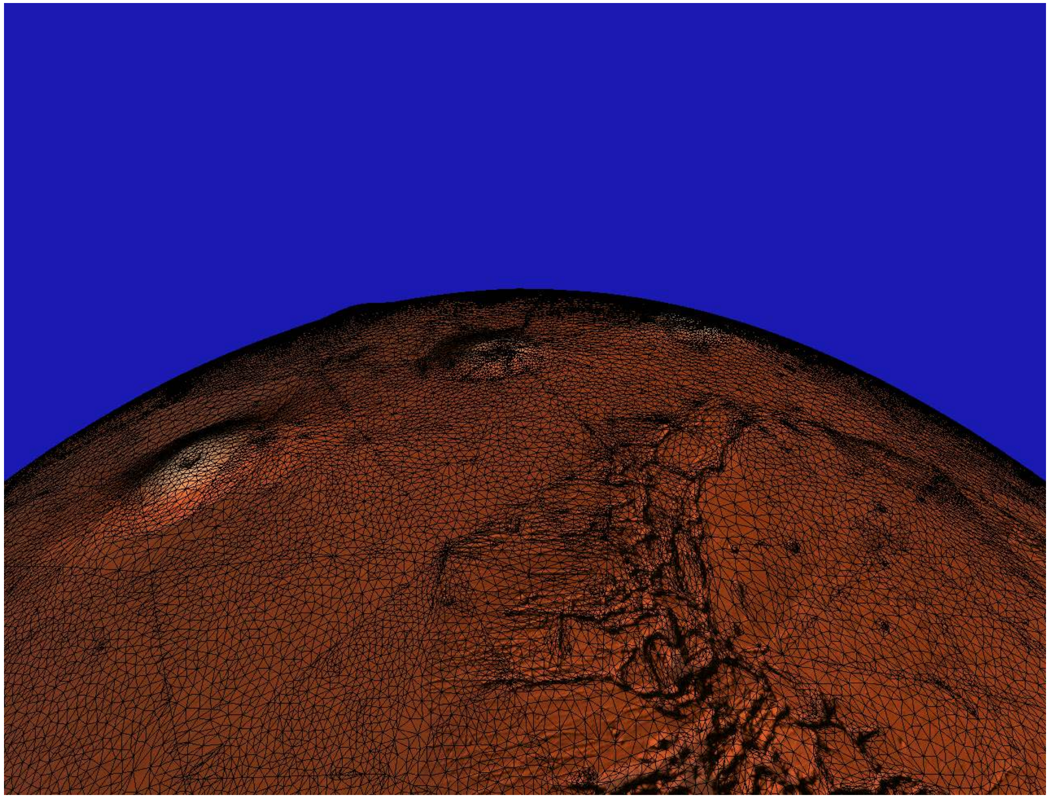
- Regular mesh refinement
  - Triangle Bintree ROAM [Duchaineau 1997 ]
  - Longest Edge Bisection SOAR [Lindstrom, Pascucci 2002]
  - ....
- Irregular mesh refinement
  - Triangulated Irregular Network [Puppo 1996]
  - Hypertriangulations [Cignoni 1997]
  - View Dependent Progressive Meshes [Hoppe 1997]
- Block based rendering
  - Digital Earth in VRML [Reddy 1999]

# Previous Works

| | Regular mesh refinement | Irregular mesh refinement | Block based rendering |
|---|---|---|---|
| Accuracy | Good with high tri count, but single precision limitations | Best with a given tricount , but single precision limitations | Low |
| Size and scale | 4GB limit, but efficient out-of-core techniques | 4GB limit, out-of-core techniques hard to implement | Efficient paging, possible problems w/ curved datasets |
| Bandwidth | Fast, but CPU bound | Slow | Fast |
| Continuity | Yes, except for tiling | Yes, except for tiling | No |
| Texturing | Simple parameterization | Hard | Simple parameterization |

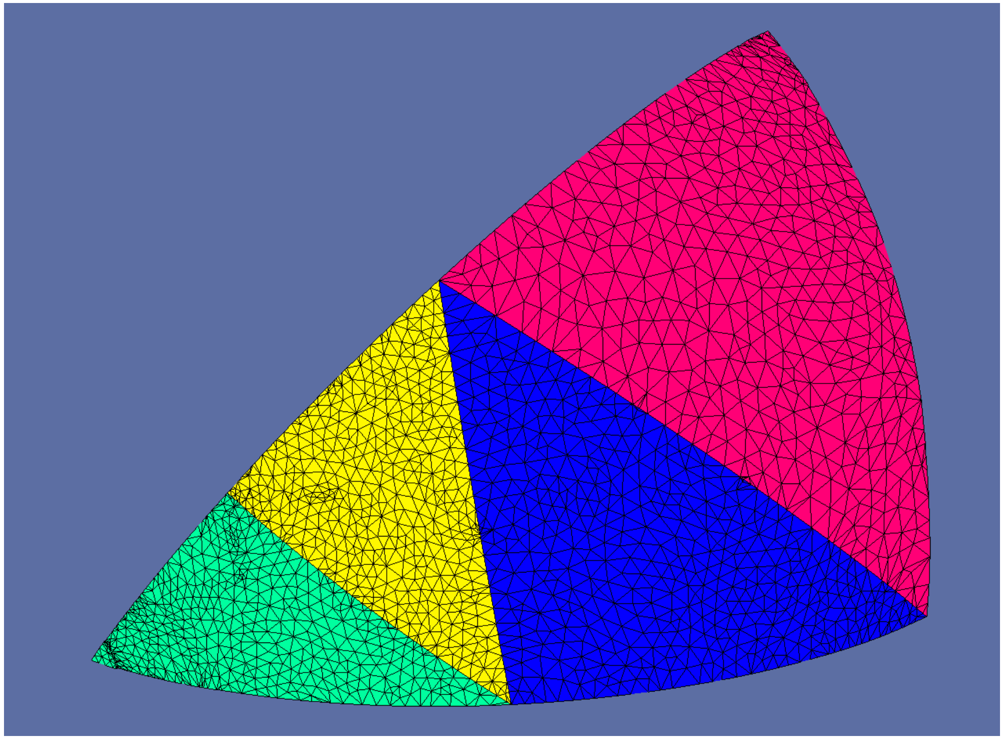
# The Claim:

- By combining
  - rough regular subdivision,
  - Triangulated Irregular Networks
  - GPU Programming
- We can solve accuracy, size, bandwidth, continuity, texturing problems.



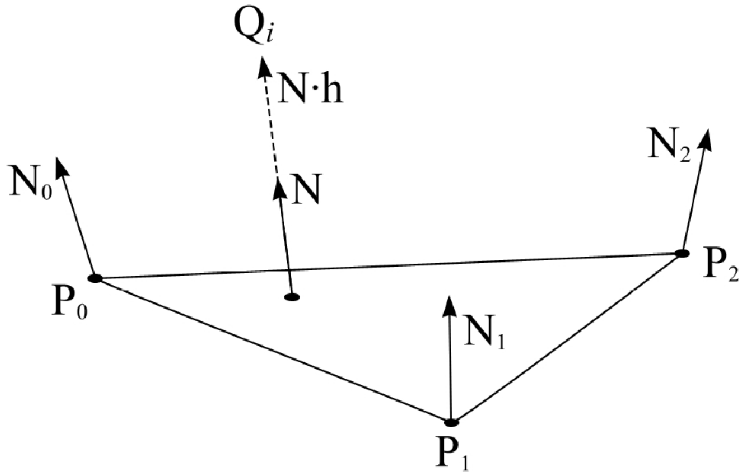
# Geometric Primitive: mesh of triangles

- Curved Surface Triangular Patch:
  - Mesh of triangles hi-quality adaptively simplified + stripified during preprocessing.
  - Take into account planet curvature.
  - Allow fast CPU-GPU communication through Opengl Vertex Array Range.
  - Preserve connectivity among adjacent levels



# Geometric Primitive: Displaced Triangle

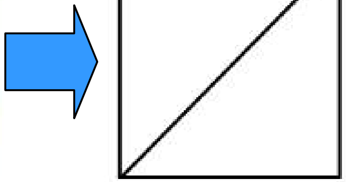
- 3 Corners Coordinates:
  - Stored in double precision.
- Internal vertices :
  - Barycentric coordinates.
  - 4 short per vertex.
  - Implicit u, v texture coordinates.
  - Extracted with linear interpolation exploiting GPU programming.
- Representation pros:
  - Compact
  - Optimized
  - Cache coherent
  - Preserve Continuity



# Texture Primitive

- Texture square tile
  - Easily mapped to geometry through Opengl
- Geometry Correspondence
  - One texture tile covers 2 triangular geometry patches
- DXT1 Compression
  - Allow compression ratio 1:6, 1:8

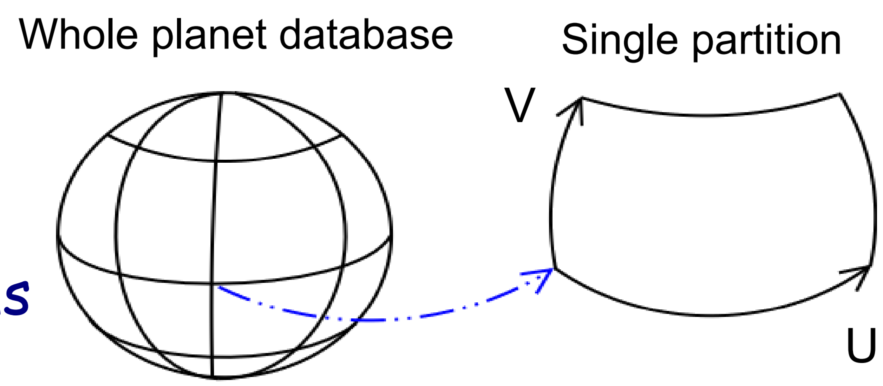
Texture Tile



Pair of geometry tiles

# Terrain Partitioning

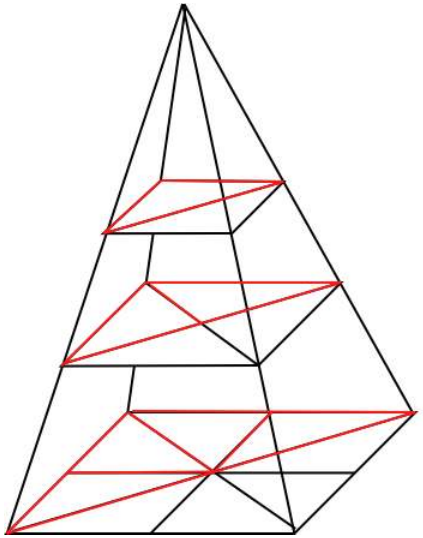
- Size + Accuracy + Continuity problems



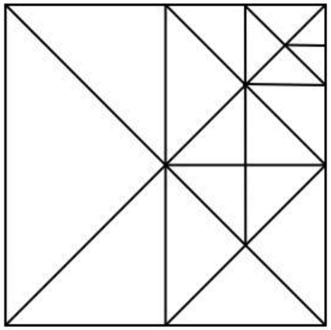

- Terrain is subdivided into manageable continuous partitions with respect to a parametric coordinate systems
- Each partition geometry is expressed with respect to a local parameterization
- Rendering is performed in view coordinates (single precision fp enough), with conversion done on the GPU

# Geometry Multiresolution

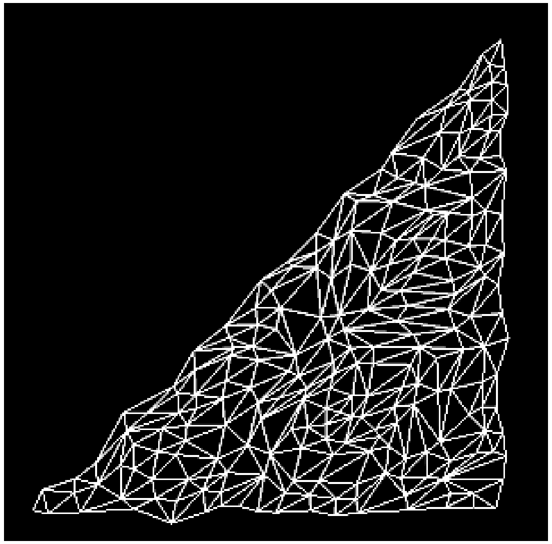
- 2 Bintrees of triangular patches.
- Triangle split along longest edge.
- Allow view dependent continuous multiresolution subdivision
- Each triangular patch is a mesh



Pyramid of geometry



Subdivision example



Mesh of a single patch

# Texture Multiresolution

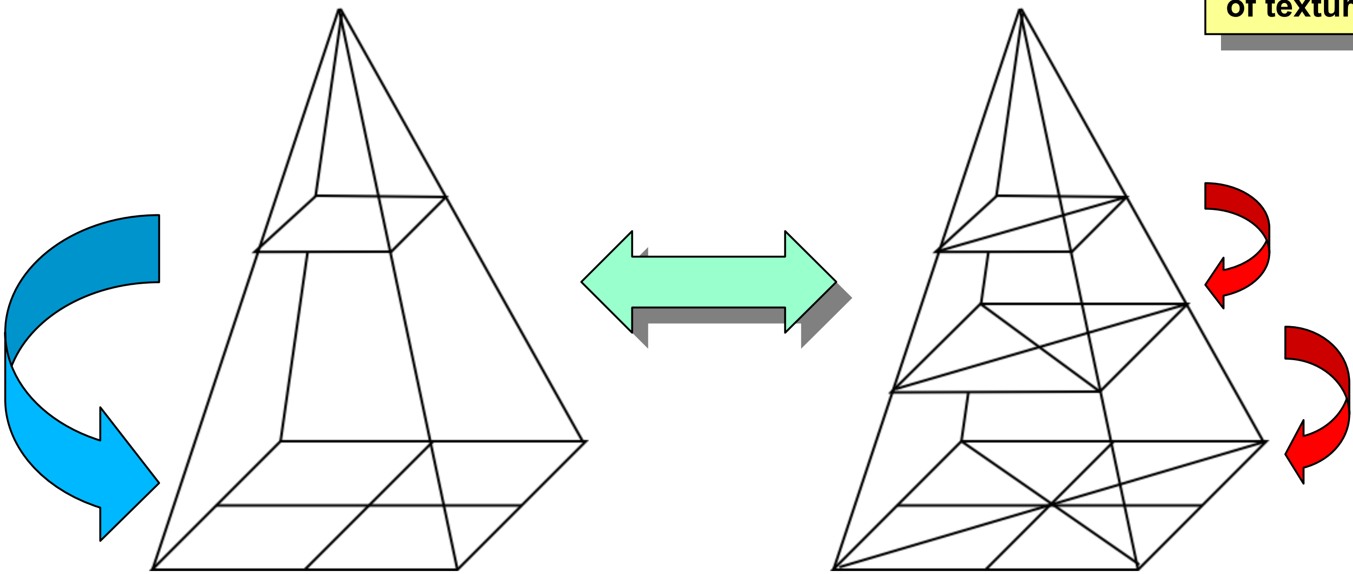
- Texture is organized in a quadtree of tiles.
- Each tile is subdivided into 4 children with double res[...]
y.

What happens when we subdivide a texture tile:

A refinement step in texture is equivalent to 2 step in geometry.

Two levels of geometry are covered by one level of texture.

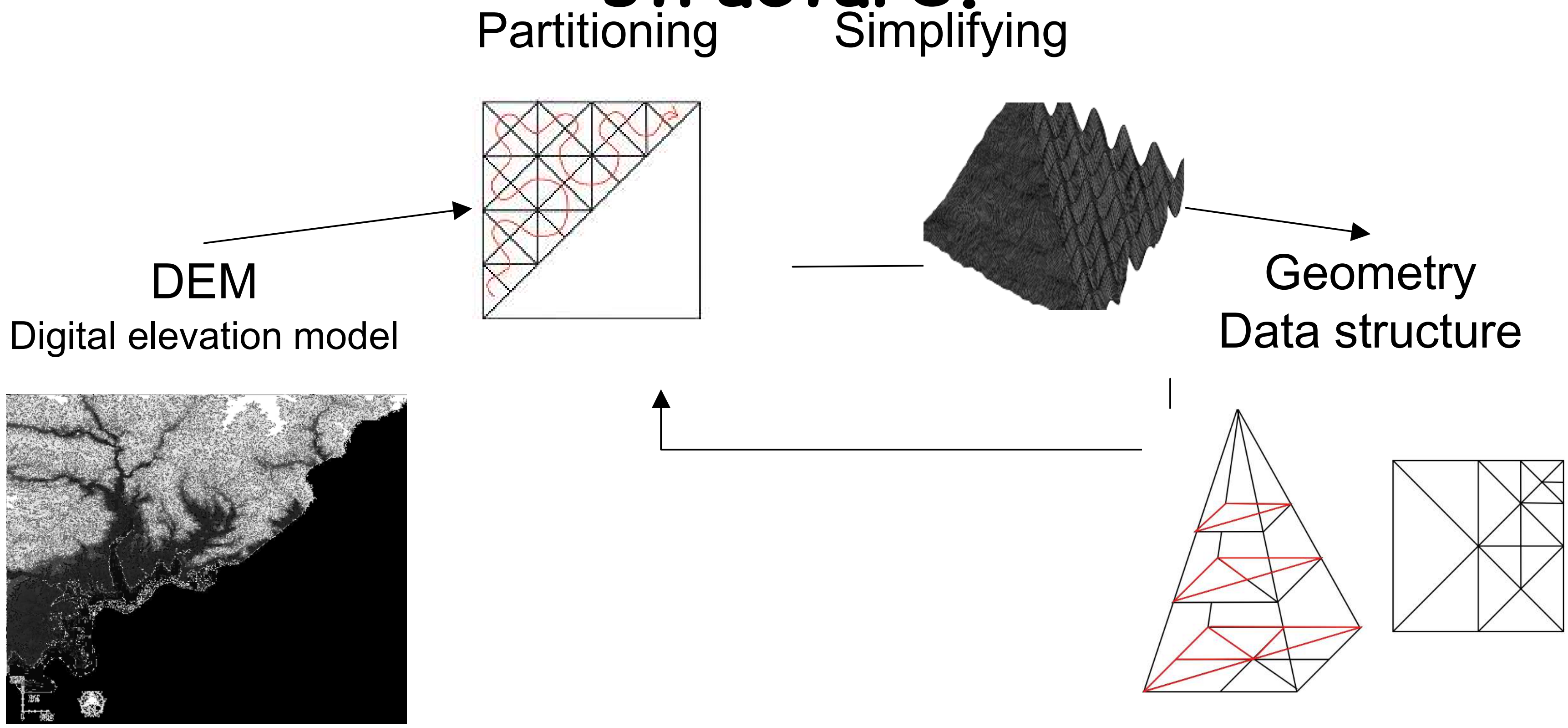
Texture - Geometry trees correspondence:



Texture tree

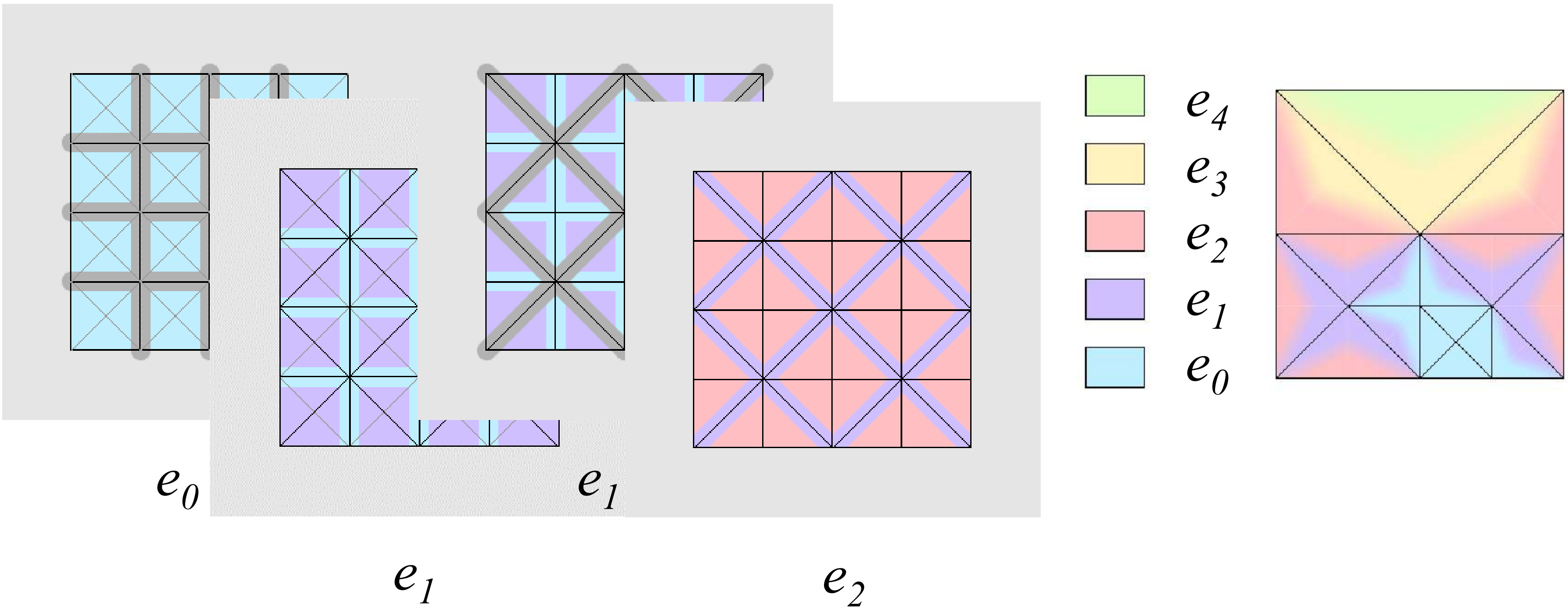
Geometry tree

# Construction of the geometry data structure.



# Mark and simplify

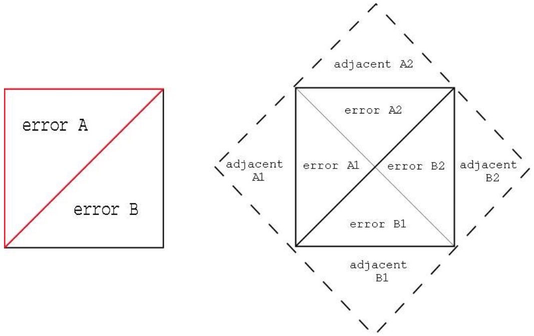
- BDAM are built by a sequence of:
  - mark boundary
  - simplify non marked areas
  - store resulting patches
- Process 4 adjacent tiles at once
- Border vertices duplicated and explicitly indexed

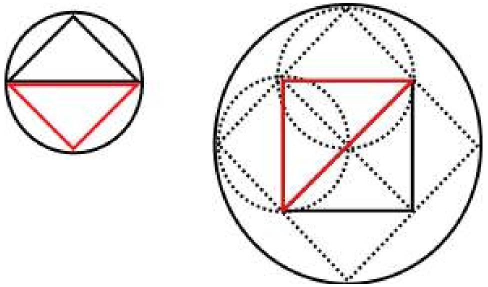


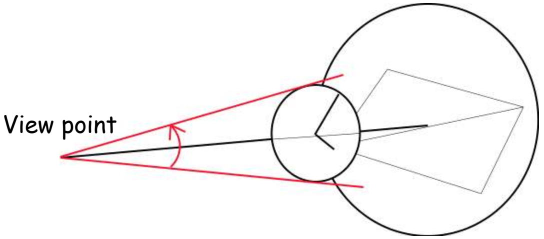
# Continuity

- Dependencies implicitly encoded in hierarchies of nested errors and bounding volumes.
  - Adjacent triangle patches along hypotenuse share same *value*
  - Patch *value* enclose children *values*.

- Embedded screen space error
  - Computed projecting maximum of texture and geometry error from the embedded bounding sphere.

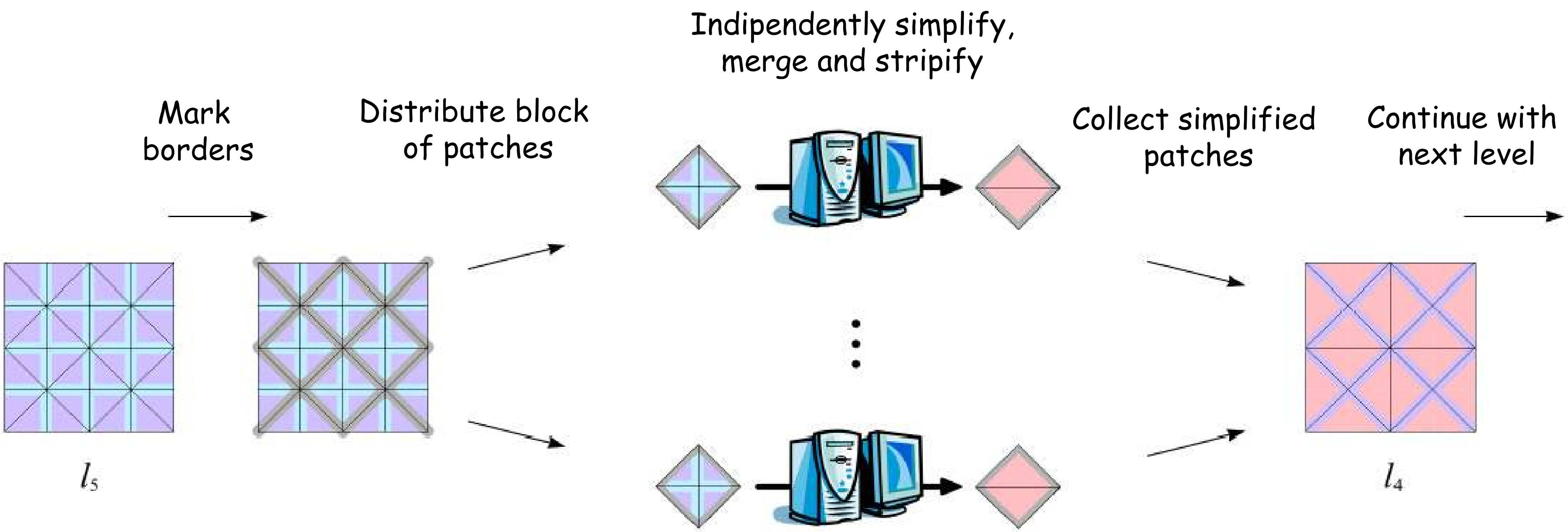




# Parallel simplification

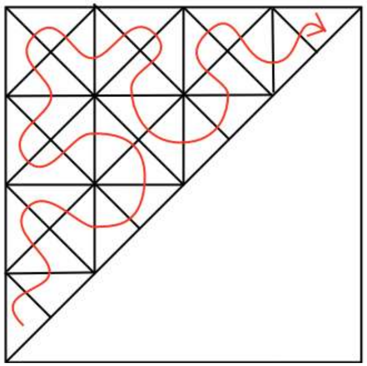
- Use network of PCs to perform simplification quickly.



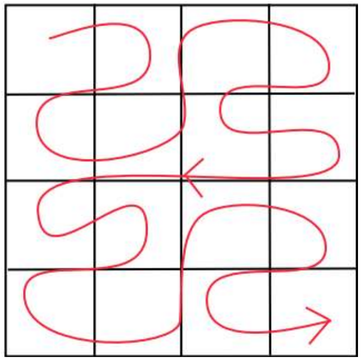

- Similar approach is used for the texture quadtree.

# On Disk Representation

- Out-of-core data management through system memory mapping functions.
  - Geometry and texture data accessed through easy indices computation.
  - Memory order reflects phisical position to minimize the number of page faults using two filling curves.
- Geometry patch compression
  - Delta encoding and LZO compression are applied to each single patch to achieve
    - ~50% size reduction
    - run-time fast decompression.



Geometry filling curve



Texture filling curve

# One Pass Rendering

- In one pass:
  1. Perform view frustum culling.
  2. Descend geometry and texture trees choosing proper texture.
  3. Further refine geometry.
  4. Generate view-dependent patch corner coordinates
  5. Draw texture mapped geometry, converting parametric representation to view coordinates on the GPU.
  6. Manage created geometry and texture objects through a Least Recently Used (LRU) strategy
- One pass is used to exploit CPU and GPU parallelism:
  - While CPU descends the 2 trees, it sends chosen tiles to the GPU, because of the size of a single tile GPU never

# Prefetch

- Perform one prefetch data traversal to diminish access disk delays
  - Traversal similar to rendering but does not send anything to GPU.
  - Touch patches with asynchronous calls *memadivse*
  - Prevision is made with linear interpolation on current path.
  - SCSI disks strongly reduce delay times.

# Partition continuity

- Continuity among adjacent partitions is obtained during rendering, exploiting :
  - Overlapping bounding volumes on the edges of adjacent partitions.
  - Embedded error hierarchies that consider also errors of patches of neighboring partitions.

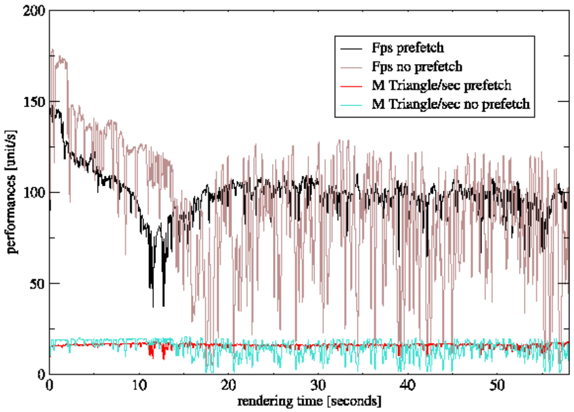
- Rendering can be done independently for each partition, because errors and bounding volumes have been embedded in the preprocessing step.

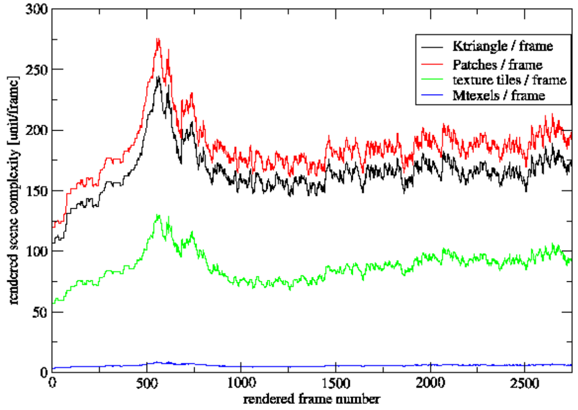
# Results

| PREPROCESSING | Original dataset | P-BDAM Compressed size | Preprocessing time |
|---|---|---|---|
| Geometry | 44K x 22K<br>1 G samples | 4.5 GB | 6.10 h |
| Texture | 1.5 G texels | 1.2 GB | 1 h |
| RENDERING | Mean | Peak | |
| Fps | 90 | 130 | |
| Tri / sec | 16 M | 18.5 M | |

Virtual fly over planet Mars.

Results obtained on an AMD Athlon MP 1900+, 1600 MHz with NVIDIA GeForce 4 Ti 4600 / AGP4X



# Future Works…

- What about 3D models ?

# Hierarchical Higher Order Face Cluster Radiosity for Global Illumination Walkthroughs of Complex Non-diffuse Environments

EUROGRAPHICS 2003

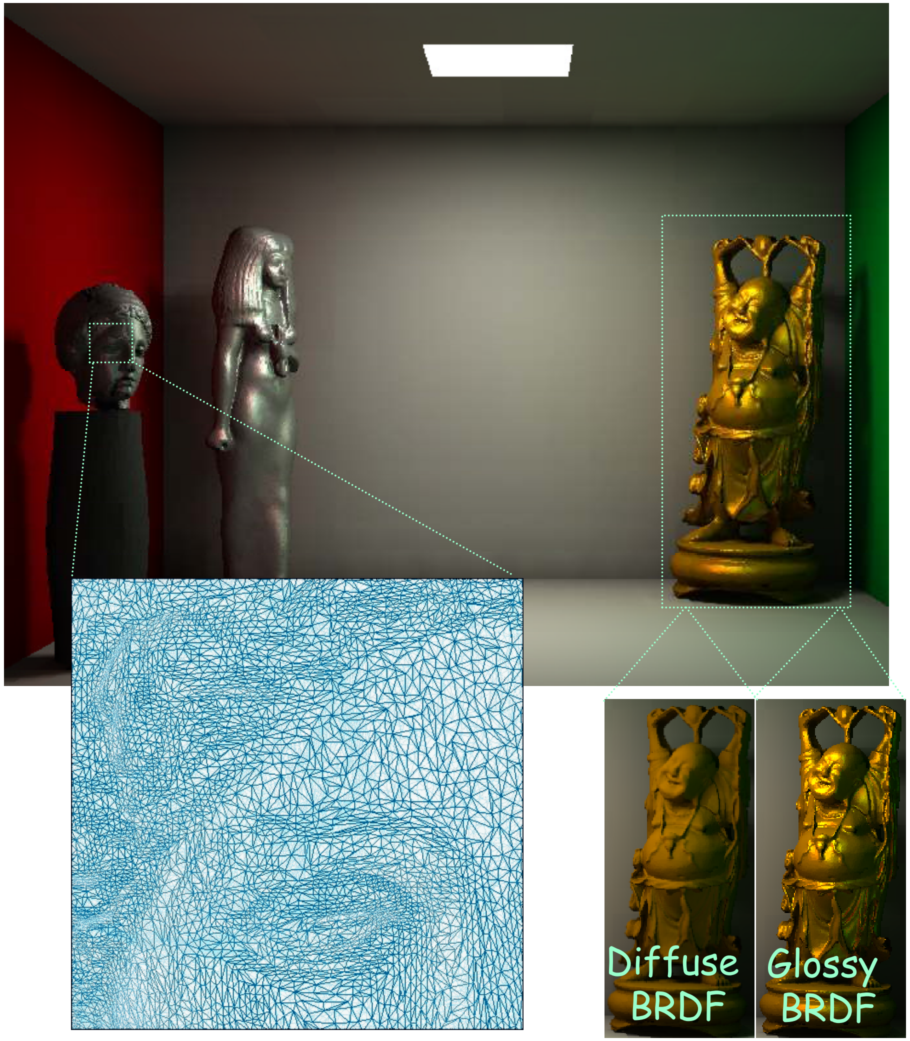
Enrico Gobbetti
Leonardo Spanò
Marco Agus
CRS4 - Visual Computing Group
Italy

# The Domain

- Radiosity on scenes with detailed polygonal models and non-diffuse materials

# Motivation

- Radiosity is a de facto industrial standard
  - Efficient for common diffuse-only / flat walls scenes
  - Blends well with walkthru applications and FEM analysis tools
- Detailed polygonal models (>> 100K faces) are increasingly common
  - 3D Scanning + Tessellated CAD models
- View-dependent lighting effects important for appreciating surface finish
  - Arbitrary BRDF

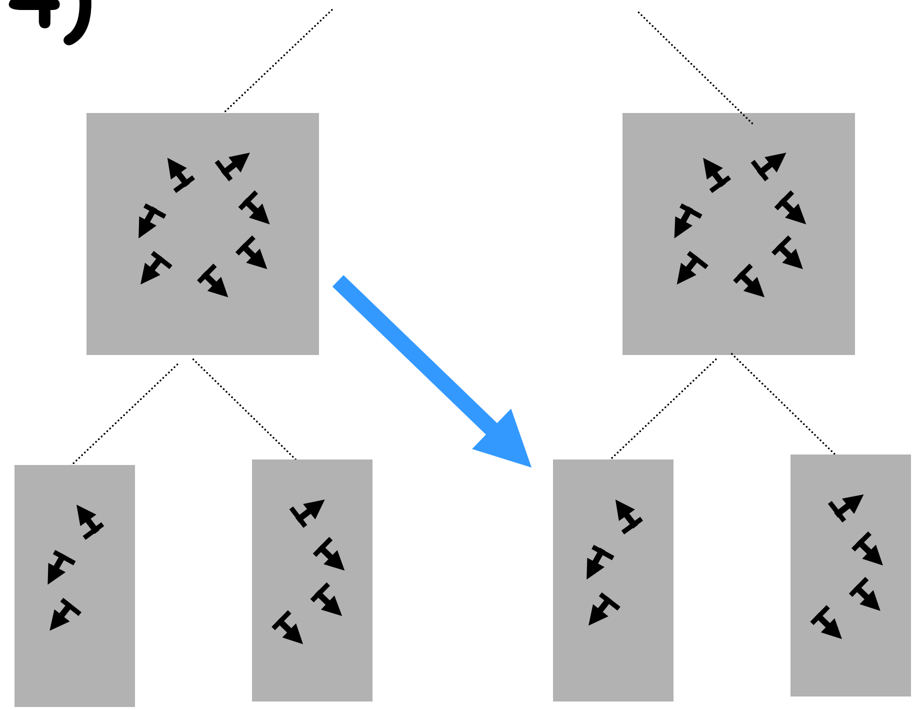


# The Claim

- By combining face clustering, higher order vector radiosity, and GPU programming techniques we can
  - Better approximate detailed model surfaces
  - Get sub-linear (constant) solution time/memory complexity
  - Roughly approximate non-diffuse BRDFs
  - Interactively inspect view-dependent solutions on standard commodity graphics platforms

# State-of-the-art (1/4)

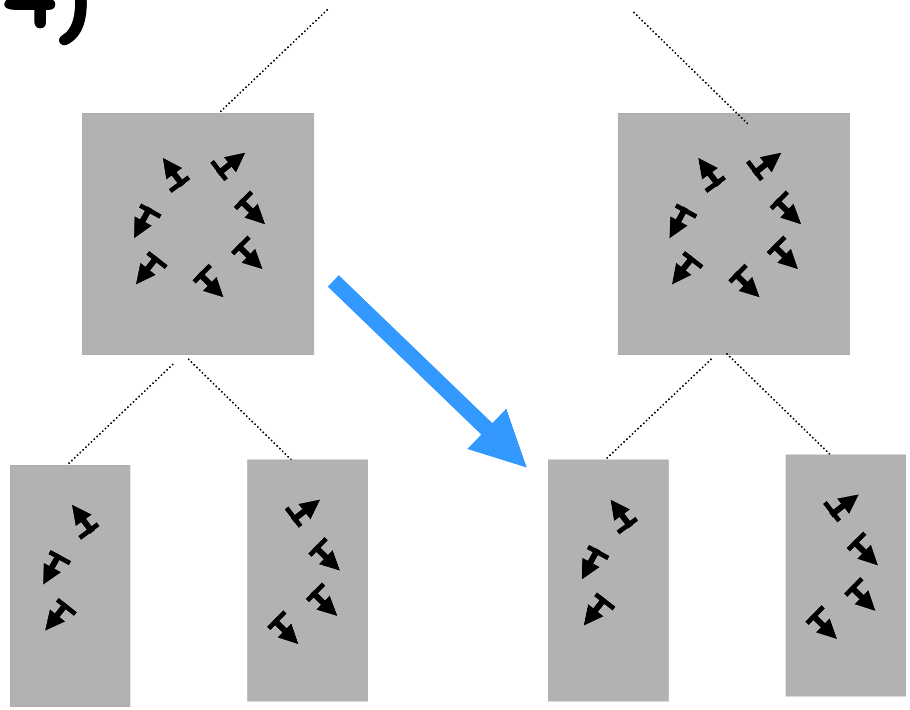
- Hierarchical Radiosity with Volume Clustering [Smits94, Sillon96, …]
  - Constructs a complete scene hierarchy above input polygons (preprocessing)
    - Volume clusters approximate a cloud of unconnected polygons
  - Handles multiresolution light transfers
    - Complexity is O(klogk+n)



$B_i = E_i + \rho_i B_j F_{ij}$

# State-of-the-art (2/4)

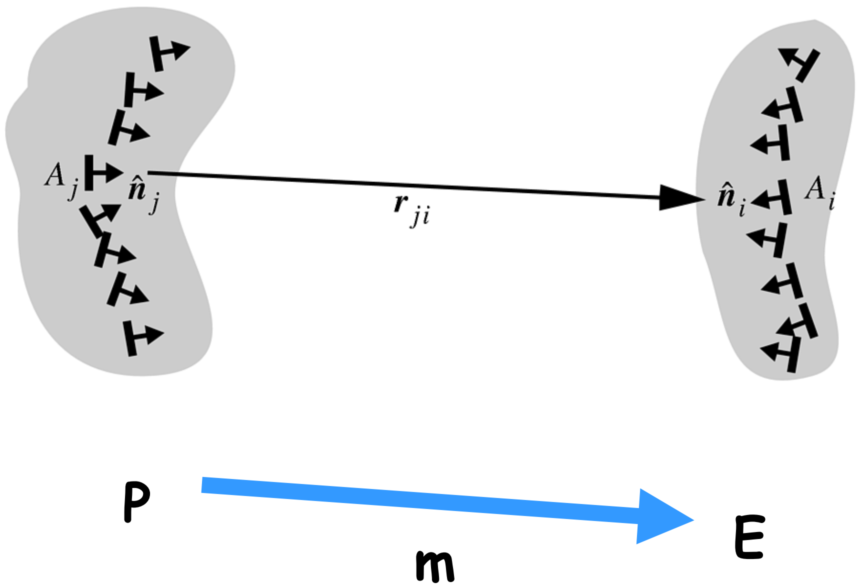
- Hierarchical Radiosity with Volume Clustering [Smits94, Sillon96, …]
  - O(klogk+n) complexity is a problem for complex scenes
    - Touches all input polygons at least at each iteration (push irradiance/pull radiosity)
  - Smoothing is difficult
    - Higher order solution representation hard (illuminated connected surfaces appear "blocky")
  - Interactive display for non-diffuse BRDF is difficult



$$B_i = E_i + \rho_i B_j F_{ij}$$

# State-of-the-art (3/4)

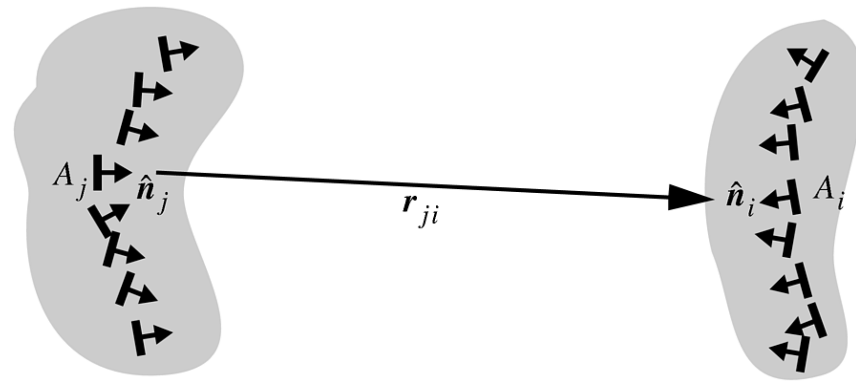
- Hierarchical Radiosity with Face Clustering [Willmott99]
  - Clusters of coplanar polygons instead of volume clusters
  - Recasts radiosity equation in terms of irradiance vector and power vector
    - Simplest representation of irradiance vector field
  - Combines vectors hierarchically to represent complex irradiance distributions

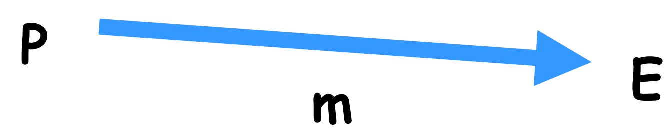



$$E_i = -m_{ij} m_{ij}^T P_j$$

# State-of-the-art (4/4)

- Hierarchical Radiosity with Face Clustering [Willmott99]
  - Sub-linear complexity - much faster for complex scenes
    - Solution complexity depends only on irradiance vector field complexity
    - Avoids push-to-leaves
  - Solutions are still “blocky”, smoothing requires a post-pass
  - Still limited to diffuse-only BRDF



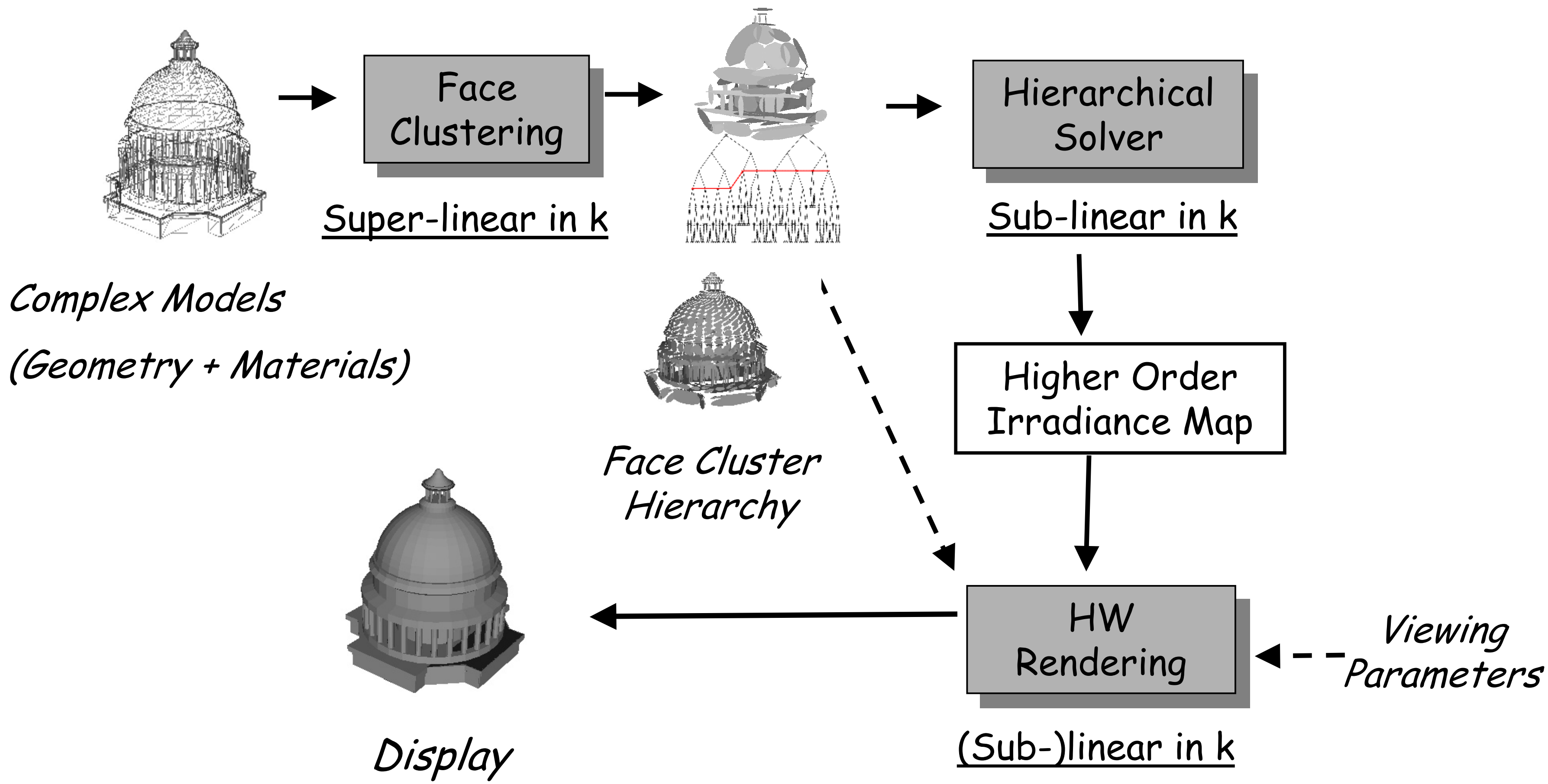




$$E_i = -m_{ij} m_{ij}^T P_j$$

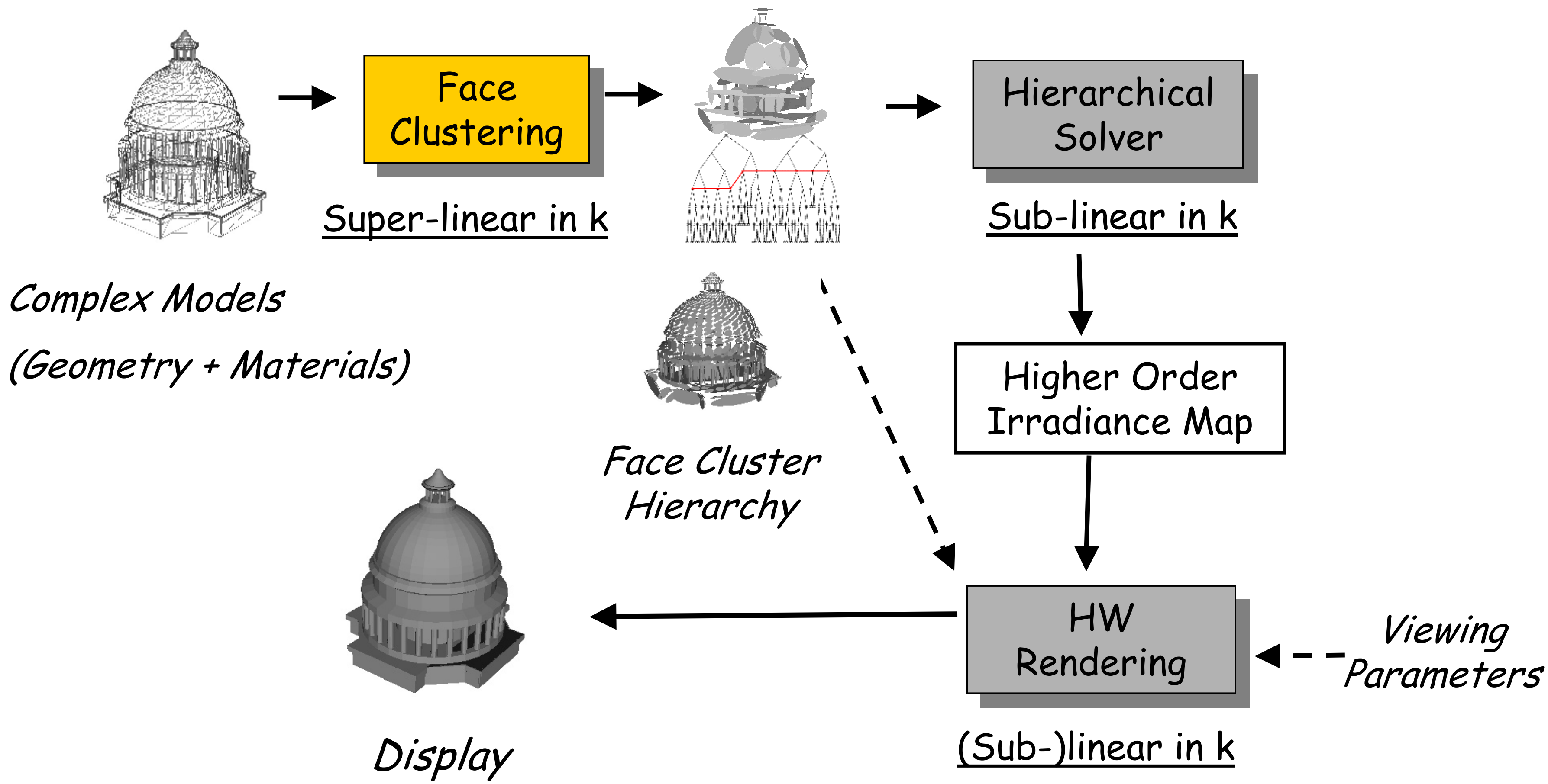
# Our contribution

- Solve higher order vector radiosity equations with limited time/memory budget
  - Extend face clusters to higher order bases (smoothing, error control)
  - Modified shooting solution method reorders computations to minimize memory
  - Result is a visually smooth vector irradiance field
- Rapidly display view-dependent solutions using commodity graphics hardware
  - Extract per vertex radiance from vector irradiance field and full BRDF at frame rendering time
  - Fully computed on the GPU using a vertex program

# Method overview

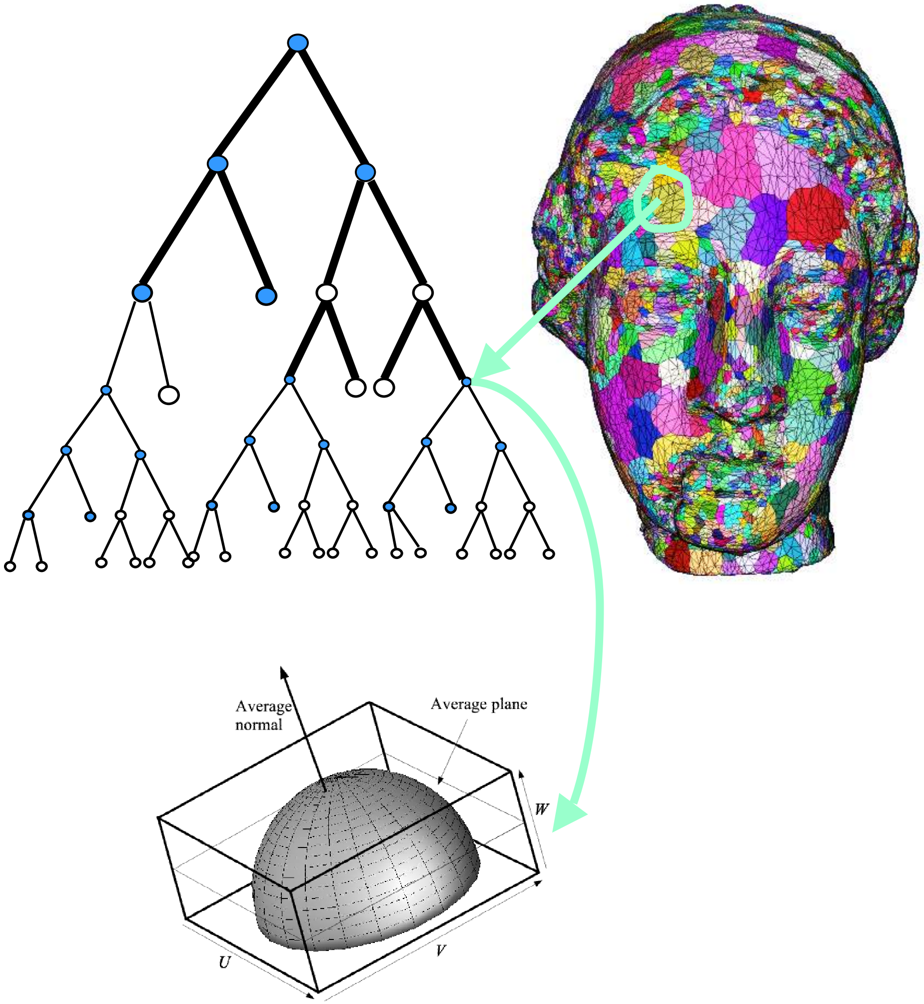


# Method overview

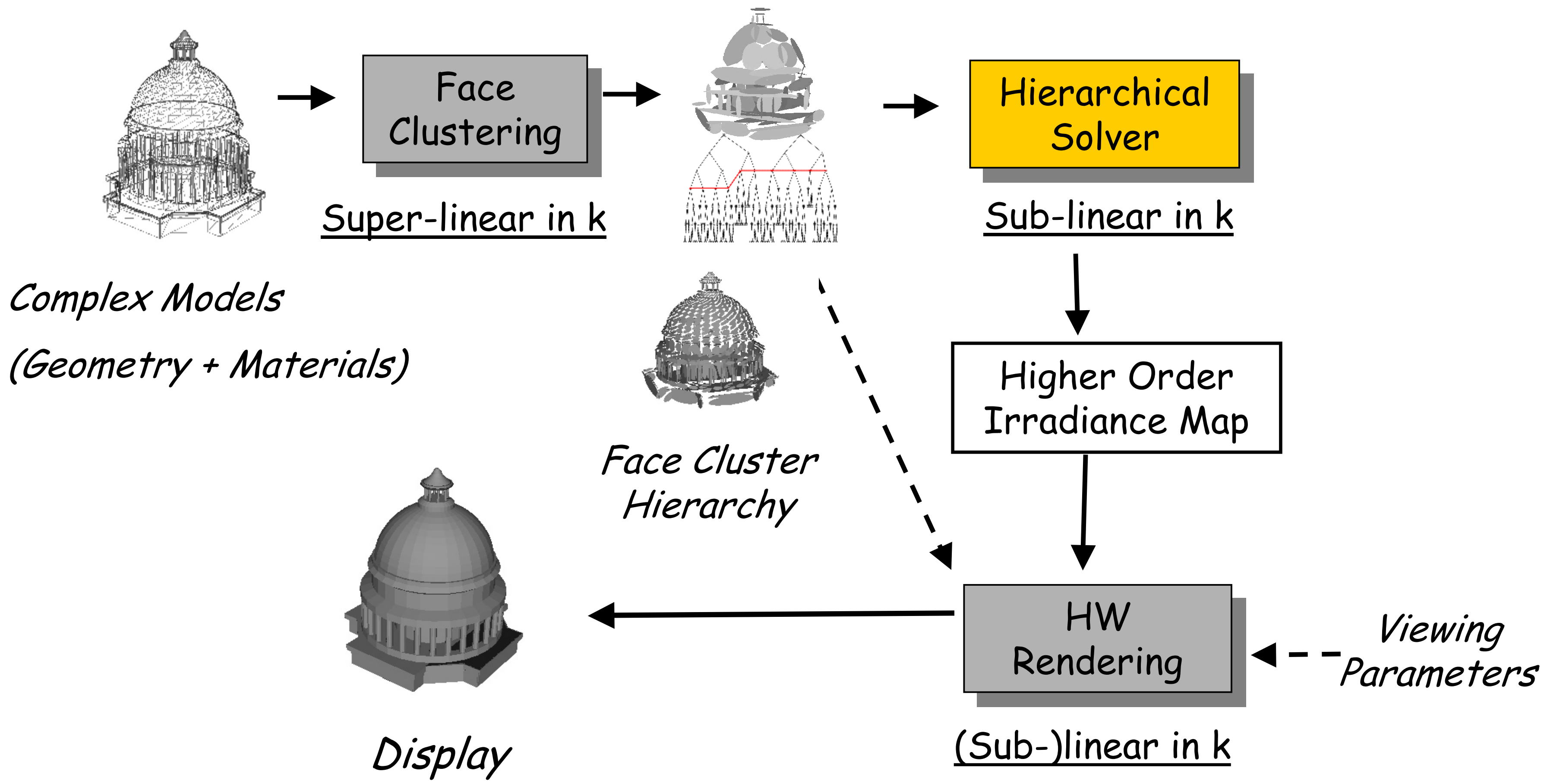


# Face clustering

- On an object-by-object basis:
  - Hierarchically group together connected faces
    - Planarity + attribute similarity criterion [Garland01+attributes]
  - Parameterize cluster
    - u,v axis on average plane, oriented as minimum area enclosing rectangle
    - w axis aligned with average normal
  - Pre-compute constants for quickly answering geometric/attribute queries
    - Min/avg/max projected areas, self-form factor, normal bounds, reflectance/emission coefficients
  - Store result in a cluster file



# Method overview



# Higher order vector radiosity (overview)

- Start with full rendering equation
- Use face cluster radiosity approximation for overall energy distribution
- Project onto cluster basis functions and transform to linear system (Galerkin method)
- Solve for vector irradiance

# Higher order vector radiosity (1/6)

- Start with familiar rendering equation

$$L(\mathbf{x},\mathbf{z}) = L_e(\mathbf{x},\mathbf{z}) + \int_A L(\mathbf{y},\mathbf{x}) f_r(\mathbf{x},\mathbf{y},\mathbf{z}) V(\mathbf{x},\mathbf{y}) G(\mathbf{x},\mathbf{y}) dAy$$

$$G(\mathbf{x},\mathbf{y}) = \frac{\left((\mathbf{y}-\mathbf{x})\cdot\mathbf{n}_x\right)_+ \left((\mathbf{x}-\mathbf{y})\cdot\mathbf{n}_y\right)_+}{\pi\left\|\mathbf{y}-\mathbf{x}\right\|^4}$$

# Higher order vector radiosity (2/6)

- Radiosity approximation:

$$L(\mathbf{x},\mathbf{z}) = L_e(\mathbf{x},\mathbf{z}) + \int_A L(\mathbf{y},\mathbf{x}) f_r(\mathbf{x},\mathbf{y},\mathbf{z}) V(\mathbf{x},\mathbf{y}) G(\mathbf{x},\mathbf{y}) dAy$$

$$B(\mathbf{x}) = B^e(\mathbf{x}) + \rho(\mathbf{x}) \int_A B(\mathbf{y}) V(\mathbf{x},\mathbf{y}) G(\mathbf{x},\mathbf{y}) dAy$$

(Assumes that overall energy distribution is well approximated by uniform emitters/receivers – OK for "moderately glossy" objects)

# Higher order vector radiosity (3/6)

- Vector radiosity representation:

$$B(\mathbf{x}) = B^e(\mathbf{x}) + \rho(\mathbf{x}) \int_A B(\mathbf{y}) V(\mathbf{x},\mathbf{y}) G(\mathbf{x},\mathbf{y}) dAy$$

$$B(\mathbf{x}) = B^e(\mathbf{x}) + \rho(\mathbf{x}) \int_A \left(\mathbf{n}_x \cdot \mathbf{E}(\mathbf{x},\mathbf{y})\right)_+ dAy$$

$$\mathbf{E}(\mathbf{x},\mathbf{y}) = \mathbf{m}(\mathbf{x},\mathbf{y}) B(\mathbf{y})$$

$$\mathbf{m}(\mathbf{x},\mathbf{y}) = V(\mathbf{x},\mathbf{y}) \frac{((\mathbf{x}-\mathbf{y}) \cdot \mathbf{n}_y)_+}{\pi \left\|\mathbf{y}-\mathbf{x}\right\|^4} (\mathbf{y}-\mathbf{x})$$

# Higher order vector radiosity (4/6)

- Face cluster approximation:

$$B(\mathbf{x}) = B^e(\mathbf{x}) + \rho(\mathbf{x}) \int_A \left(\mathbf{n}_x \cdot \mathbf{E}(\mathbf{x},\mathbf{y})\right)_+ dAy$$

$$B(\mathbf{x}) \approx B^e(\mathbf{x}) + \rho(\mathbf{x}) \mathbf{n}_x \cdot \mathbf{E}_x$$

$$\mathbf{E}_x = \sum_j \int_{A_j} \mathbf{m}(\mathbf{x},\mathbf{y}) B(\mathbf{y}) dAy$$

(Assumes that all points within an emitter are close together and far from receiver – OK because of clustering + refinement)

# Higher order vector radiosity (5/6)

- Introduce per cluster basis functions:

$$B(\mathbf{x}) \approx B^e(\mathbf{x}) + \rho(\mathbf{x})\mathbf{n}_x \cdot \mathbf{E}_x$$

$$\mathbf{E}_x = \sum_j \int_{A_j} \mathbf{m}(\mathbf{x},\mathbf{y}) B(\mathbf{y}) dAy$$

$$\sum_{i,\alpha} B_{i,\alpha} \Phi_{i,\alpha}(\mathbf{x}) \approx \sum_{i,\alpha} B^e_{i,\alpha} \Phi_{i,\alpha}(\mathbf{x}) + \rho(\mathbf{x})\mathbf{n}_x \cdot \mathbf{E}_x$$

$$\mathbf{E}_x \approx \sum_{j,\beta} B_{j,\beta} \int_{A_j} \mathbf{m}(\mathbf{x},\mathbf{y}) \Phi_{j,\beta}(\mathbf{y}) dAy$$

(Assumes that radiosity is well approximated by a linear combination of non-overlapping orthogonal basis functions – OK because of clustering + refinement)

# Higher order vector radiosity (6/6)

- Resulting linear system

$$\mathbf{K}_{i,\alpha;j,\beta} = \frac{\int_{Ai} \Phi_{i,\alpha}(\mathbf{x}) \int_{Aj} \mathbf{m}(\mathbf{x},\mathbf{y}) \Phi_{j,\beta}(\mathbf{y})\, dA_y\, dA_x}{\int_{Ai} \Phi_{i,\alpha}(\mathbf{x})^2\, dA_x}$$ [Coupling]

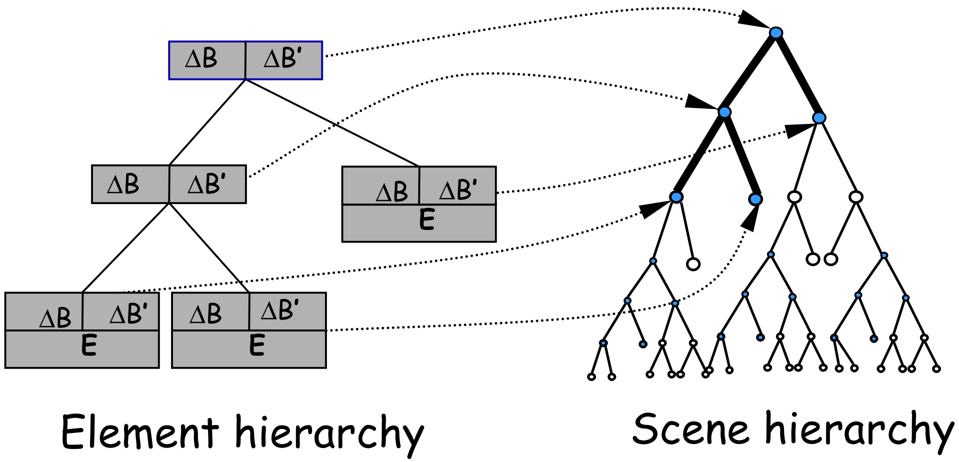
$$\mathbf{E}_{i,\alpha} = \sum_{j,\beta} \mathbf{K}_{i,\alpha;j,\beta} B_{j,\beta}$$ [Irradiance vector (unknown)]

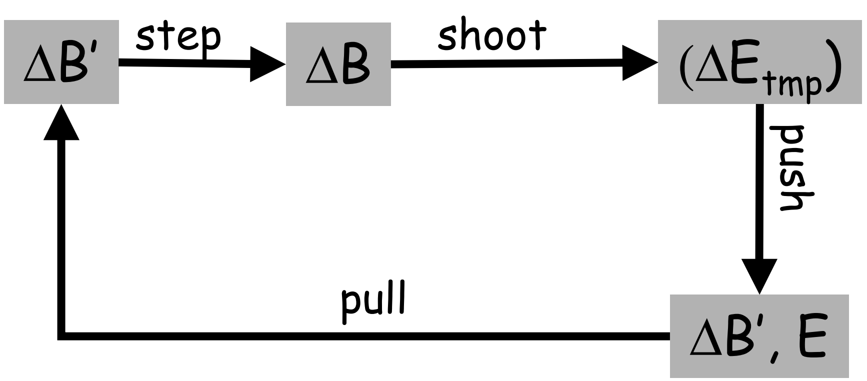
$$B_{j,\beta} = B^{e}_{j,\beta} + \rho_j \mathbf{n}_j \cdot \mathbf{E}_{j,\beta}$$ [Radiosity (temporary)]

(Galerkin method: inner product of left and right-hand side equation with each basis function $\Phi_{i,\alpha'}$)

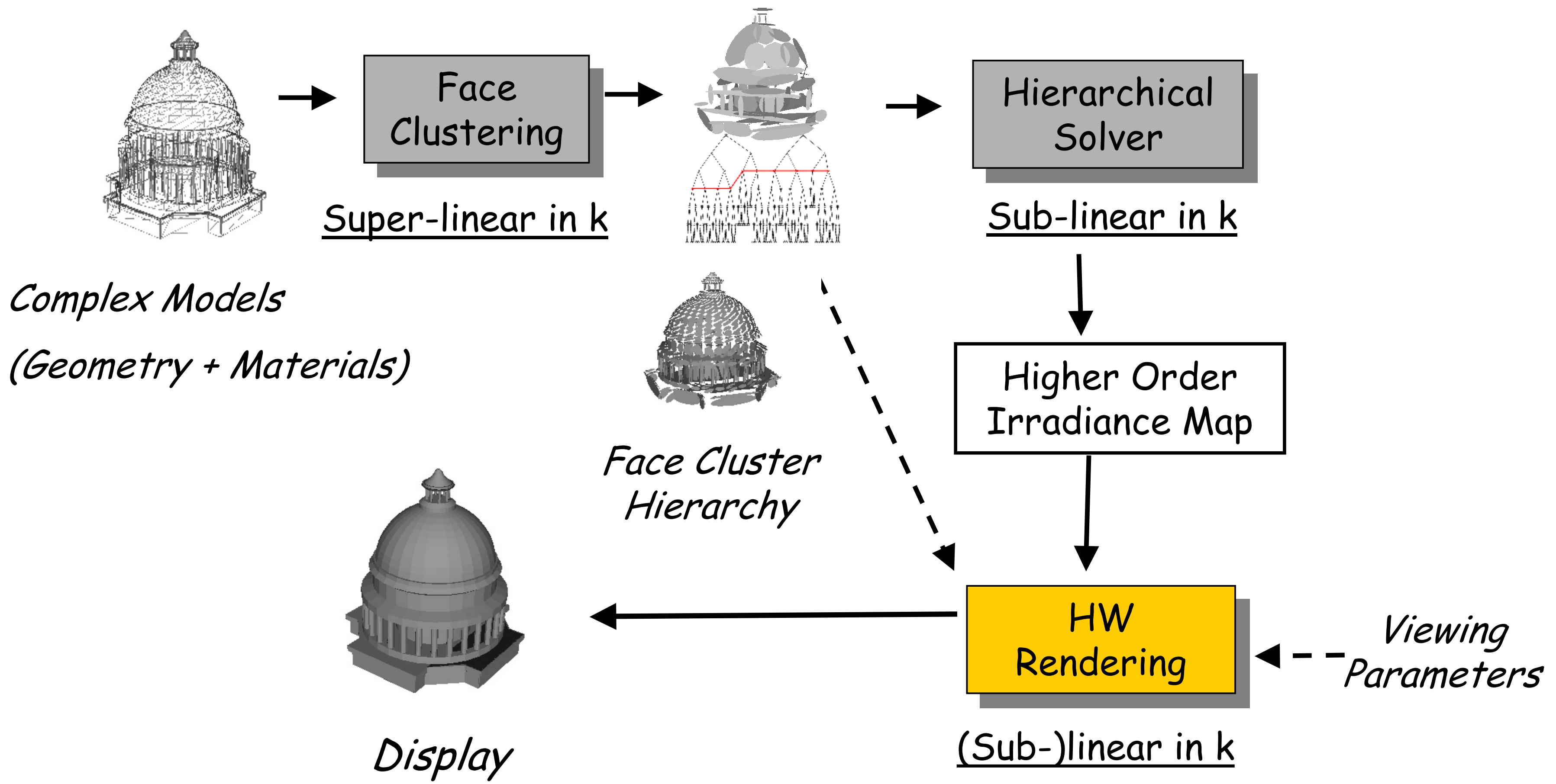
# A practical solution method

- Keep a separate element hierarchy
  - Push/pull/transfer only access nodes participating in the solution
- Minimize storage needs
  - don't store $\mathbf{K}_{i,\alpha;\ j,\beta}$ => Shooting method
  - store **E** only at the leaves => reorder energy exchanges
- Exploit face hierarchy for visibility queries
  - No need for auxiliary data structure
  - Multiresolution visibility reduces required resident set size
- (see paper for details)



# Method overview



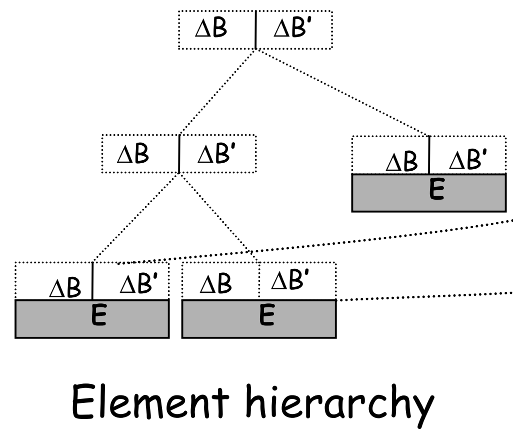
# Display (1/2)

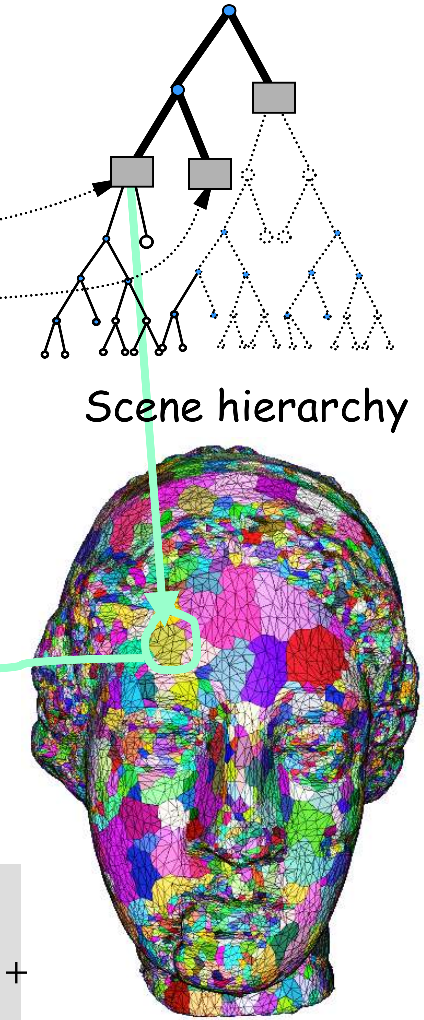
- Solution leaves partition the input model
  - Associated set of polygons
  - Associated smooth local representation of the global illumination environment

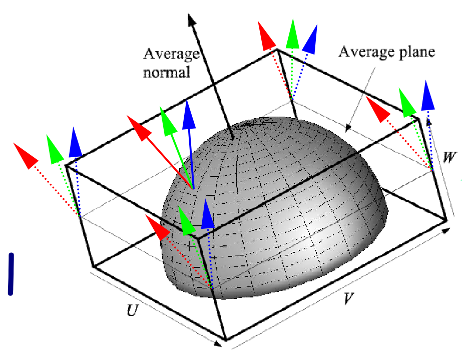
$$\mathbf{E}_i(\mathbf{x}) = \sum_{\alpha} E_{i,\alpha} \Phi_{i,\alpha}(\mathbf{x})$$

- Rendering can transform irradiance to radiance using full BRDF

$$L(\mathbf{x},\mathbf{z}) = L^e(\mathbf{x},\mathbf{z}) + f_r(\mathbf{x},\mathbf{x}+\mathbf{E}(\mathbf{x}),\mathbf{z})(\mathbf{n}_x \cdot \frac{\mathbf{E}(\mathbf{x})}{\pi})_+$$

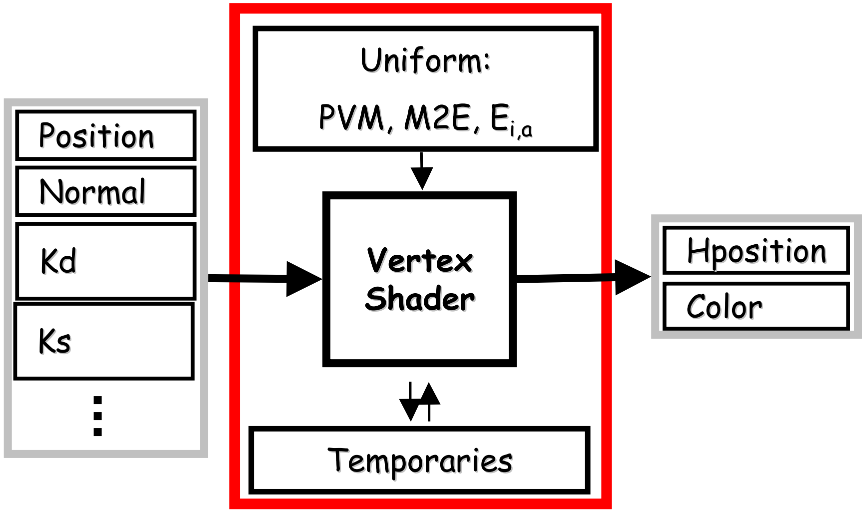


# Display (2/2)

- View-dependent results are not physically accurate…
  - Glossy reflections limited to final stage of any illumination paths for a single dominant light direction
- … but visually compelling…
- … and radiance computation can be computed very quickly on the GPU using vertex shader
  - For each leaf cluster:
    - Store irradiance coefficients into uniform program parameters
    - For each leaf polygon
      - Send BRDF coefficients, normal, and position at each vertex

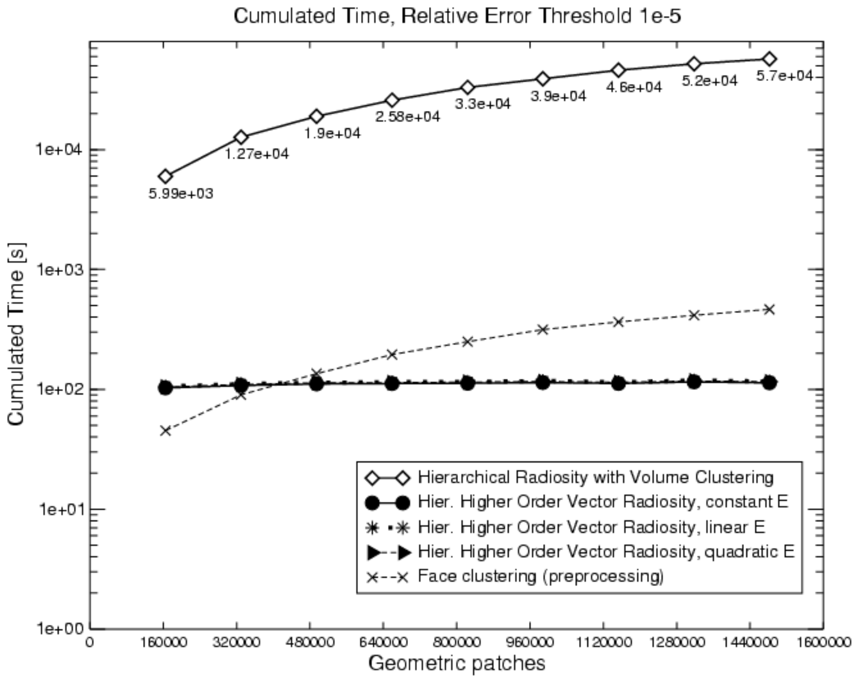


# Results (1/5)

- Test scene (1.5M polygons)
  - Closed box with colored walls, single area light source, 3 scanned models + tessellated implicit surface, glossy BDRF
  - Tested at several scene resolutions, along with HRVC radiosity algorithms (*renderpark*)
  - Linux box (Athlon XP 1600 MHz, 2GB RAM, NVIDIA GeForce4 Ti4600)

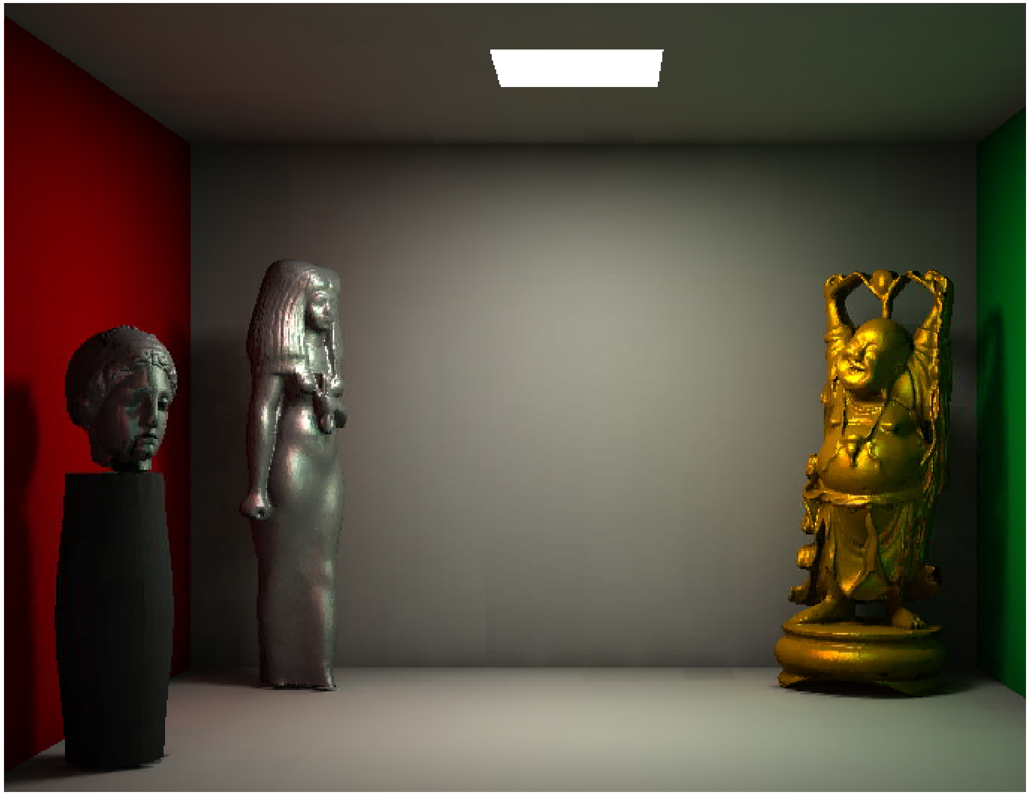
# Results (2/5)

- Solution time
  - Same scene, progressively fewer polygons
  - HRVC: 1h37 to 15h50
  - HHOFCR: ~100s (constant)
  - Clustering: 45s to 464s
- HHOFCR has constant solution time



# Results (3/5)

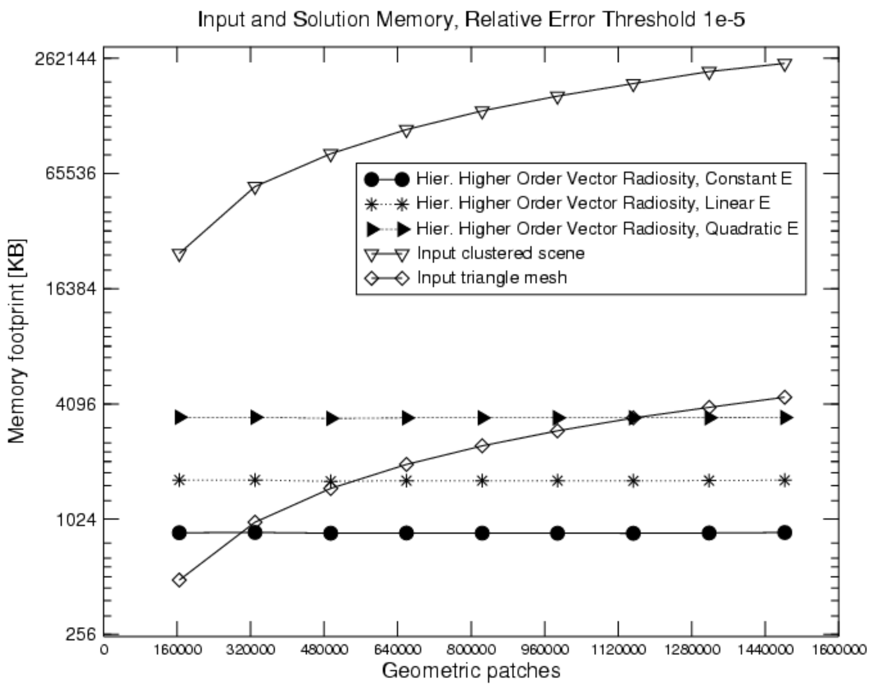
- Energy transfers
  - Same scene, progressively fewer polygons
  - Constant radiosity basis
  - Constant, linear, quadratic irradiance vector bases
- Higher order bases reduce energy transfers



Constant/Quadratic: 5.8 K leafs, 104K transfers

# Results (4/5)

- Memory requirements
  - Same scene, progressively fewer polygons
  - Constant radiosity basis
  - Constant, linear, quadratic irradiance vector bases
- Constant solution/working set memory
  - Solution memory is constant for a given basis:
    - 1MB for constant to 3.5MB quadratic basis
  - Working set is constant
    - ~10MB per simulation



# Results (5/5)

*Session close-up (366x847 subimage cut from 1280x1024 snapshots)*


*Live Video (divx compressed 512x384)*

Video demonstration: real time solution + inspection sequences

# Results (5/5)

*Session close-up*
*(366x847 subimage cut from 1280x1024 snapshots)*

*Live Video (divx compressed 512x384)*

Video demonstration: real time solution + inspection sequences

# Conclusions (1/3)

- The techniques proved highly effective for extending radiosity to detailed non-diffuse models
  - Extremely detailed scenes
  - Sub-linear performance in the number of input polygons
  - Low memory / CPU usage
  - Roughly approximates non-diffuse BRDFs
  - Supports interactive inspection of view-dependent solutions on standard commodity graphics platforms

# Conclusions (2/3)

- Method has also a number drawbacks…
  - Material range limited (diffuse to moderately glossy)
  - A few visible artifacts (sharp shadows)
  - Implementation rather complex (the devil is in the details)
- … mostly shared with other advanced radiosity methods

# Conclusions (3/3)

- Appropriate for a number of application domains
  - Rapid design cycle (interactive material “tweaking” possible at rendering time!)
  - Games
  - Interactive walkthroughs

# Future work

- Combine with other standard radiosity optimizations
  - Full decoupling of visibility
  - Smart links
- Extend to other surface types
  - Bump mapped surfaces, point sampled surfaces
- Improve approximation error analysis
- Move shading equations to the pixel level
  - More accurate, possibly faster
  - Requires full floating point graphics pipeline (GeForceFX)

# Contact/infos

CRS4 Visual Computing Group

http://www.crs4.it/vic/

(Additional tech reports, images, videos available)