<< RETURN

Reusing Shading for Interactive Global Illumination

GDC 2004

Kavita Bala Bruce Walter

Cornell University

Introduction

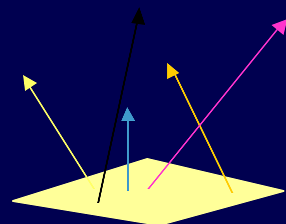
- What is this course about?
- Schedule
  - What is Global Illumination?
  - Computing Global Illumination
  - Reusing Shading
    - Image-space
    - Mesh-based
    - Fast GI

What is Global Illumination?

Session I

Assumptions: Geometric Optics

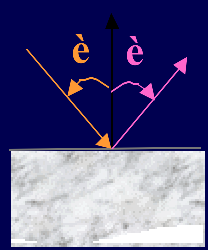
- Light travels in straight lines
- Rays do not interact with each other
- Rays have color(wavelength), intensity



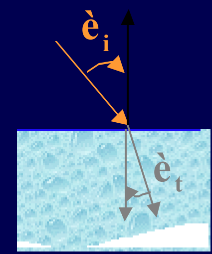
Reflections/Refractions

- Interface between 2 materials

reflection



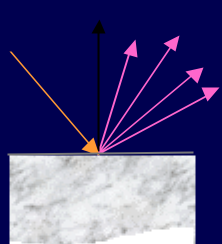
refraction



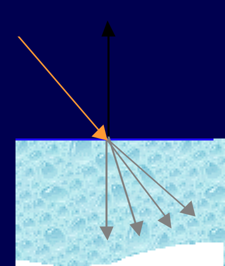
- Specular reflections and refractions
  - One direction

Realistic Reflections/Refractions

reflection

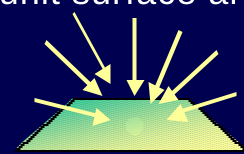


refraction



Radiometric Terms

- Power: energy per unit time
- Irradiance: Incident power per unit surface area
  - From all directions
  - Watt/m^2



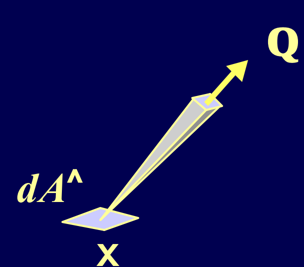
- Radiosity: Exitant power per unit surface area
  - Same units

GDC 2004

Radiance

- Radiance is radiant energy at x in direction θ: 5D function
  - $L(x \rightarrow \Theta)$: Power
    - per unit projected surface area
    - per unit solid angle

$$L(x \rightarrow \Theta) = \frac{d^2 P}{dA^{\perp} dw_{\Theta}}$$

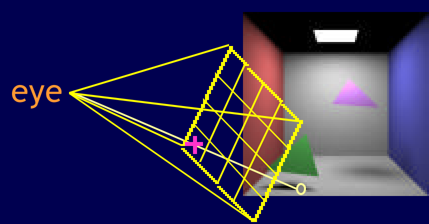


  - units: Watt / m^2.sr

GDC 2004

Why is radiance important?

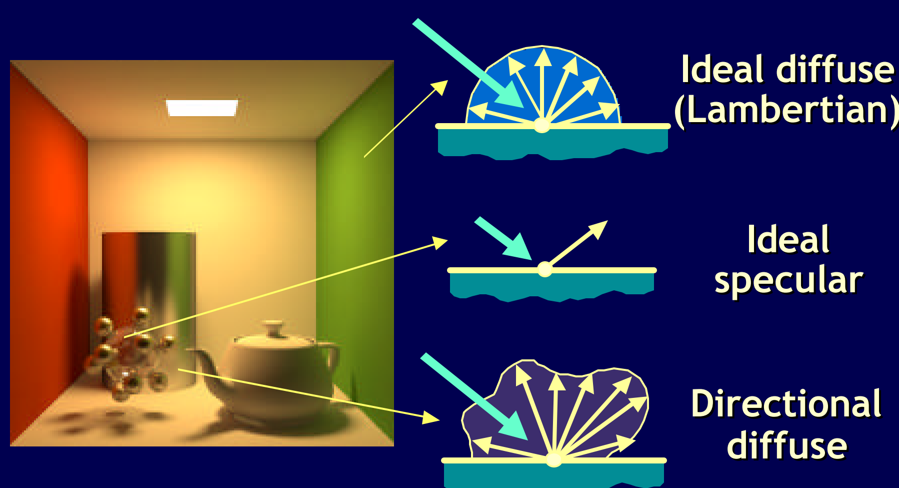
- Response of a sensor (camera, human eye) is proportional to radiance



- Pixel values in image proportional to radiance received from that direction

GDC 2004

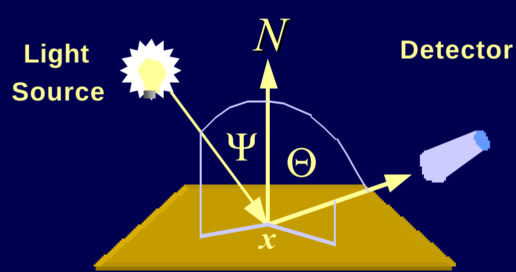
Materials - Three Forms



GDC 2004

BRDF

- Bidirectional Reflectance Distribution Function



$$f_r(x, \Psi \rightarrow \Theta) = \frac{dL(x \rightarrow \Theta)}{dE(x \leftarrow \Psi)} = \frac{dL(x \rightarrow \Theta)}{L(x \leftarrow \Psi)\cos(N_x, \Psi) dw_\Psi}$$

GDC 2004 Copyright 2004 Kavita Bala and Bruce Walter, Cornell University

BRDF special case: ideal diffuse

Pure Lambertian

$$f_r(x, \Psi \rightarrow \Theta) = \frac{r_d}{p}$$

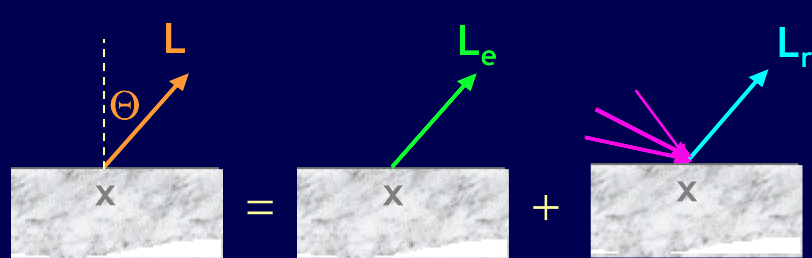
$$r_d = \frac{Energy_{out}}{Energy_{in}} \qquad 0 \leq r_d \leq 1$$

GDC 2004 Copyright 2004 Kavita Bala and Bruce Walter, Cornell University

Rendering Equation (RE)

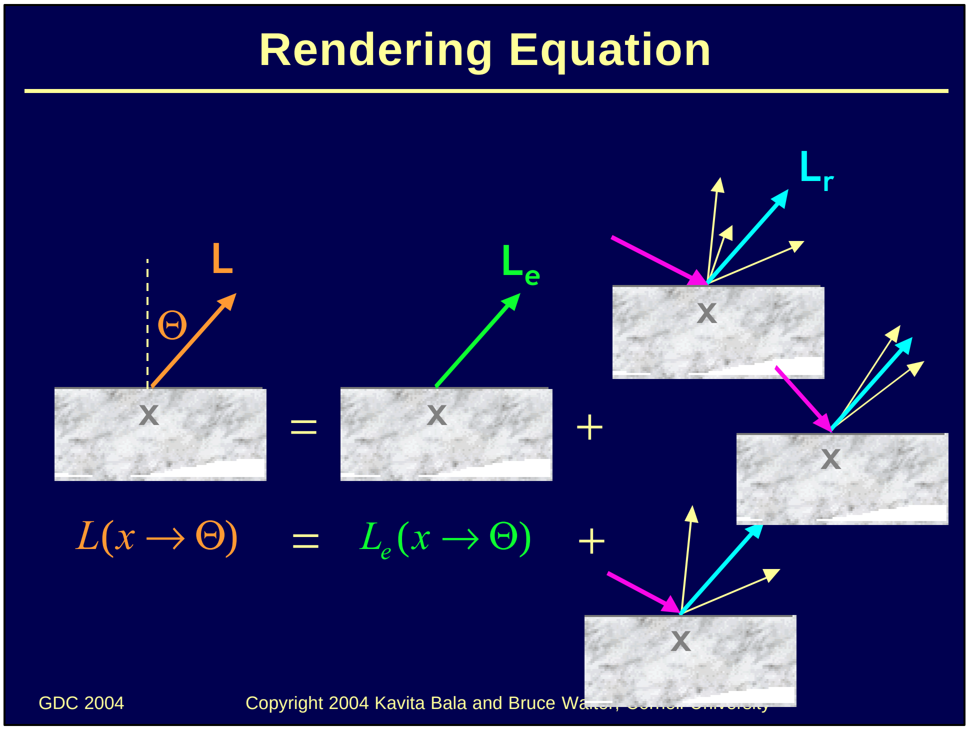
- RE describes energy transport in scene
- Input
  - Light sources
  - Surface geometry
  - Reflectance characteristics of surfaces
- Output: value of radiance at all surface points in all directions

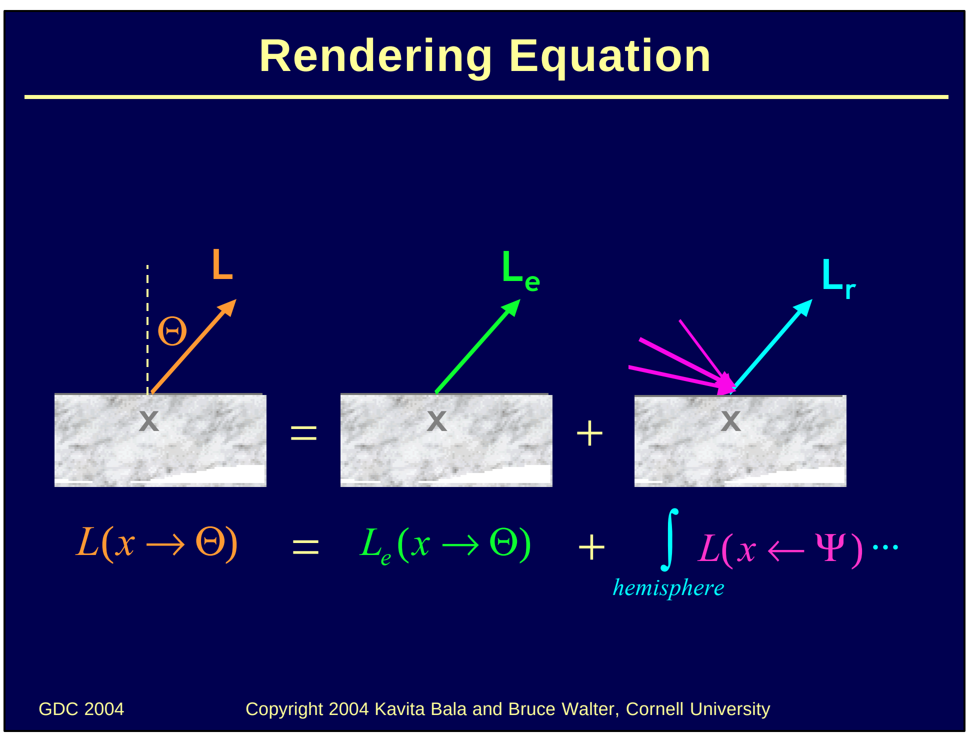
Rendering Equation



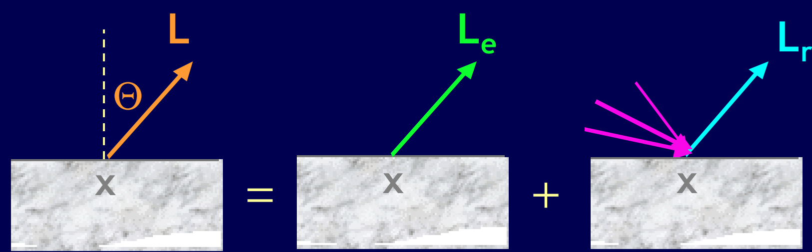
$$L(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + L_r(x \rightarrow \Theta)$$

GDC 2004 Copyright 2004 Kavita Bala and Bruce Walter, Cornell University


Rendering Equation
L
Θ
x
=
Le
x
+
Lr
x
x
x
L(x → Θ) = Le(x → Θ) +
GDC 2004
Copyright 2004 Kavita Bala and Bruce Walter, Cornell University


Rendering Equation
L
Θ
x
=
Le
x
+
Lr
x
L(x → Θ) = Le(x → Θ) + ∫ L(x ← Ψ) ···
hemisphere
GDC 2004
Copyright 2004 Kavita Bala and Bruce Walter, Cornell University

Rendering Equation

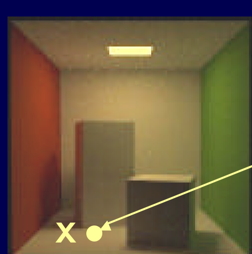


$$L(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + \int_{hemisphere} L(x \leftarrow \Psi)\ f_r(x, \Psi \leftrightarrow \Theta) \cos(\mathbf{N}_x, \Psi) dw_\Psi$$

- Applicable for each wavelength

GDC 2004

Rendering Equation



$$L(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + \int_{hemisphere} L(x \leftarrow \Psi) f_r(x, \Psi \leftrightarrow \Theta) \cos(\mathbf{N}_x, \Psi) dw_\Psi$$

incoming radiance

GDC 2004

Computing Global Illumination

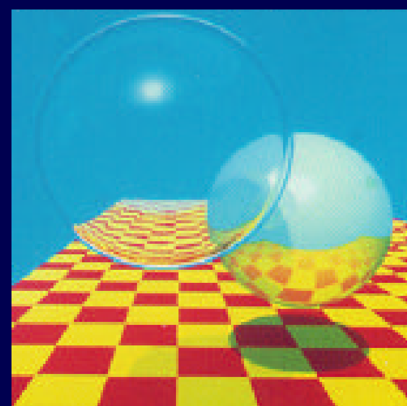
Session II

Classic Ray Tracing

- Introduced in 1980 by Turner Whitted
- Existing rendering:
  - Phong shading
  - Local illumination (specular, diffuse)

Insights

- Trace rays from eye into scene
  - Backward ray tracing
- Find visible objects
- Shade visible points
  - Shadows
  - Reflections
  - Refractions



Whitted 1980: First ray traced image

- First global illumination algorithm!

Whitted RT Assumptions

- Light Source: point light source
  - Hard shadows
  - Single shadow ray direction
- Material: Blinn-Phong model
  - Diffuse with specular peak
- Light Propagation
  - Occluding objects
  - Specular interreflections only
    - trace rays in mirror reflection direction only

GDC 2004 Copyright 2004 Kavita Bala and Bruce Walter, Cornell University

History

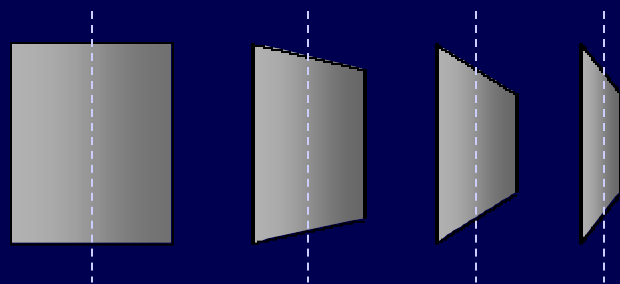
- Problems with classic ray tracing:
  - Not realistic: only perfect specular and perfect refractive/reflection between surfaces
  - View-dependent
- Radiosity (1984)
  - Global Illumination in diffuse scenes
  - Discretize scene
- Monte Carlo Ray Tracing (1986)
  - Global Illumination for any environment

Radiosity Advantages

- Physically based approach for diffuse environments
- Can model diffuse interactions, color bleeding, indirect lighting and penumbra (area light sources)
- Boundary element (finite element) problem
- Accounts for very high percentage of total energy transfer

GDC 2004
Copyright 2004 Kavita Bala and Bruce Walter, Cornell University

Key Idea #1: diffuse only

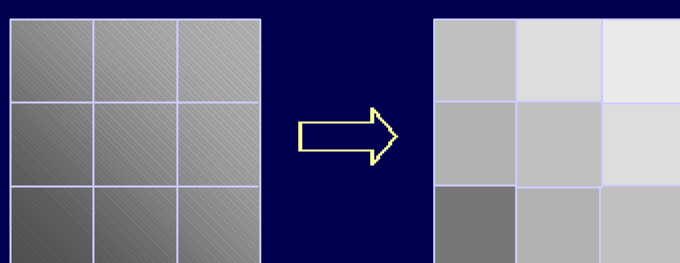


- Radiance independent of direction
- Surface looks the same from any viewpoint
- No specular reflections

GDC 2004

Key Idea #2: “constant”polygons

- Radiosity solution is an approximation, due to discretization of scene into patches



- Subdivide scene into small polygons

GDC 2004

Radiosity Equation

- Radiosity equation for each polygon i

$$Radiosity_1 = Radiosity_{self,1} + \sum_{j=1}^{N} a_{j\to 1} Radiosity_j$$

$$Radiosity_2 = Radiosity_{self,2} + \sum_{j=1}^{N} a_{j\to 2} Radiosity_j$$

$$\ldots$$

$$Radiosity_N = Radiosity_{self,N} + \sum_{j=1}^{N} a_{j\to N} Radiosity_j$$

- N equations; N unknown variables

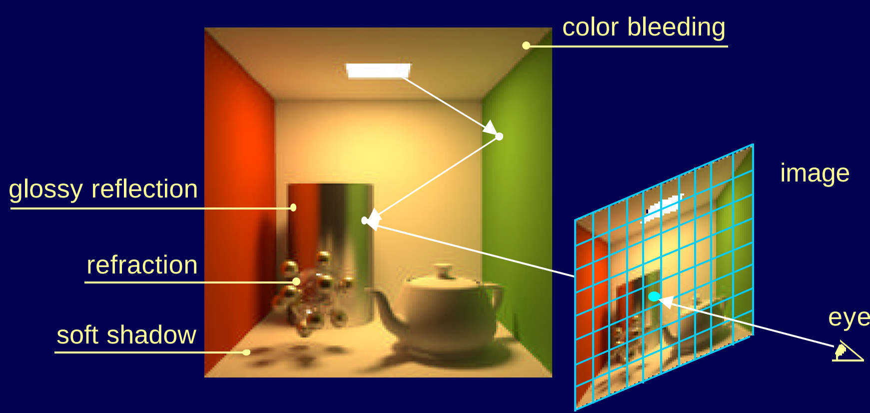
Solutions

(RenderPark)

Radiosity: Typical Image

GDC 2004 Copyright 2004 Kavita Bala and Bruce Walter, Cornell University

Rendering



GDC 2004 Copyright 2004 Kavita Bala and Bruce Walter, Cornell University

Path Tracing

- Monte Carlo Method
  - Randomly choose potential light paths
  - Average contribution over many such paths
- Problems
  - Noise
  - Slow to converge

Approaches

- Bidirectional
- Metropolis
- Photon Mapping
- Instant Radiosity

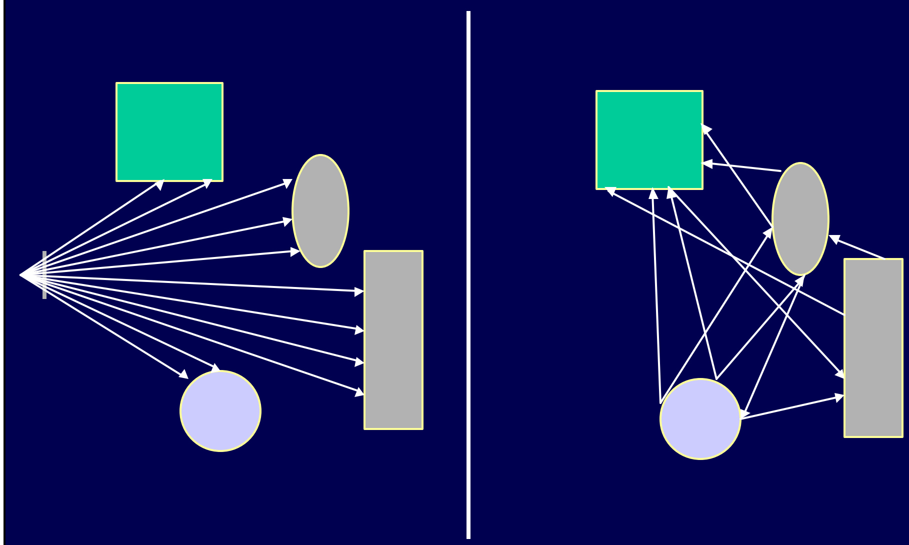
Fast Global Illumination

Session III

Hardware's Strengths

- What is the hardware good at?
  - Fast visibility determination (z-buffer)
  - Fast texture map lookup
  - Fast shading
    - Can even be per-pixel with latest boards

Hardware’s Strengths



Hardware’s Strengths

- Fast visibility determination (z-buffer)
  - Fast in an amortized sense
    - One rendering determines the visible surface seen at all pixels simultaneously
  - Great for some visibility queries types
    - Primary (eye) rays
    - Shadow rays (point sources)
  - Not so good for other types
    - Reflection & refraction from curved surfaces
    - Indirect illumination
    - Adaptive sampling

Hardware's Strengths

- Fast texture map lookup
  - Cheap anti-aliasing & anisotropic filtering
  - Most flexible part of graphics hardware
    - Surface texturing
    - Bump mapping
    - Reflection mapping
    - Shadow mapping
    - Even arbitrary BRDF approximations
  - Very powerful
    - But not as compute intensive

Hardware's Strengths

- Fast shading
  - Latest boards can do per-pixel shading
  - Programmable, but limited operations
  - Local shading only
    - All inputs must be provided ahead of time
    - Non-local shading can only be approximated
      - Shadows, reflections, indirect, etc

Hardware's Strengths

- Conclusion
  - Hardware still has the edge due to its dedicated pipeline
  - Advantage may actually be decreasing
  - Software attractive for its flexiblility
    - If it can be made "fast enough" for interactive use
    - And handle scene and/or effects the hardware cannot handle

Interactive Software Rendering

- Interactive
  - User-driven, not pre-scripted animation
  - At least a few frames per second (fps)
- Software
  - Major shading done in software
    - Can use hardware to help
- Rendering
  - Online, not pre-computed or captured
    - Eg, lightfields are pre-computed

Ray Tracing (or Ray Casting)

- Very common visibility tool for software
  - Flexible
  - Efficient for large models
    - Using an acceleration structure (grids, bsp, etc)
  - Usually the largest computational bottleneck
  - Easily parallelizable
    - Each pixel can be computed in parallel

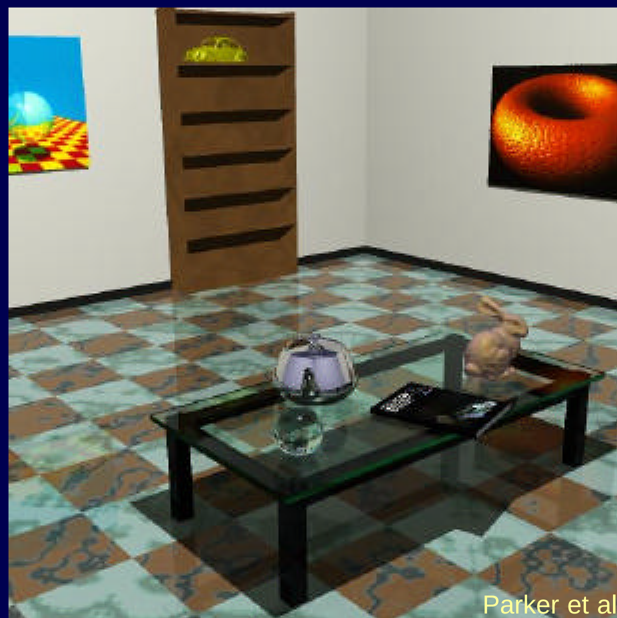
GDC 2004

Interactive RT [Parker et al.]

- SGI Origin 2000
  - 64 processors
  - Shared memory
- Whitted-style ray tracing
  - Shadow, reflection, and refraction rays
- Non-polygonal primitives
  - Spheres and splines

15 fps

GDC 2004



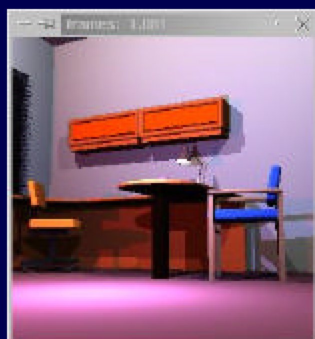
4 fps

Coherent RT [Wald et. al.]

- Created a highly optimized ray tracing engine for Intel-based PCs
  - Carefully profiled their ray tracer
    - Discovered it was often memory bound
  - Hand-crafted and tuned their code
    - Created both C and assembly versions
    - Used compact, cache-friendly data structures
    - Optimized for SIMD (SSE)
    - Reordered computations for better coherence

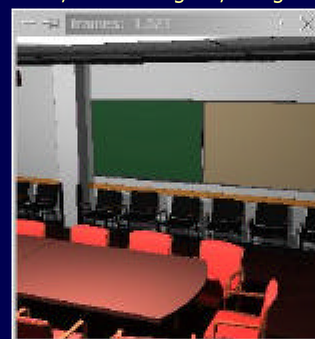
Test Scenes I

- Test scenes: 800 triangles to >8 million



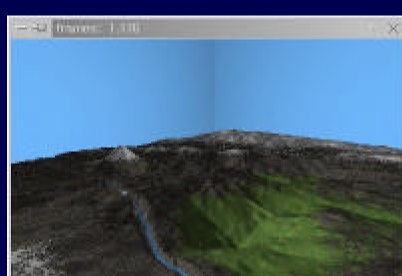
Office:
34,000 triangles, 3 lights

Conference Room:
280,000 triangles, 2 lights



Test Scenes II

- Test scenes: 800 triangles to >8 million

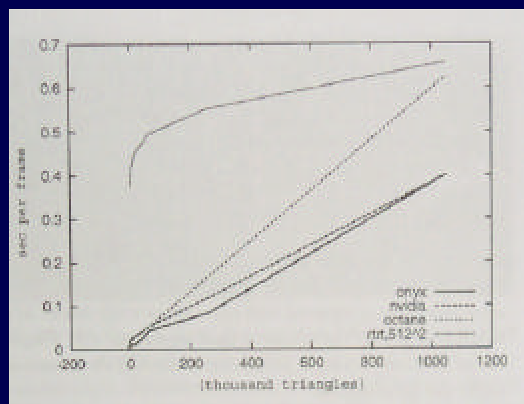


Terrain:
1 million triangles (textured)

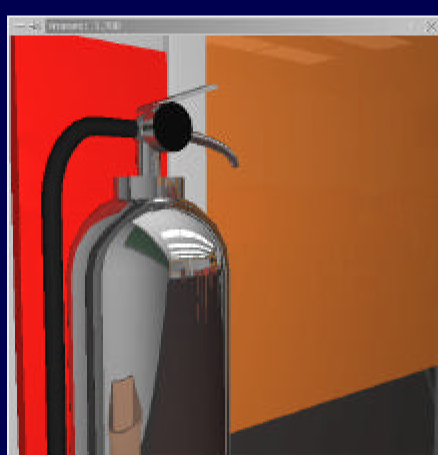
Berkeley Soda Hall:
1.5 to 8 million triangles

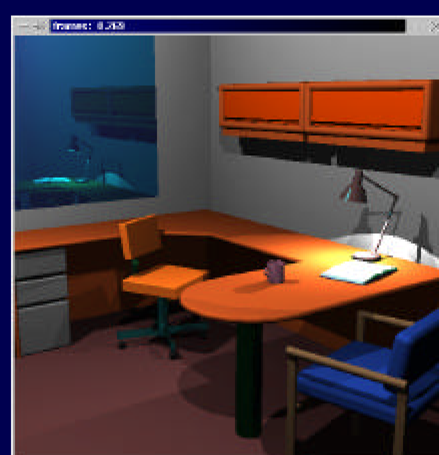
Performance Results III

- Comparison to Rasterization-Hardware
  - Ray tracing scales well for large environments



Sample Images





Fast RT [Benthin et. al.]

- SIMD and careful hand coding of inner loops
- 0.4 – 2 Million rays per second
  - Athlon MP 1800+
- *Instant Radiosity* used for global illumination
- Up to 48 processors for interactive performance

Reusing Shading for Global Illumination

Session IV

Reusing Shading

- Full simulation can take hours
- Interactive rendering by exploiting coherence

GDC 2004 Copyright 2004 Kavita Bala and Bruce Walter, Cornell University

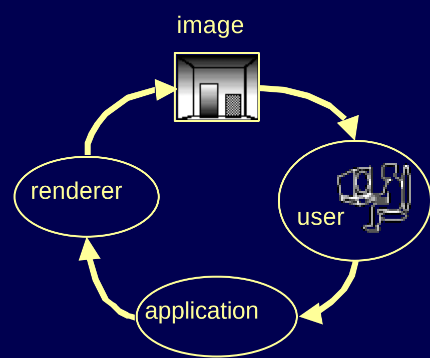
Display Process

- Automatically exploit spatial and temporal coherence
- Layered on top of an existing (slow) global illumination renderer
- Provide interactive performance

GDC 2004 Copyright 2004 Kavita Bala and Bruce Walter, Cornell University

Interactive Global Illumination

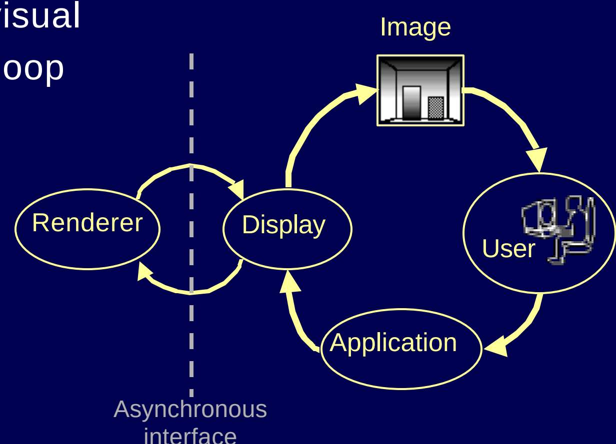
- Standard visual feedback loop
  - Entirely synchronous
  - Framerate is limited by the renderer



GDC 2004

Interactive Global Illumination

- Modified visual feedback loop



GDC 2004

Interactive Global Illumination

- Interactive requirements
  - Image quality
  - Responsiveness
    - Don't make the user wait
    - Provide rapid user feedback
  - Consistency
    - Don't surprise or distract the user
    - Avoid sudden changes if possible
      - Eg, in quality, frame rate, popping, etc.

GDC 2004

Issues

- How to cache results for reuse?
  - Interpolate images from sparse data
  - Handle camera motion between frames
    - Handle occlusion changes
- Which samples should be rendered?
  - Prioritize for maximum benefit
- What if scene or shading changes?
  - Detect and discard data that is no longer valid

GDC 2004

Approaches

- Image based
  - Post-rendering Warp *(Mark97)*
  - Corrective texturing *(Stamminger00)*
- Point based
  - Render Cache *(Walter99,02)*
  - Edge-and-Point Rendering (Bala03)
- Mesh based
  - Tapestry *(Simmons00)*
  - Shading Cache *(Tole02)*
- 4D approaches
  - Holodeck *(Ward98,99)*
  - Radiance Interpolants (Bala99)

Frameless Rendering

- Update pixels as they are computed
  - Don't wait for full frame to finish

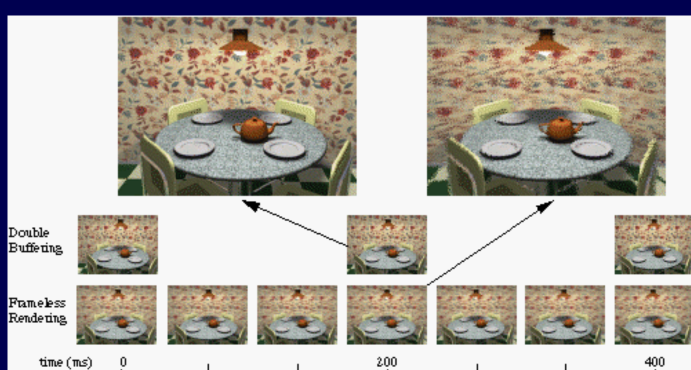


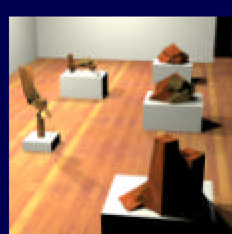
Figure 1. The bottom row shows 7 frames of a 15 Hz frameless rendering sequence with 33% of the pixels updated in each frame. The middle row shows 3 frames of a double-buffering sequence updated at 5 Hz.

Post-Rendering 3D Warp

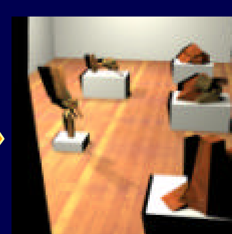
- Render subset of frames
  - E.g, every 6th frame is rendered
- Use standard image warping techniques to compute the other frames

Post-Rendering 3D Warp

- Problem: holes and missing data



Reference frame

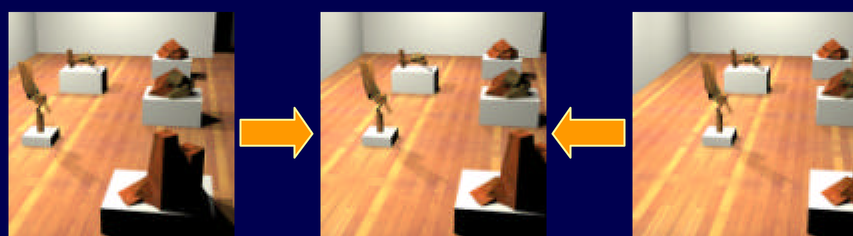


Warped frame

The camera is moving to the left in this example.

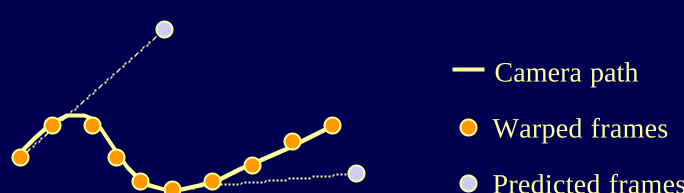
Post-Rendering 3D Warp

- Warp from both past and future reference frames
  - Heuristics for combining pixel results



Post-Rendering 3D Warp

- Must predict the locations of future frames
  - Longer predictions become rapidly less accurate



GDC 2004 Copyright 2004 Kavita Bala and Bruce Walter, Cornell University

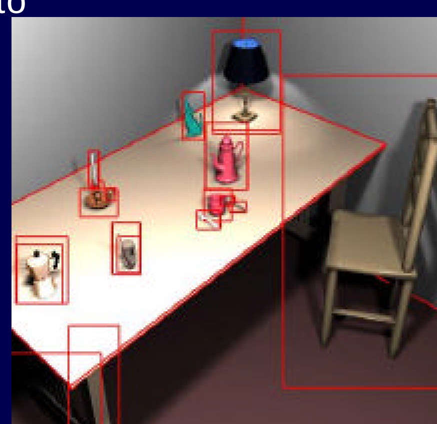
Corrective Texturing

- Start with a standard hardware rendering of scene
  - Graphics hardware very good at interactive display
  - Start with a radiosity solution
- Compare to underlying renderer
  - Apply corrections where they differ
  - Corrections applied as projective textures

GDC 2004 Copyright 2004 Kavita Bala and Bruce Walter, Cornell University

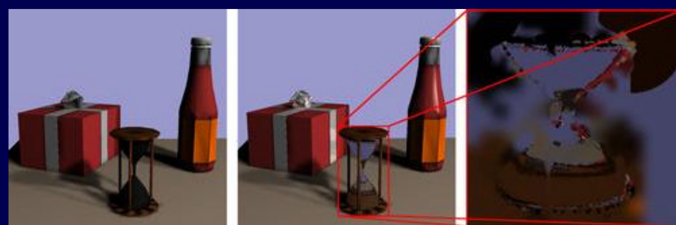
Corrective Texturing

- Corrective textures are dynamically assigned to objects



GDC 2004 Copyright 2004 Kavita Bala and Bruce Walter, Cornell University

Corrective Texturing



Radiosity solution | Corrected image | Corrective texture

Corrective Texturing

- Sparse rendered samples compared to hardware displayed results
  - Differences splatted into textures
  - More samples generated near points which had large differences
  - Samples which are likely to have changed are deprecated so that can be overwritten by future results

Approaches

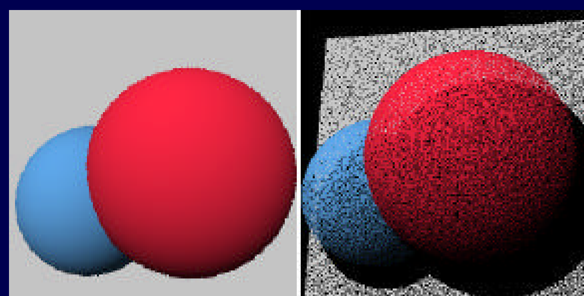
- Image based
  - Post-rendering Warp *(Mark97)*
  - Corrective texturing *(Stamminger00)*
- Point based
  - Render Cache *(Walter99,02)*
  - Edge-and-Point Rendering (Bala03)
- Mesh based
  - Tapestry *(Simmons00)*
  - Shading Cache *(Tole02)*
- 4D approaches
  - Holodeck *(Ward98,99)*
  - Radiance Interpolants (Bala99)

Render Cache

- Results stored as cloud of unordered points with:
  - 3D position (located on surfaces)
  - Color
  - Age
  - Object identifier

Render Cache

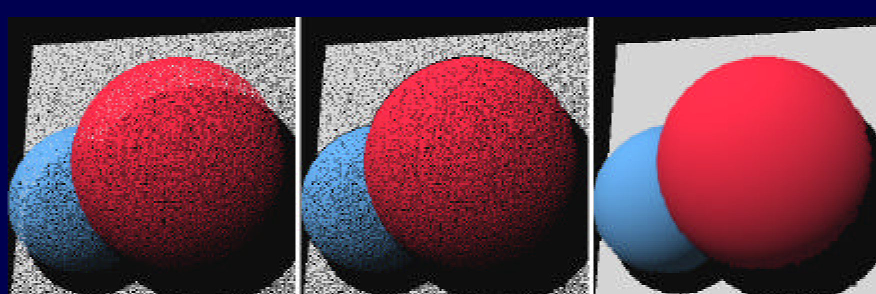
- Reproject points into current frame
  - Occlusion errors
  - Holes in data



Initial view After reprojection

Render Cache

- Use occlusion culling heuristic
- Interpolation to fill holes
  - Fixed size kernels, 3x3 and 7x7



Reprojection Occlusion cull Interpolation

GDC 2004
Copyright 2004 Kavita Bala and Bruce Walter, Cornell University

Render Cache

- Priority image for sampling
  - High priority for sparse regions
  - High priority for old points
- Convert priority image to sparse set of locations to be rendered
  - Uses error-diffusion dither
- Also uses predictive sampling
  - Try to sample new regions just before they become visible

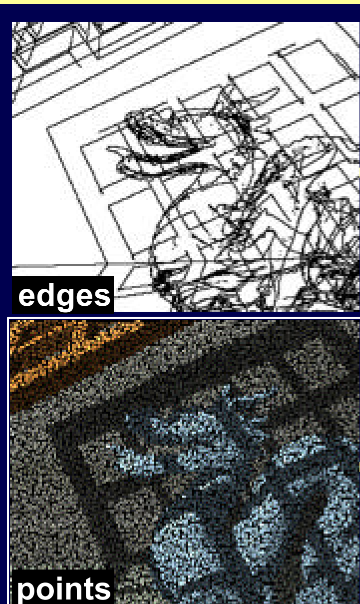
Render Cache

Displayed image   Priority image   Requested pixels

Render Cache

- Recomputes old samples to detect changes
  - Nearby points are aged to raise priority and cause point invalidation
- Object motion
  - Associated points can be transformed along with the object

Edge-and-Point Rendering



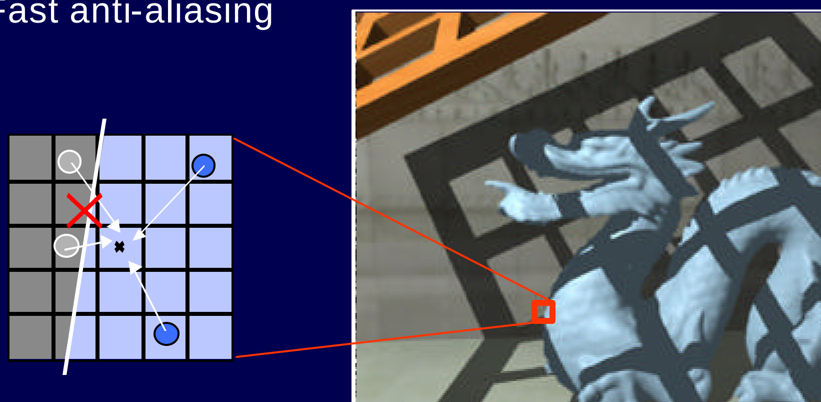
Edges: important discontinuities

– Silhouettes and shadows

Points: sparse shading samples

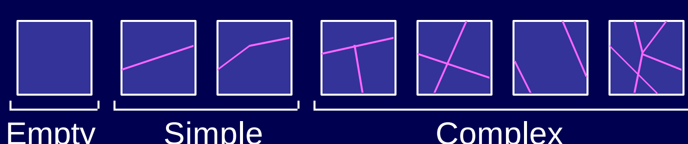
Edge-and-Point Image

- Alternative display representation
- Edge-constrained interpolation preserves sharp features
- Fast anti-aliasing



Pixel types

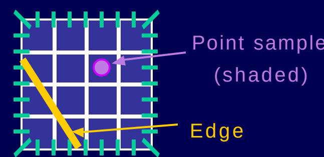
- Pixels can have arbitrary edge complexity
- Classify pixels into 3 groups
  - Empty: no edges
  - Simple: can be approximated by 1 edge
  - Complex: everything else



  - Typical pixel classification statistics
    - empty (85-95%), simple (4-10%), complex (1-4%)

Edge-and-Point Image (EPI)

- Goal: compact and fast
  - Store at most one edge and one point per pixel
  - Limited sub-pixel precision



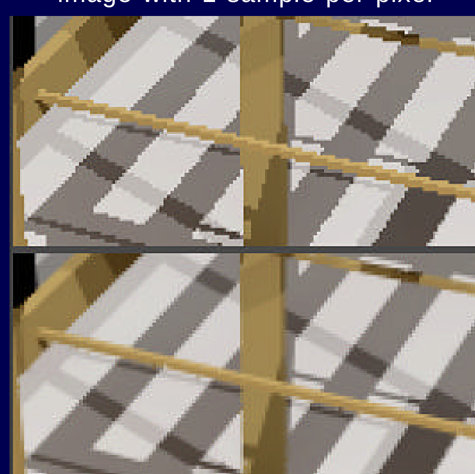
EPI pixel

- Combine edges and points in image space
  - View-driven, lazy evaluation

Fast Anti-aliasing

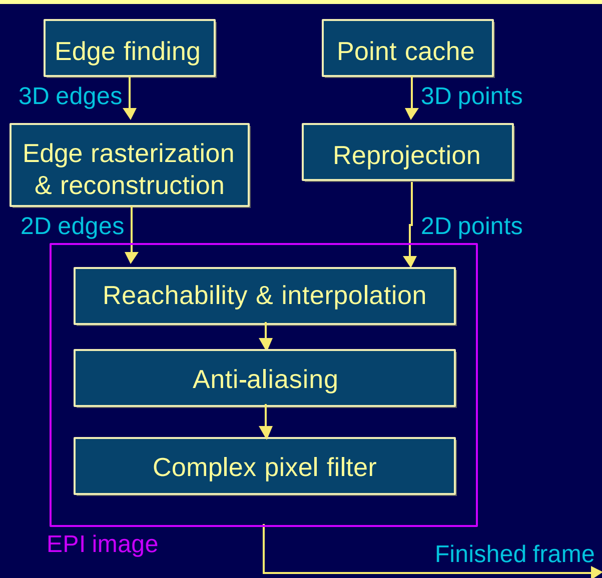
Our result using
<1 sample per pixel

Magnified view of a ray traced image with 1 sample per pixel



Magnified view of our results

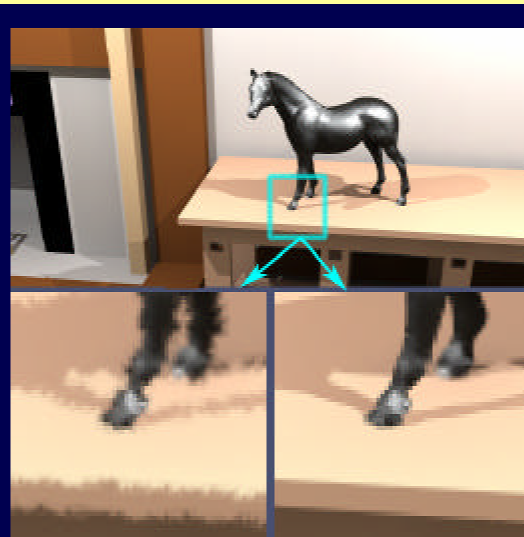
Putting it Together



GDC 2004

Results: Quality

- Global Illumination
  - 3 lights
  - 150k polygons
- Sparseness Ratio
  - 100: 1
- Performance
  - 8-14 fps



Without Edges **With Edges**

GDC 2004

Approaches

- Image based
  - Post-rendering Warp *(Mark97)*
  - Corrective texturing *(Stamminger00)*
- Point based
  - Render Cache *(Walter99,02)*
  - Edge-and-Point Rendering (Bala03)
- Mesh based
  - Tapestry *(Simmons00)*
  - Shading Cache *(Tole02)*
- 4D approaches
  - Holodeck *(Ward98,99)*
  - Radiance Interpolants (Bala99)

GDC 2004

Holodeck

- Uses Radiance as its renderer
- Rendered samples stored in a 4D data structure
  - Similar to Light Field or Lumigraph
  - Can be very large
    - Paged to disk if necessary
  - Lazily evaluated
    - Samples generated near current viewpoint
  - Position and other parameters are specified by the user

GDC 2004

Holodeck

- Uses Gouraud-shaded triangle mesh
  - Get samples near current viewpoint
  - Samples become vertices in a mesh
  - Delaunay triangulation of samples in direction space about a center of projection
  - Hardware provides fast display including interpolation between samples

Holodeck

- Mesh construction
  - Choose center point
  - Construct Delaunay triangulation
    - Based on sample point's projection onto a sphere about the center point
  - Display mesh using hardware
  - Update incrementally with new samples
- If user moves too far, then must choose new center and rebuild mesh

Holodeck

- Depth heuristic to reduce occlusion errors
- Special techniques for designated moving objects

GDC 2004

Tapestry

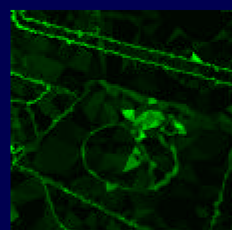
- Based on Holodeck system with several enhancements:
  - Prioritized sampling
  - Incremental “recentering” of spherical Delaunay mesh as viewpoint moves
  - Fixed cache size
    - Max vertices = pixels
  - Sample invalidation
    - Occlusion and color change heuristics

GDC 2004

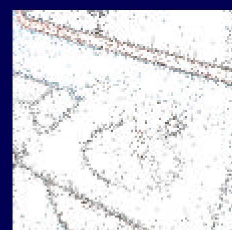
Tapestry

- Each triangle assigned a priority
  - Color & depth differences and age
  - Rasterize priority using hardware
  - Quasi-random sampling with rejection

Image



Priority



Samples

Shading Cache

- Display mesh is refinement of original scene mesh
  - No occlusion errors
  - Hardware handles textures
  - Display mesh >= original mesh
  - Easier to handle moving objects
- Decouples frame update from mesh update

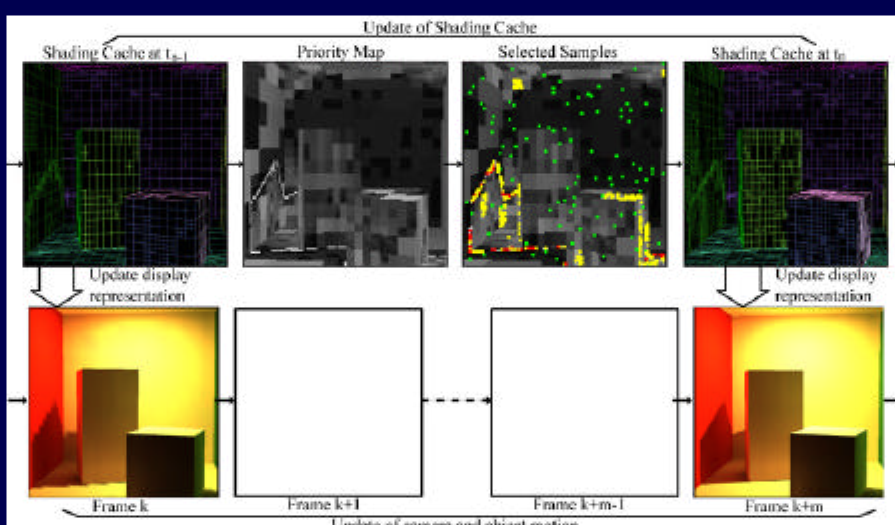
GDC 2004
Copyright 2004 Kavita Bala and Bruce Walter, Cornell University

Shading Cache

- Adds flood-fill heuristic for sampling
  - Discontinuities require locally dense subdivision
- Mesh de-refinement
  - If not recently visible
  - If denser than pixel spacing
  - If color changes are detected

GDC 2004 Copyright 2004 Kavita Bala and Bruce Walter, Cornell University

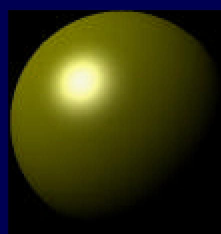
Shading Cache

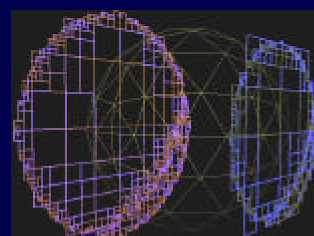


GDC 2004 Copyright 2004 Kavita Bala and Bruce Walter, Cornell University

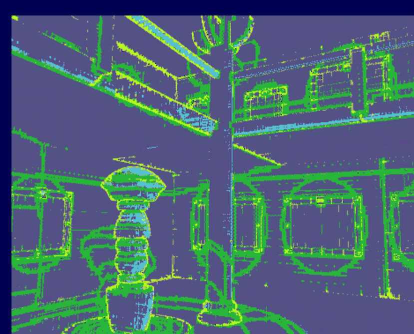
Radiance Interpolants Bala[96,97,99]

- Radiance interpolant:
  - Set of sparse samples (in 4D line space) of radiance function
  - Bounded-error, accurate reconstruction





Results: Museum Scene



gray: interpolation success
yellow: silhouettes; green: shadows

GDC 2004 Copyright 2004 Kavita Bala and Bruce Walter, Cornell University

Downloadable Versions

- Render Cache
  - http://www.graphics.cornell.edu/research/interactive/rendercache/
- Holodeck
  - http://radsite.lbl.gov/radiance

Conclusion

- Software Interactive Rendering is possible now with current machines
  - Good scaling with scene complexity
  - Much greater shading flexibility
- Many interesting challenges still remain
  - Higher resolutions
  - Fully dynamic environments
  - Anti-aliasing

GDC 2004
Copyright 2004 Kavita Bala and Bruce Walter, Cornell University

Bibliography

- Parker, Martin, Sloan, Shirley, Smits, Hansen, **Interactive Ray Tracing**, Symposium on Interactive 3D Graphics, April 1999
- Wald, Benthin, Wagner, Slusallek, **Interactive Rendering with Coherent Ray Tracing**, Proceedings of Eurographics 2001
- Walter, Drettakis, Parker, **Interactive Rendering using the Render Cache**, Rendering Techniques 1999
- Bala, Walter, Greenberg, **Combining edges and points for Interactive high quality rendering**, SIGGRAPH 2003

Bibliography

- Stamminger, Haber, Schirmacher, Seidel, **Walthroughs with Corrective Texturing**, Rendering Techniques 2000
- Simmons, Séquin, **Tapestry: A Dynamic Mesh-based Display Representation for Interactive Rendering**, Rendering Techniques 2000
- Bala, Dorsey, Teller: **Radiance Interpolants for Accelerated Bounded-Error Ray Tracing**, Transactions on Graphics 1999