Using RenderMan for ray tracing and global illumination in complex scenes

Per Christensen

Pixar Animation Studios

DTU, June 2005

PIXAR

Overview

- Pixar and Pixar movies
- RenderMan
- Recent research: ray tracing and global illumination

PIXAR

Pixar

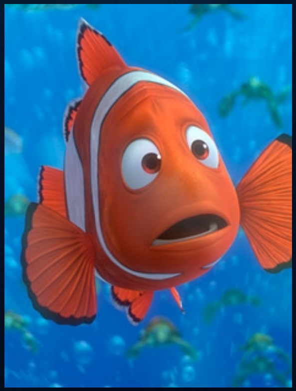
- Founded in 1986
- ~700 employees: artists, programmers, …
- Headquarter in Emeryville (California)
- Small group in Seattle (Washington)

Pixar movies

- Toy Story
- A Bugs Life
- Toy Story 2
- Monsters, Inc.
- Finding Nemo
- The Incredibles
- Cars (2006)

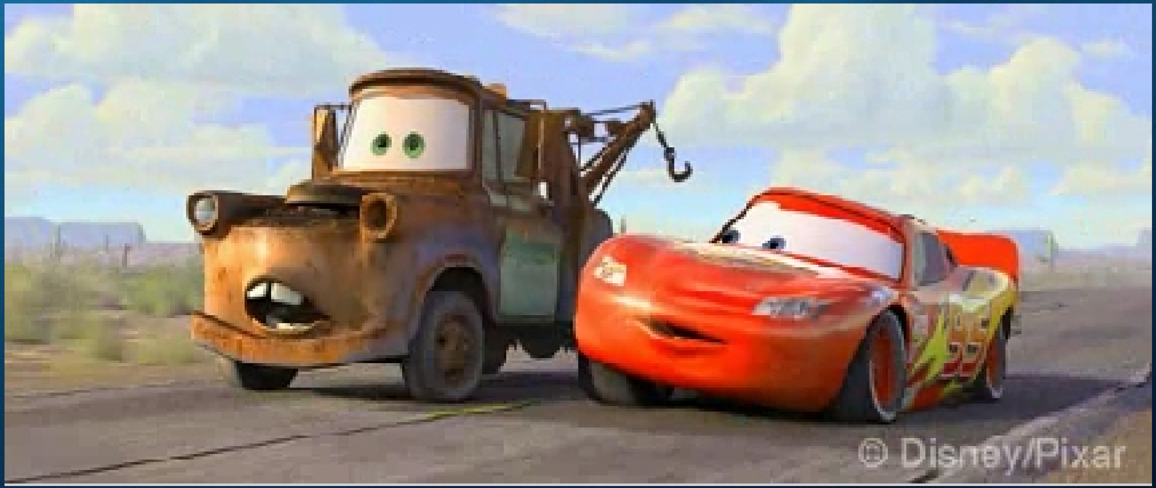







PIXAR

Cars challenges



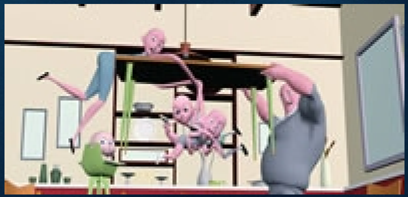
- Animation: cars that move, talk, “think”
- Rendering:
  - geometric complexity
  - reflections

PIXAR

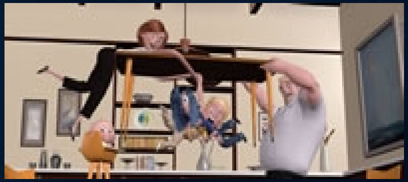
Making a Pixar movie

- Story development

- Layout, timing



- Modeling

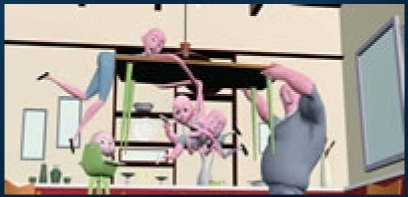


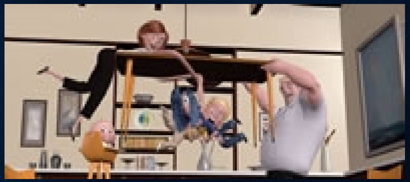
- Animation, simulation

- Shading, lighting
- Rendering

Making a Pixar movie

- Story development
- Layout, timing
- Modeling
- Animation, simulation
- Shading, lighting
- Rendering !!





PIXAR

Typical scene at Pixar

- 100s of lights
- 1,000s of textures – too many to fit in mem!
- 10,000s of objects
- 100,000,000s of polygons – too many to fit!
- Programmable shading

PIXAR

Rendering requirements

- Render at hi-res (~2000 pixels)
- Motion blur

- Depth of field

- No spatial or temporal aliasing (staircase effects, “crawlies”, popping, …)

PIXAR

RenderMan

- Used to render all Pixar movies (CG)
- Used by most other movie studios, too, for special effects:
  - The Abyss, Terminator 2, Jurassic Park, …, Lord of The Rings, Harry Potter, Star Wars

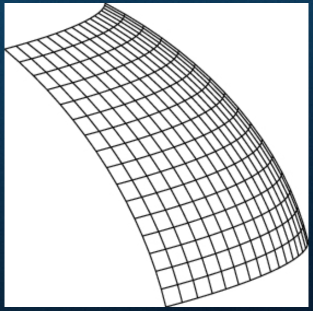
RenderMan

- Very robust and flexible
- Can handle very complex scenes
- Industry standard
- C and C++
- Based on scanline rendering, but now extended with ray tracing and global illumination

PIXAR

Scanline rendering

- Split each object into surface patches
- Tessellation: divide each patch into many tiny micropolygons (“quads”)



- Compute a color for each micropolygon

PIXAR

Scanline rendering

PIXAR

Scanline rendering

- Advantages:
  - Fast
  - One image tile at a time: only needs small fraction of objects+textures
  - Can deal with very complex scenes
- Limitations:
  - Shadow maps (limited resolution)
  - Reflection maps (no interreflections)

PIXAR

Recent research & development

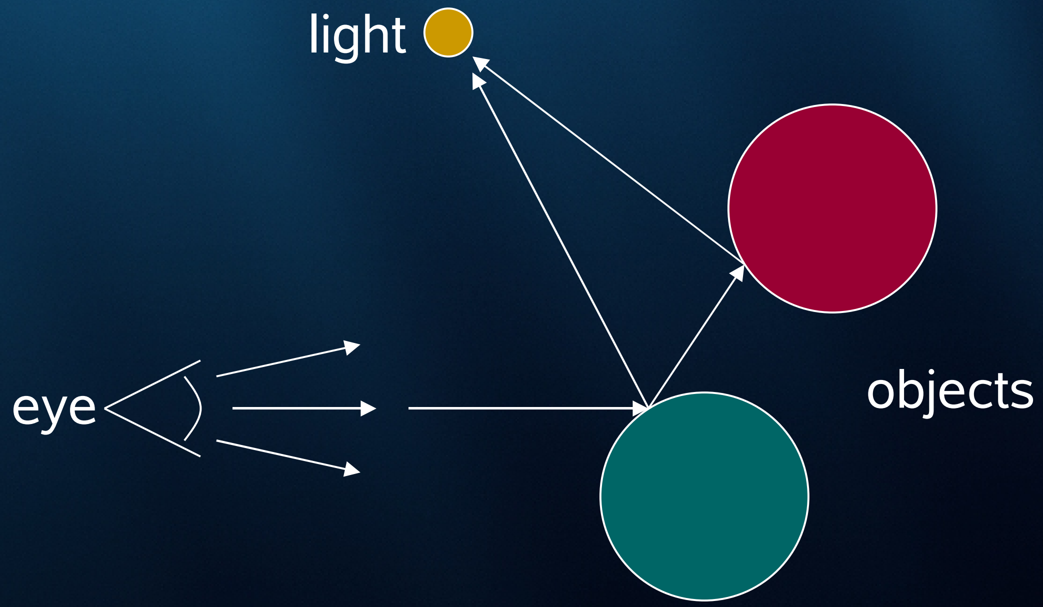
We extended RenderMan with:

- Ray tracing
- Global illumination

PIXAR

What is ray tracing?

- Recursive algorithm to compute color of a pixel [Whitted 1980]



PIXAR

Ray tracing effects

Ray tracing

- Advantages:
  - Fine shadow details
  - Interreflections
- Disadvantage: rays fly all over the scene
  - Needs all objects+textures all the time
  - Can *not* deal with very complex scenes

Goal: best of both

- Ray tracing
- Very complex scenes (as scanline)

Main question

- Some rays fly all over
- Some rays require high geometric / texture precision
- But *not all rays fly all over and require high precision!*
- Which rays require which precision?

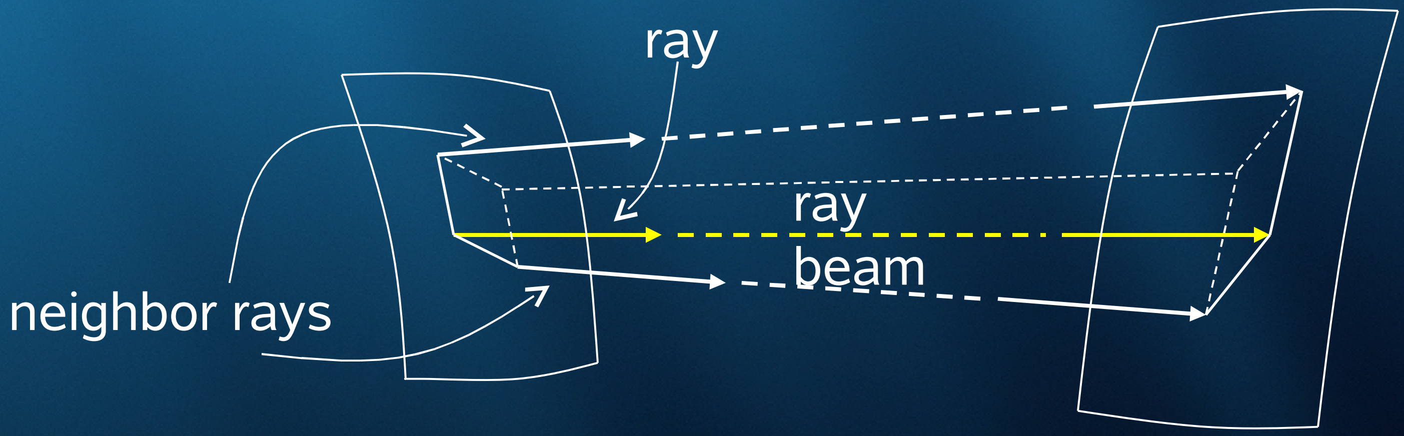
PIXAR

Ray differentials to the rescue

- Keep track of differences between "neighbor" rays
- Trace rays; each ray represents a beam [Igehy 1999]

PIXAR

Ray differentials and ray beam



- “Narrow ray”: ray beam cross-section is small
- “Wide ray”: ray beam cross-section is large

PIXAR

Ray differentials: use

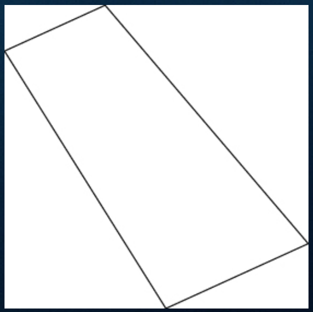
Ray differentials tell us:

- Required tessellation rate of geometry
  - Quad sizes ~ ray beam cross-section
- Required texture resolution
  - Pixel sizes ~ ray beam projected onto surface

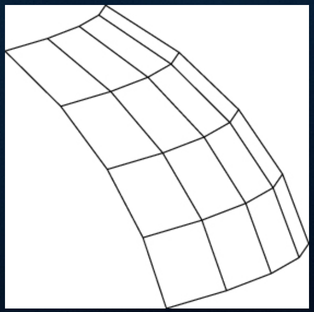
PIXAR

Multi-resolution geometry cache

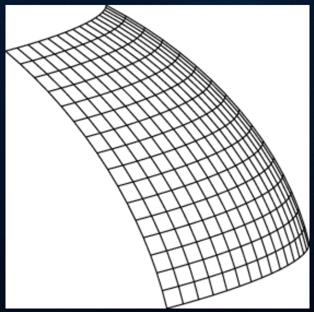
- Split objects into patches (as usual)
- Tessellate each patch on demand
- Use ray width to determine which tessellation to use:



1 quad



4x4 quads



16x16 quads

PIXAR

Multi-resolution geometry cache

- Store tessellation in coarse, medium, or fine sub-cache
- Result: can render scenes *100 x larger* than cache size !

PIXAR

Example: parking lot

15 cars; 240M quads; 80M rays

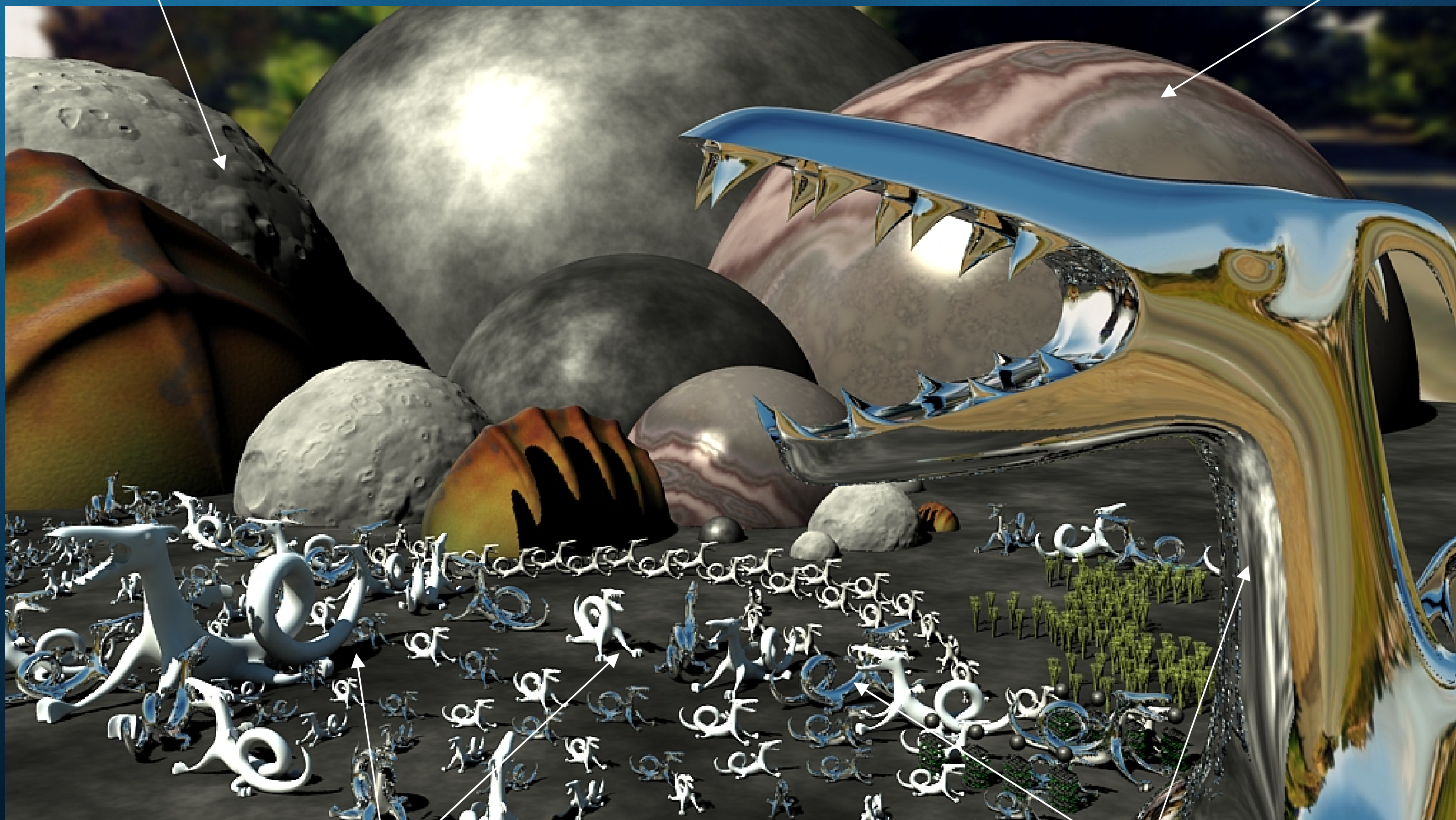
PIXAR

Parking lot: cache stats

- 1 billion geometry cache lookups
- No cache: run time > 4 days
- Single-resolution cache:
  - hit rate 97.7%
  - run time: 11 hours
- Multi-resolution cache:
  - hit rate 99.9%
  - run time: 6 hours

PIXAR

Example: 94 dragons



PIXAR

94 dragons: cache stats

- 18 million geometry cache lookups
- 3MB multi-res. cache performs well – less than 1/200 of the fully tessellated scene
- Single-res. vs. multi-res. geometry cache:
  - 1MB multi-res. cache beats 100MB single-res. cache (#recomputed vertices)

PIXAR

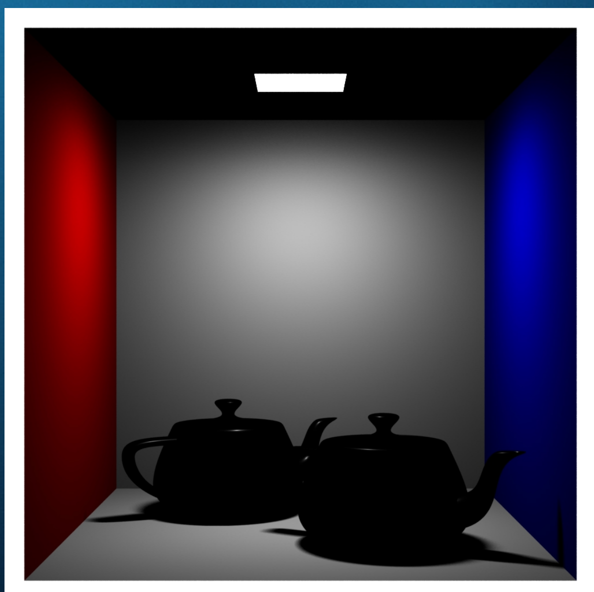
What is global illumination?

Light is reflected everywhere:

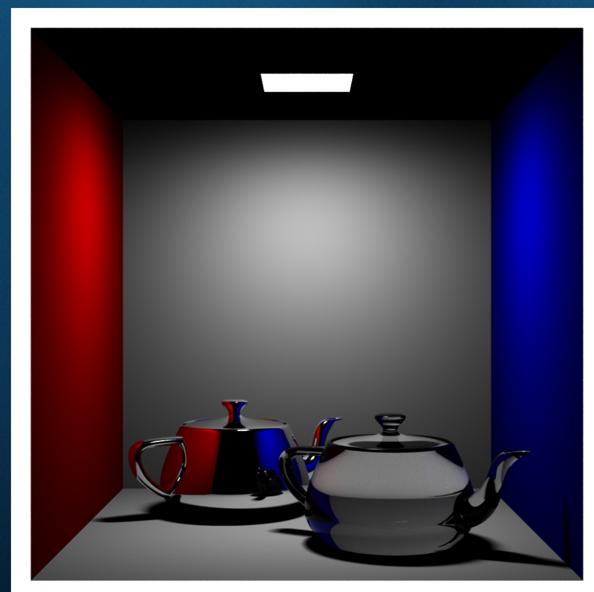
- All objects are illuminated by each other (not just by light sources)
- Hard problem:
  - infinitely many equations
  - infinitely many unknowns
- Active area of research

PIXAR

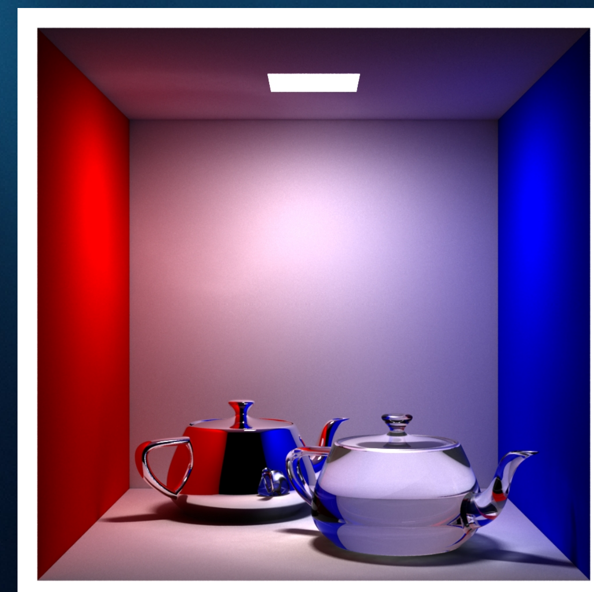
What is global illumination?



direct illumination



direct illumination + ray tracing



direct illumination + ray tracing + global illumination

PIXAR

Global illumination

Goals:

- Film-quality global illumination
- Very complex scenes

Global illumination methods

- Finite element methods
  - radiosity
- Monte Carlo simulation
  - distribution ray tracing
  - path tracing
  - bi-directional path tracing
  - photon mapping

PIXAR

Global illumination methods

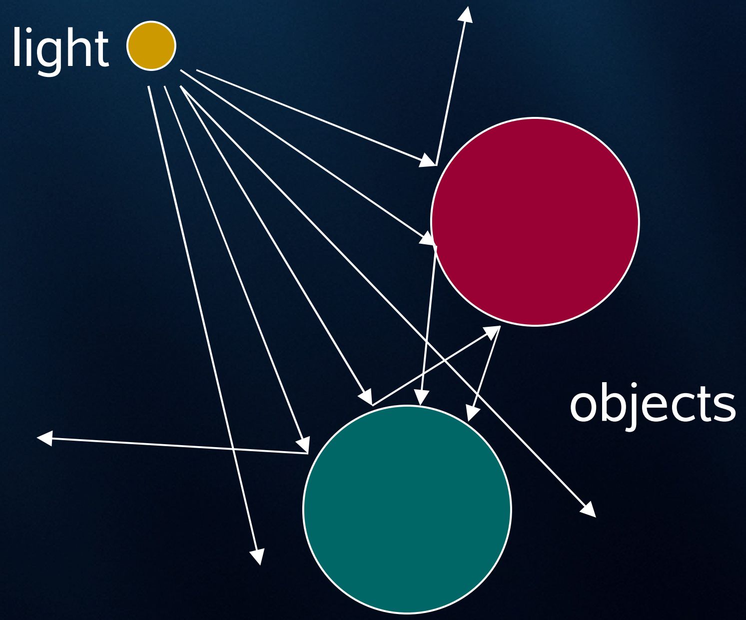
Our chosen method:

- Extend the photon mapping method
- Use a “brick map” representation of photon information

PIXAR

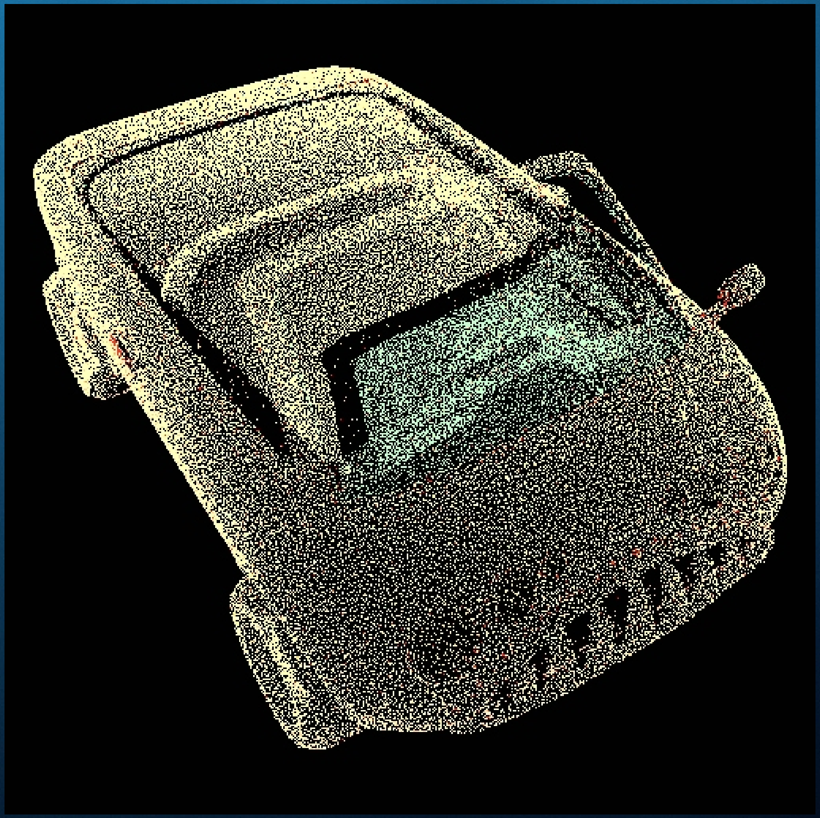
The photon mapping method

- Original photon map [Jensen 1996] 3 steps:
  - emit, trace, store photons
  - sort photon map (kd-tree)
  - render

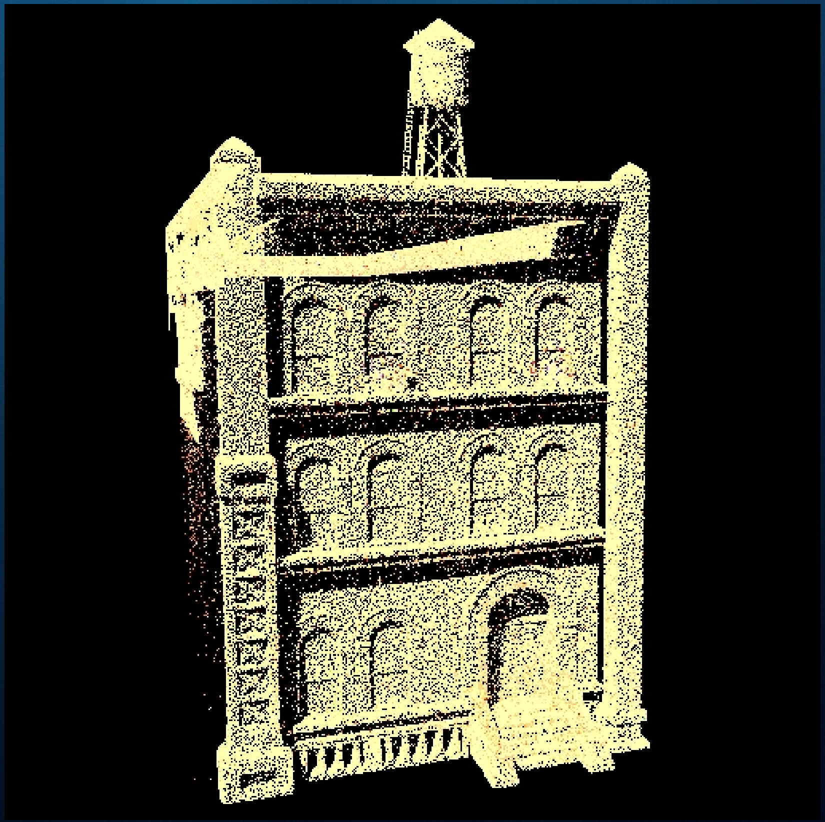


PIXAR

Photon maps



76,000 photons



3.4 million photons

PIXAR

The photon mapping method

- Very general and flexible:
  - all types of reflection
  - all types of geometry
- Relatively fast
- But: cannot deal with more photons than memory!

PIXAR

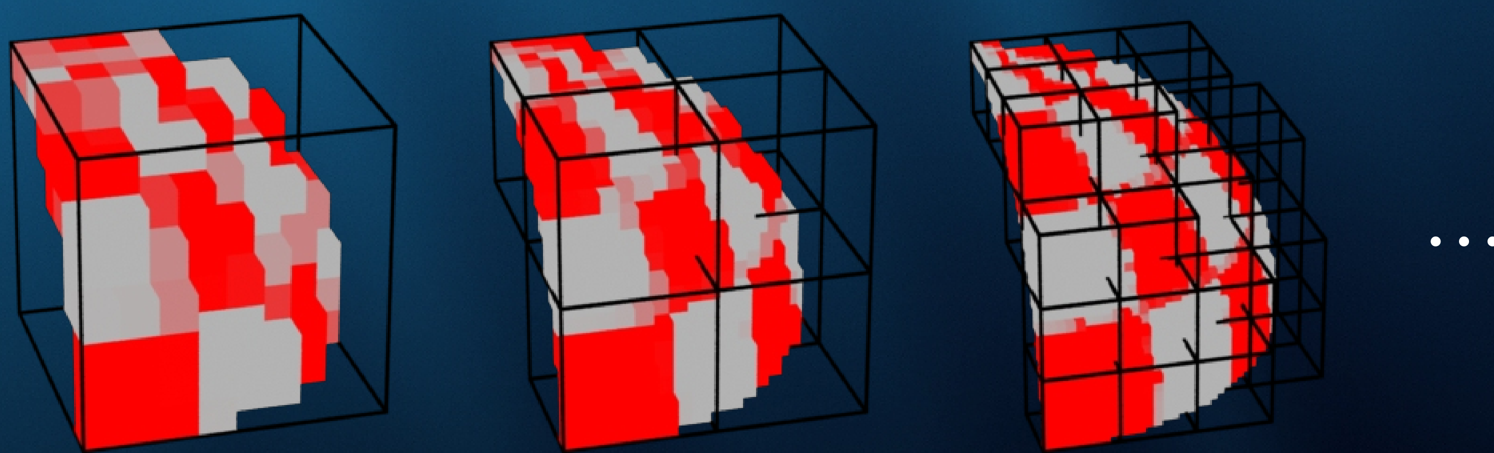
The brick map

- Tiled, 3D MIP map representation of surface and volume data:
- Adaptive octree with a *brick* in each node
- A brick is a 3D generalization of a tile; each brick has 8^3 voxels

PIXAR

Brick map example

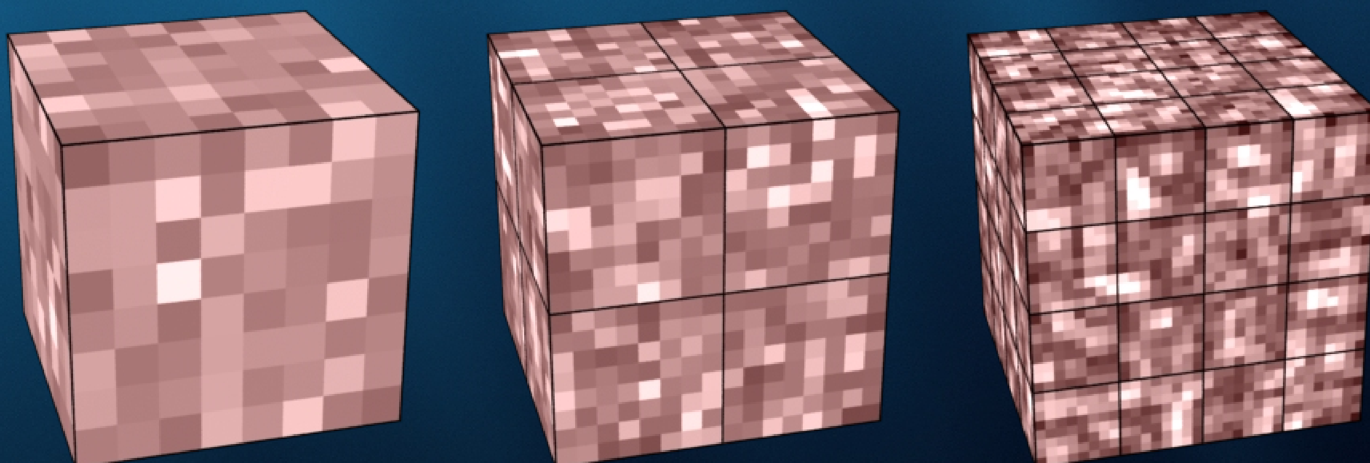
Sparse brick map for surface data:



Level 0 | Level 1 | Level 2

Brick map example #2

Dense brick map for volume data:



Level 0

Level 1

Level 2

. . .

PIXAR

The brick map

- Can be used for general data; surface or volume
- Here we use it for illumination on surfaces

Test scene: direct illumination

118 million quads (render time: 6 min)

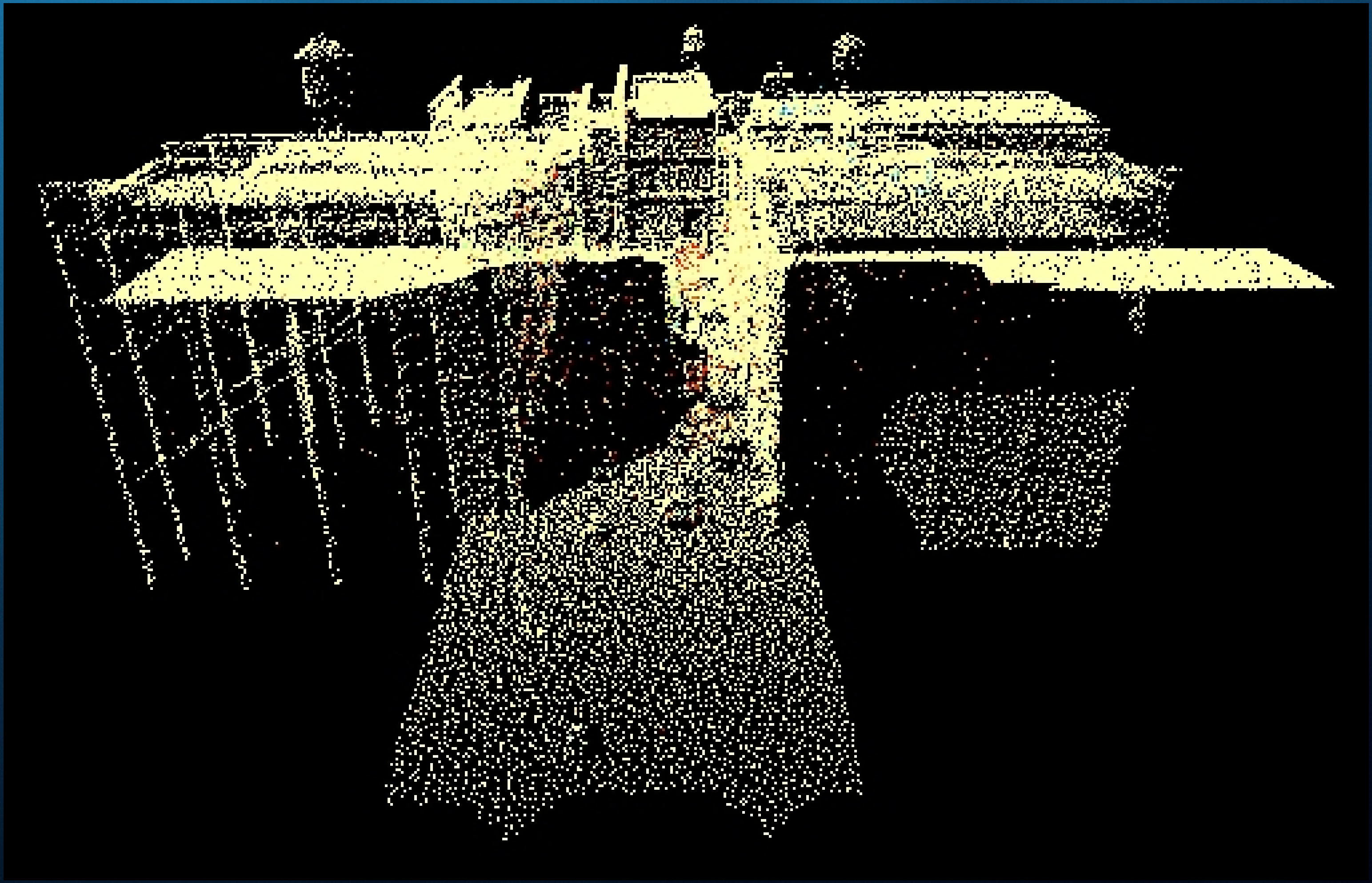
PIXAR

Global illumination using brick maps

Step 1:

- Emit + trace photons as usual, but write to photon map files
- Collection of photon maps: *photon atlas*

Photon atlas



52 million photons (0.1%)

PIXAR

Photon tracing

- Photon tracing: 29 minutes
- Stored 52 million photons in 41 photon map files (2.2 GB)

Global illumination using brick maps

Step 2:

- Estimate irradiance at photon locations
- Construct a brick map from each photon map
- Collection of irradiance brick maps: *irradiance atlas*

PIXAR

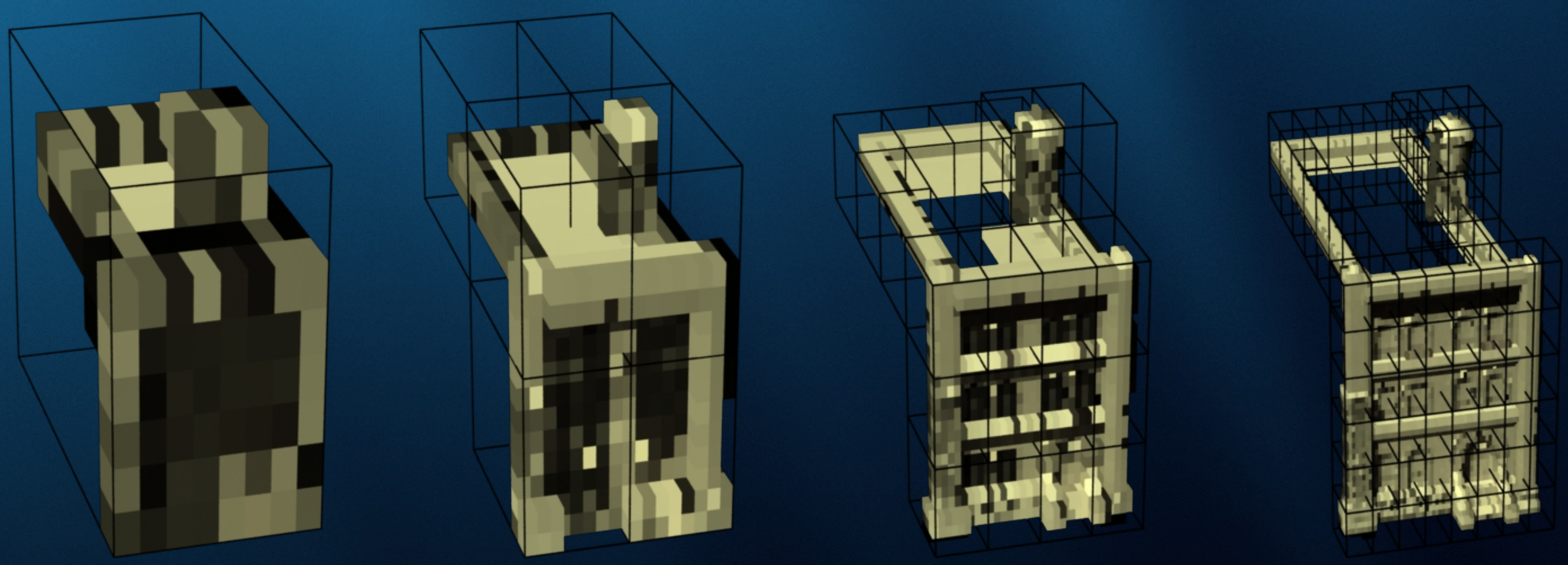
Irradiance brick map for car

960 bricks (69 MB)

PIXAR

Irradiance brick map for building

31,700 bricks (190 MB)



Level 0 | Level 1 | Level 2 | Level 3

PIXAR

Irradiance atlas

253,000 bricks (2.4 GB): 200x brick cache capacity

PIXAR

Generating the irradiance atlas

- Estimating irradiance at all photon positions (using nearest 50 photons): 18 minutes
- Computing irradiance brick maps: 25 min.

Irradiance times diffuse color

Global illumination using brick maps

Step 3:

- Render using final gather
- At final gather hit points: look up in irradiance atlas
- Read bricks from file on demand
- Store in brick cache (10MB, LRU replacem.)

Rendering: final image

77 million rays; 3.8 hours

PIXAR

More information …

- Book: *Advanced RenderMan*
- “Ray differentials and multiresolution geometry caching for distribution ray tracing in complex scenes”, Eurographics 2003
- “An irradiance atlas for global illumination in complex production scenes”, Eurographics Symposium on Rendering 2004

Conclusion (part 1)

- Use multi-resolution geometry cache
- Use multi-resolution texture cache
- Use ray differentials to select resolution

PIXAR

Conclusion (part 2)

- Introduced the brick map, a tiled 3D MIP map format for general data
- Improved the photon map method with efficient representation + caching of global illumination data
- Result: Can now render ray tracing and global illumination in production scenes – same complexity as scanline !

PIXAR

Acknowledgments

Thanks to:

- RenderMan team
- Andreas Baerentzen and Bent Larsen for inviting me
- You for listening

PIXAR

Questions?

PIXAR