Global Illumination Across Industries

Ray Tracing vs. Point-Based GI for Animated Films

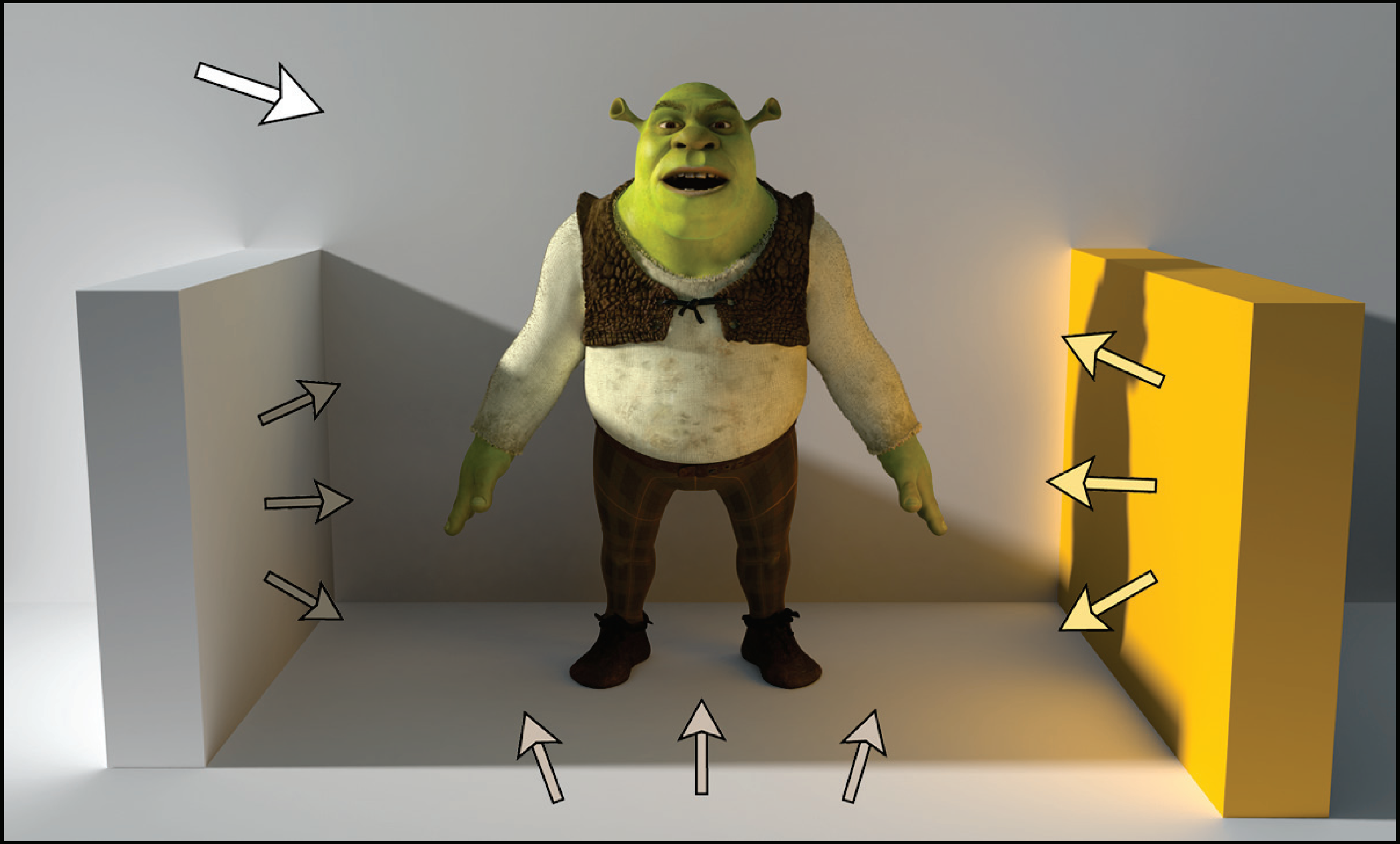
Eric Tabellion
et@pdi.com

Introduction

- Global Illumination (GI) usage at DreamWorks Animation
  - Bounce lighting
  - Ambient occlusion
  - HDR Environment map lighting (IBL)
  - Used on many films since “Shrek 2”
    - Used in a variety of shots
    - Various characters, props and environments
- Global Illumination system
  - Using Ray Tracing and Irradiance Caching (IC)
    - [Tabellion and Lamorlette 2004]
    - [Krivanek et al. 2008]
  - Using point-based GI
    - [Christensen 2008]

Direct Lighting Only

Direct + Indirect Lighting



Example: “How To Train Your Dragon ”

Example: “How To Train Your Dragon ”

Example: "Shrek Forever After"

GI in Animated Film Production

- Requirements:
  - High Quality
    - No noise, buzzing or popping
  - Artistic control
    - Offer physically correct starting point
    - Let the user tweak shaders further
      - Good shader integration
      - Speedy interaction
    - Add bounce cards to control lighting
    - Apply localized filters to GI
  - Scene complexity
    - Scene complexity is unbounded (more or less tuned)

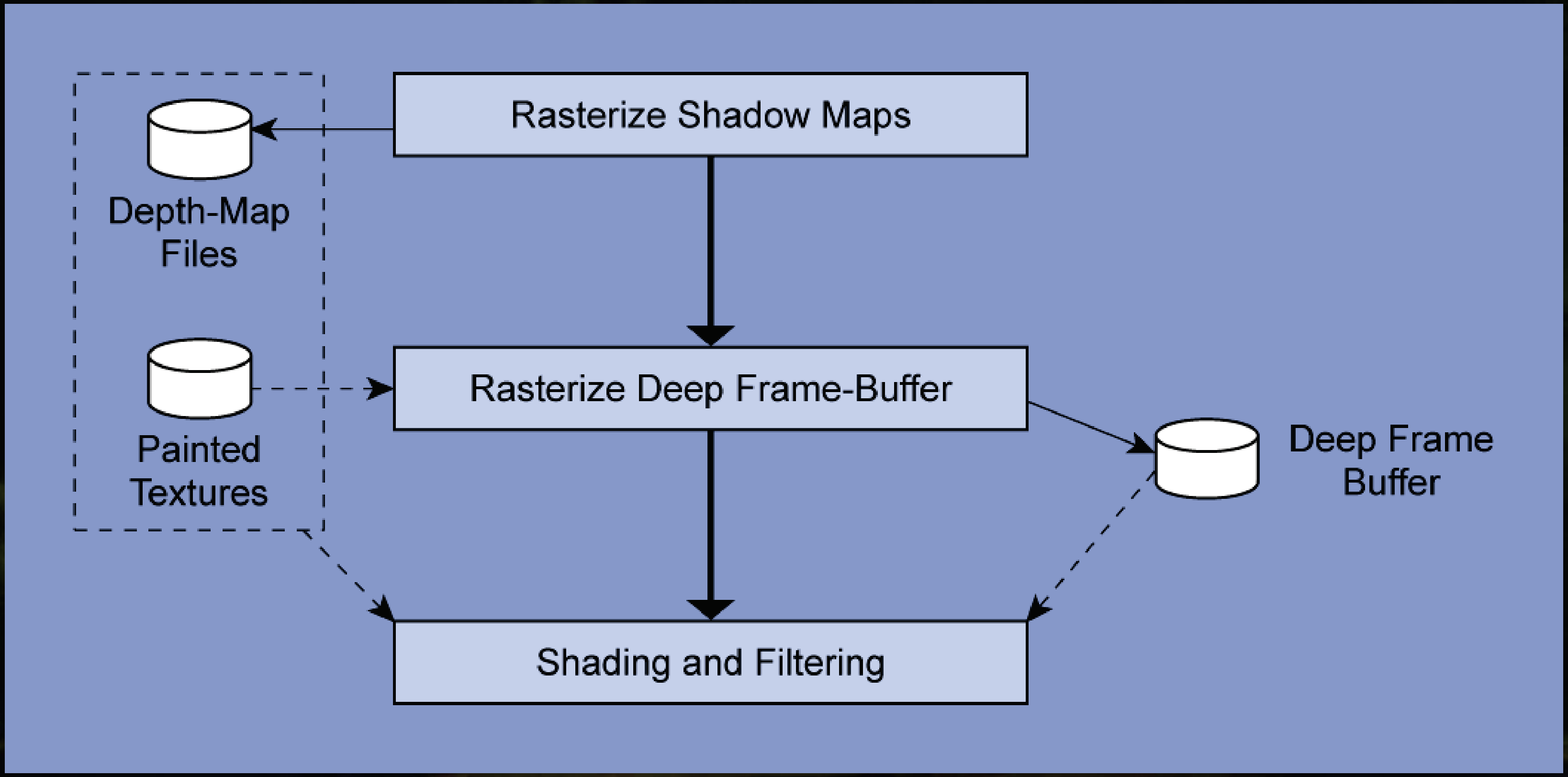
GI in Animated Film Production

- Multi-department Impact:
  - Modelling & Animation
    - Geometry with bad contact / penetrations
  - Surfacing
    - Account for GI as a lighting scenario
    - Define bouncing characteristics
  - FX
    - Effects do illuminate and occlude
  - Lighting
    - GI not always intuitive
      - Light is coming from everywhere
    - GI produces simpler lighting rigs
    - Propagates well to other shots / sequences

Ray-Tracing-Based GI

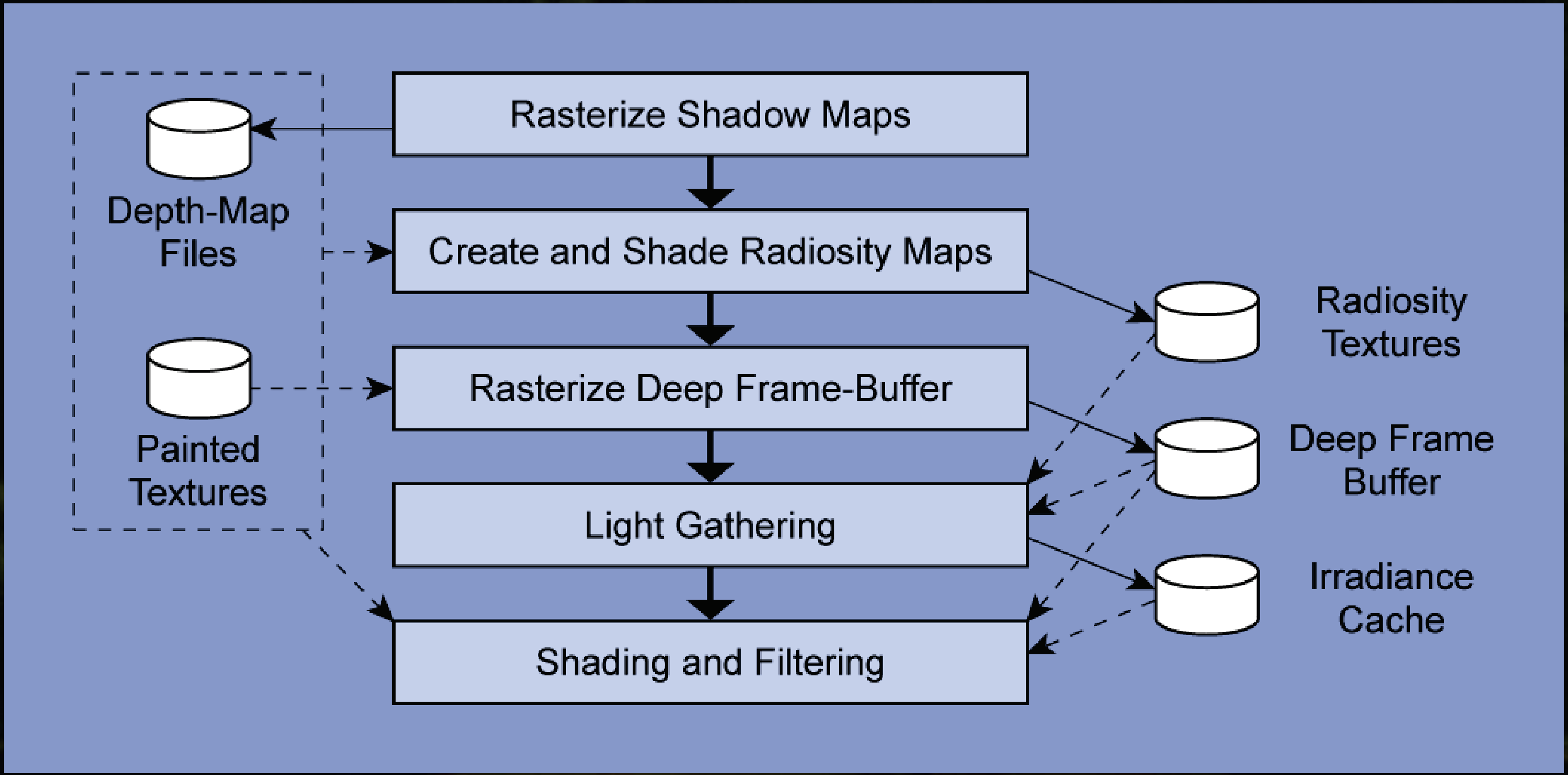
System Overview

- Micropolygon Deep Frame Buffer Renderer
- Use Ray Tracing and Irradiance Caching
- Lighting Tool for Interactive lighting



System Overview

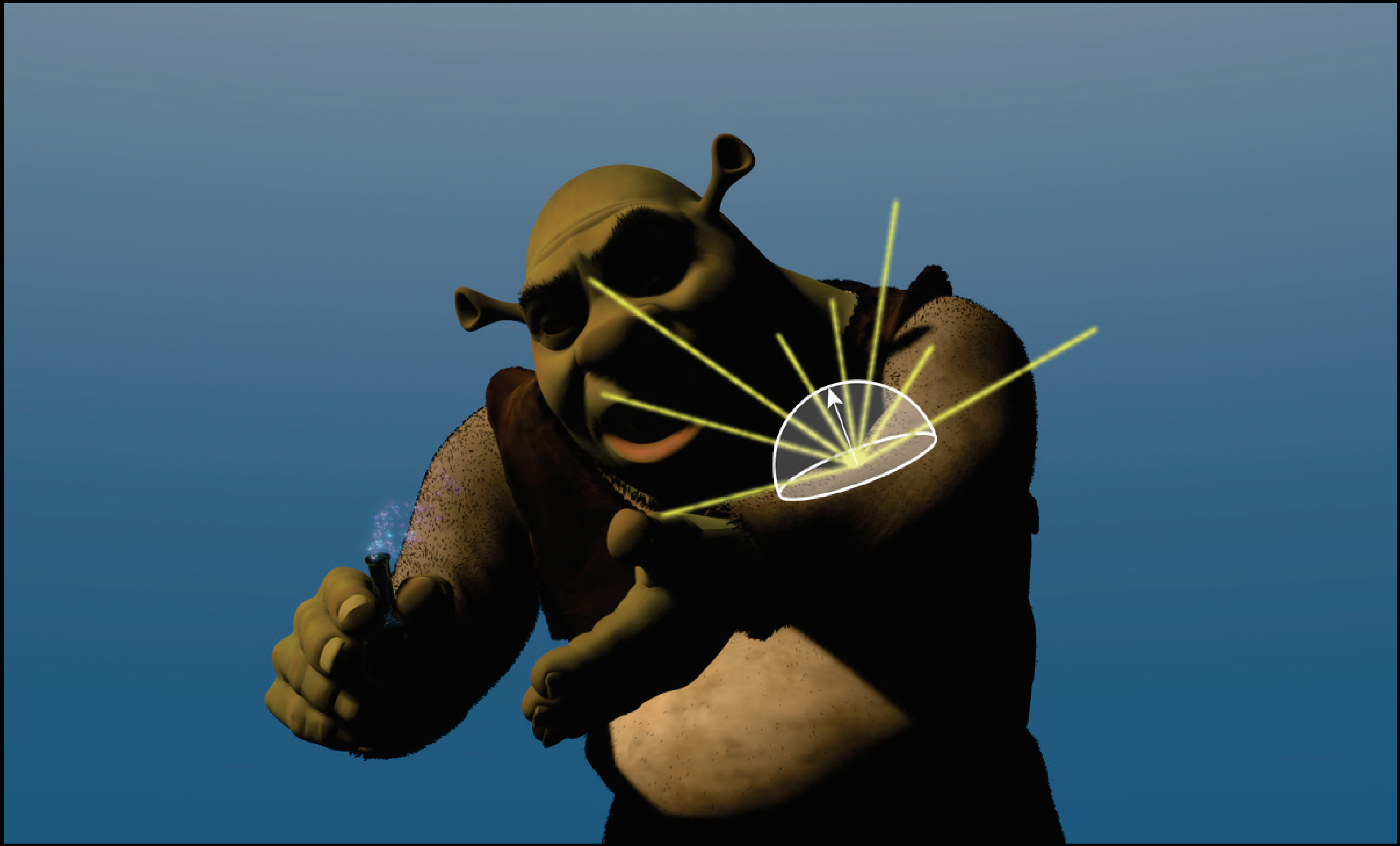
- Micropolygon Deep Frame Buffer Renderer
- Use Ray Tracing and Irradiance Caching
- Lighting Tool for Interactive lighting



Direct Lighting Only

Credit: "Shrek 2"

Light Gathering



Credit: “Shrek 2”

Indirect Lighting Only

Credit: "Shrek 2"

Indirect + Direct Lighting

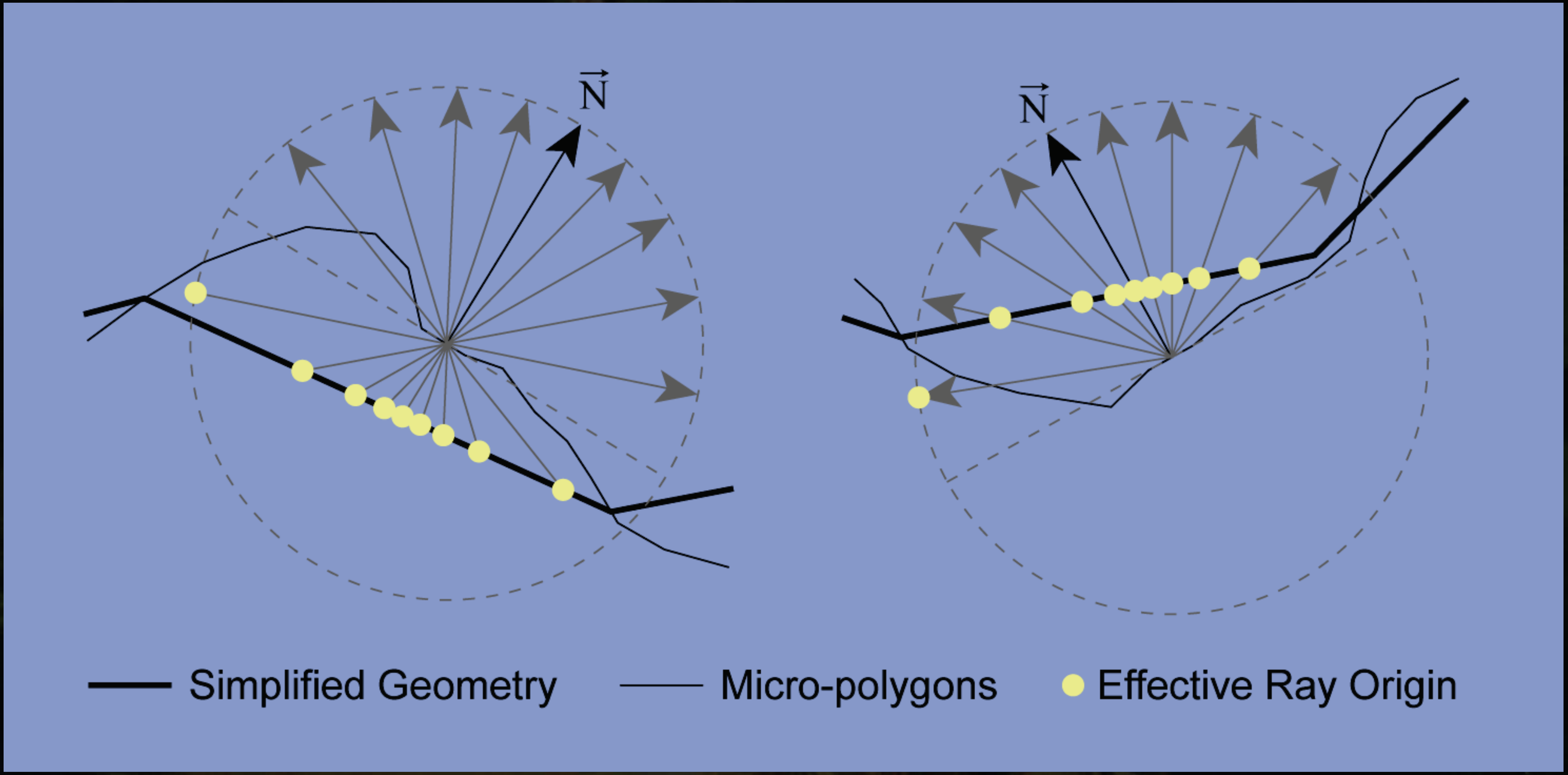
Credit: "Shrek 2"

Geometry 2-Levels of Detail 1/3

- Primary visibility: rasterize micropolygons
- Secondary visibility
  - Raytrace coarser tessellation and no/coarse fur
  - Greatly increases renderable scene complexity
  - Use “smart bias”

Geometry 2-Levels of Detail 2/3

- Ray Origin Offsetting Algorithm



Geometry 2-Levels of Detail 3/3

Raytracing 2 million micropolygons

Raytracing 2 thousand polygons

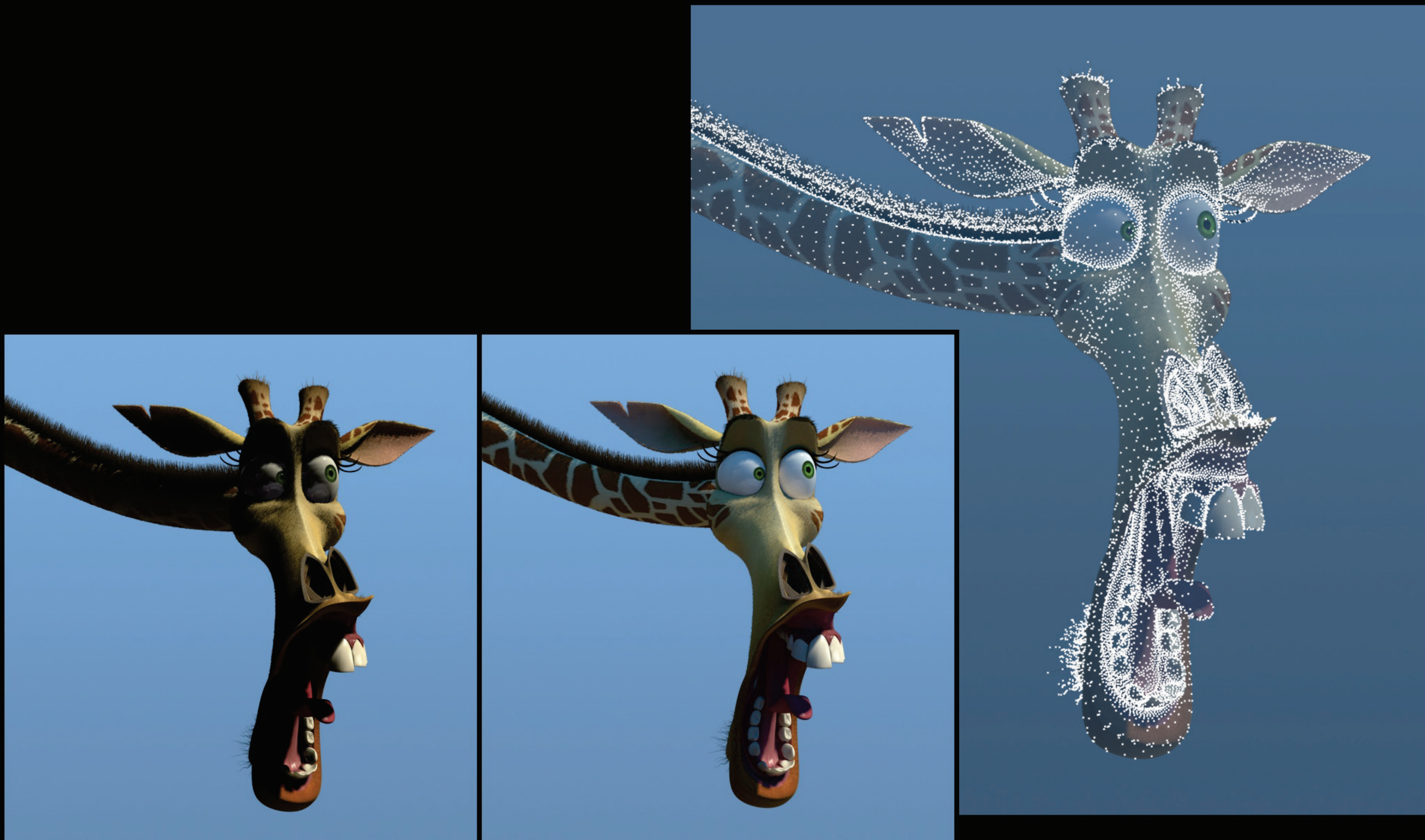
Geometry Multiresolution LOD

- Multiple coarser levels of detail
  - [Christensen 2003]
- Use even coarser tessellations
- Use for larger ray footprints
- Problems:
  - Based on tessellation LOD only
  - Often need less polys than primitives

Irradiance Caching (IC)

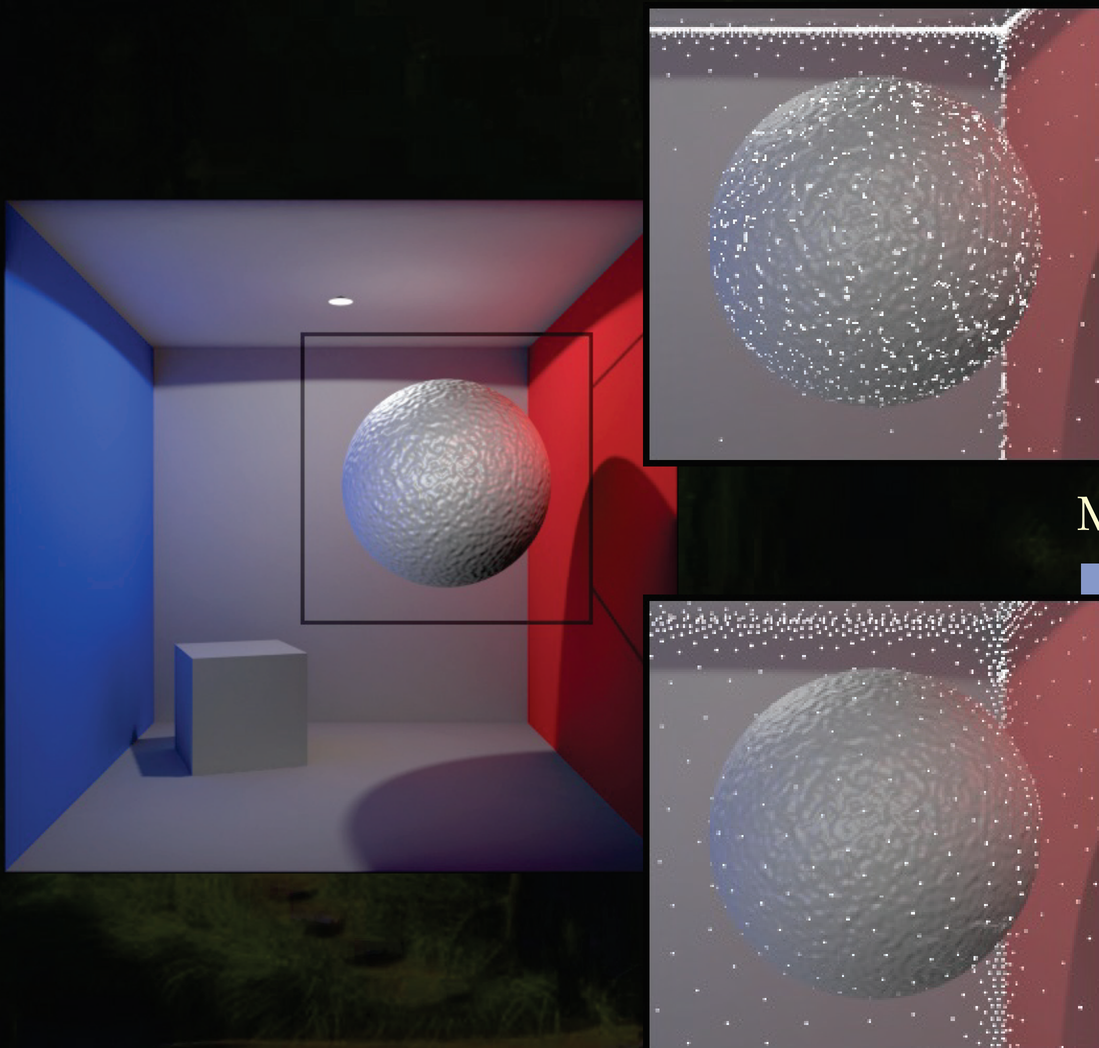
Credit: "Shrek 2"

Irradiance Caching (IC)



Credit: “Madagascar: Escape 2 Africa”

IC Error Metric



[Ward and Heckbert 1992]

$$w_i(\vec{p},\vec{n}) = \frac{1}{\frac{\|\vec{p}-\vec{p}_i\|}{R_i} + \sqrt{1-\vec{n}\cdot\vec{n}_i}}$$

Modified, screen-space dependent

$$w_i(\vec{p},\vec{n}) = 1-\kappa\cdot\max\left(\varepsilon_{pi}(\vec{p}),\varepsilon_{ni}(\vec{n})\right)$$

$$\varepsilon_{pi}(\vec{p}) = \frac{\|\vec{p}-\vec{p}_i\|}{\max\left(\min\left(\frac{R_i}{2},R_+\right),R_-\right)}$$

$$\varepsilon_{ni}(\vec{n}) = \frac{\sqrt{1-\vec{n}\cdot\vec{n}_i}}{\sqrt{1-\cos(\alpha_+)}}$$

IC Record Spacing

- Record spacing flickers
- Interpolated irradiance does not
  - Requires stable gradients
  - Rays per Record > Rays per Pixel
- IC still a win if:
  - Total rays with IC < Total rays without IC
- Challenge: keep spacing sparse
  - Even with high frequency detail: bump, displacement, fur, foliage, etc.
  - Use knowledge from the scene

IC with Displacement

Credit: “Shrek The Third”

IC with Displacement

Credit: "Shrek The Third"

IC with Displacement

Credit: “Shrek The Third”

IC with Fur / Grass

Credit: "Kung Fu Panda"

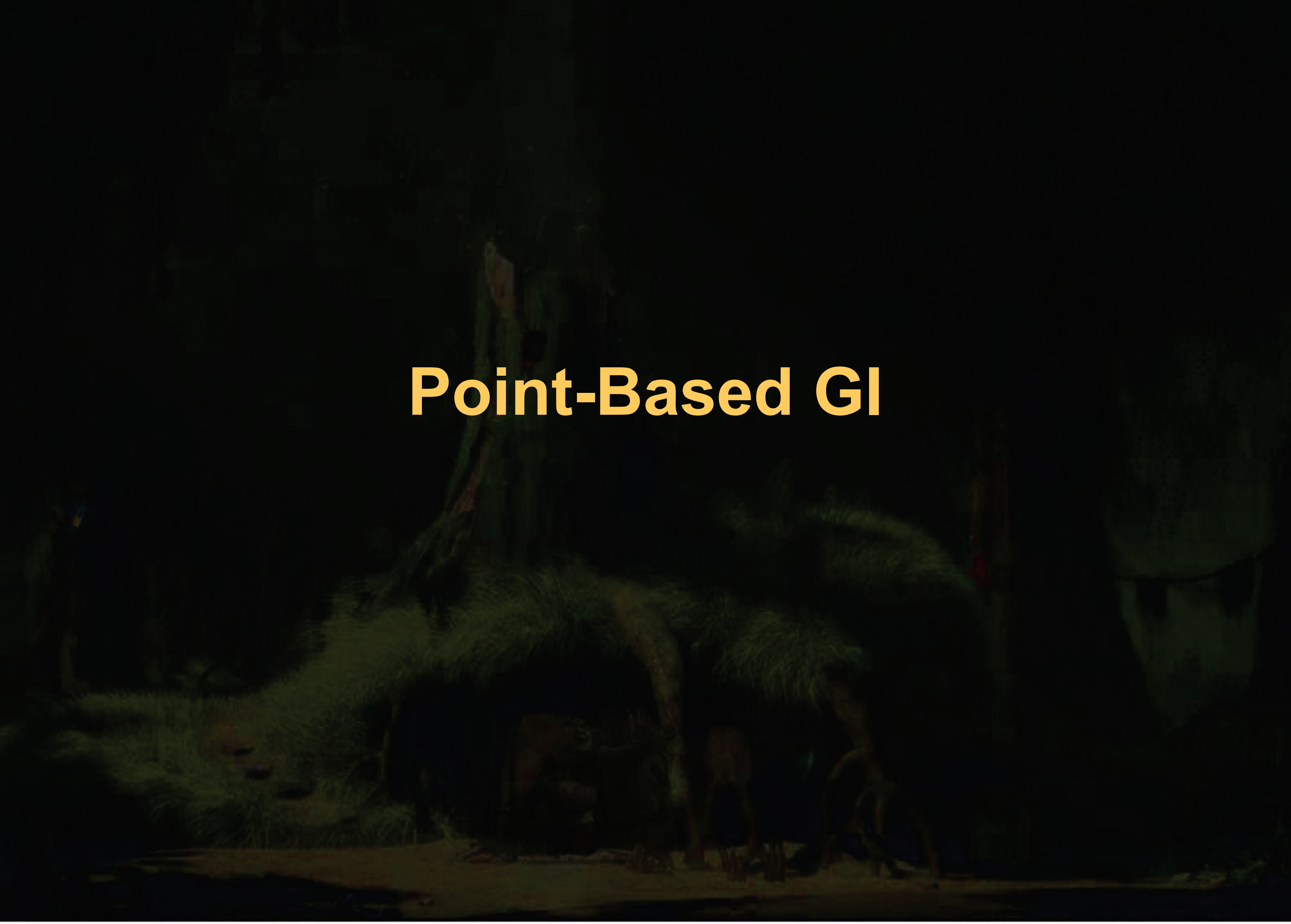
IC with Fur / Grass

Credit: "Kung Fu Panda"

Irradiance Caching Performance

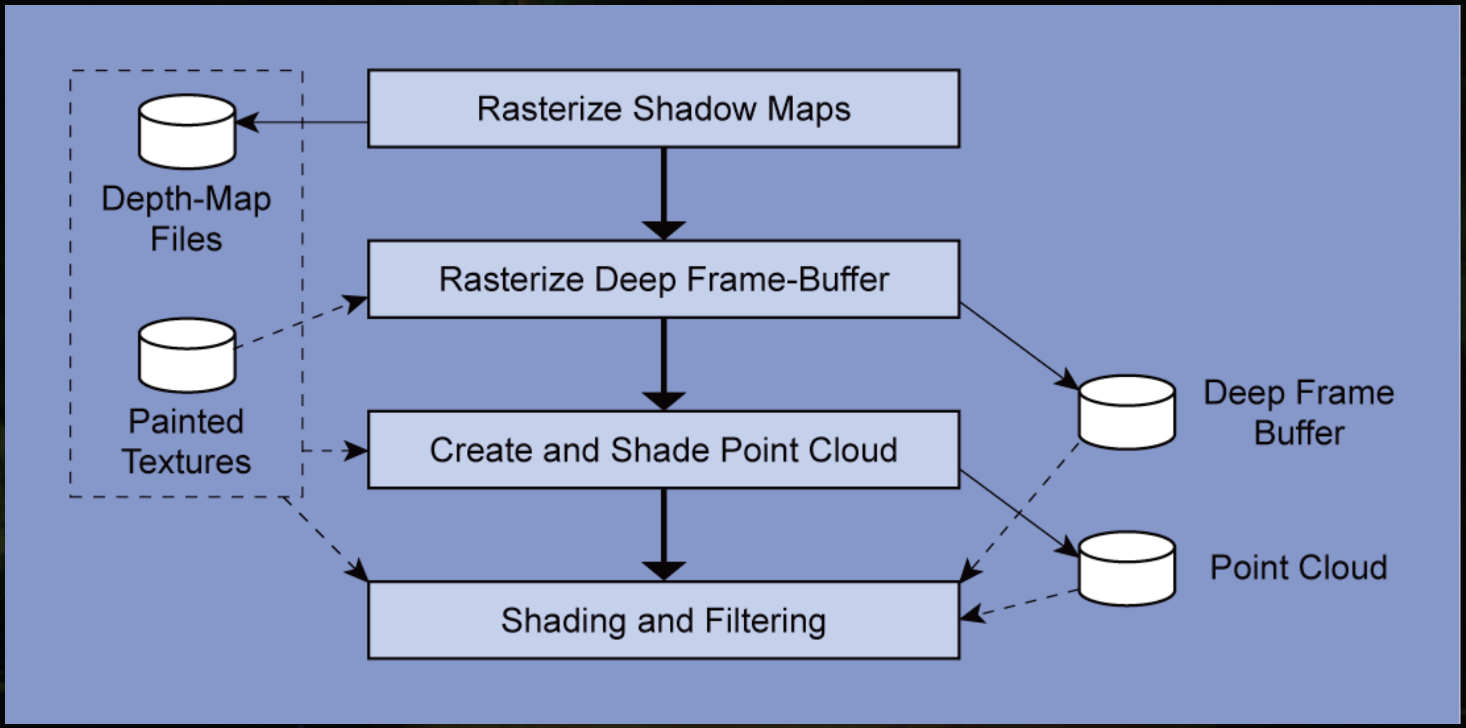
	Gathering	Shading	Total	Rays / Record	Cache Records	Rays / MP
Shrek AO - 1/3 res – wo IC	6:39	-	6:39	-	-	250
Shrek AO - 1/3 res	0:58	0:13	1:11	453	8282	22
Shrek AO - Film res	2:33	0:54	3:27	453	19294	7
Melman - 1/3 res	2:59	1:21	4:20	453	8559	18
Melman - Film res	6:44	4:03	10:47	453	16799	9
Shifu - 1/3 res	6:01	3:01	9:02	1000	17924	34
Shifu - Film res	15:48	9:32	25:20	1000	38187	22

Point-Based GI

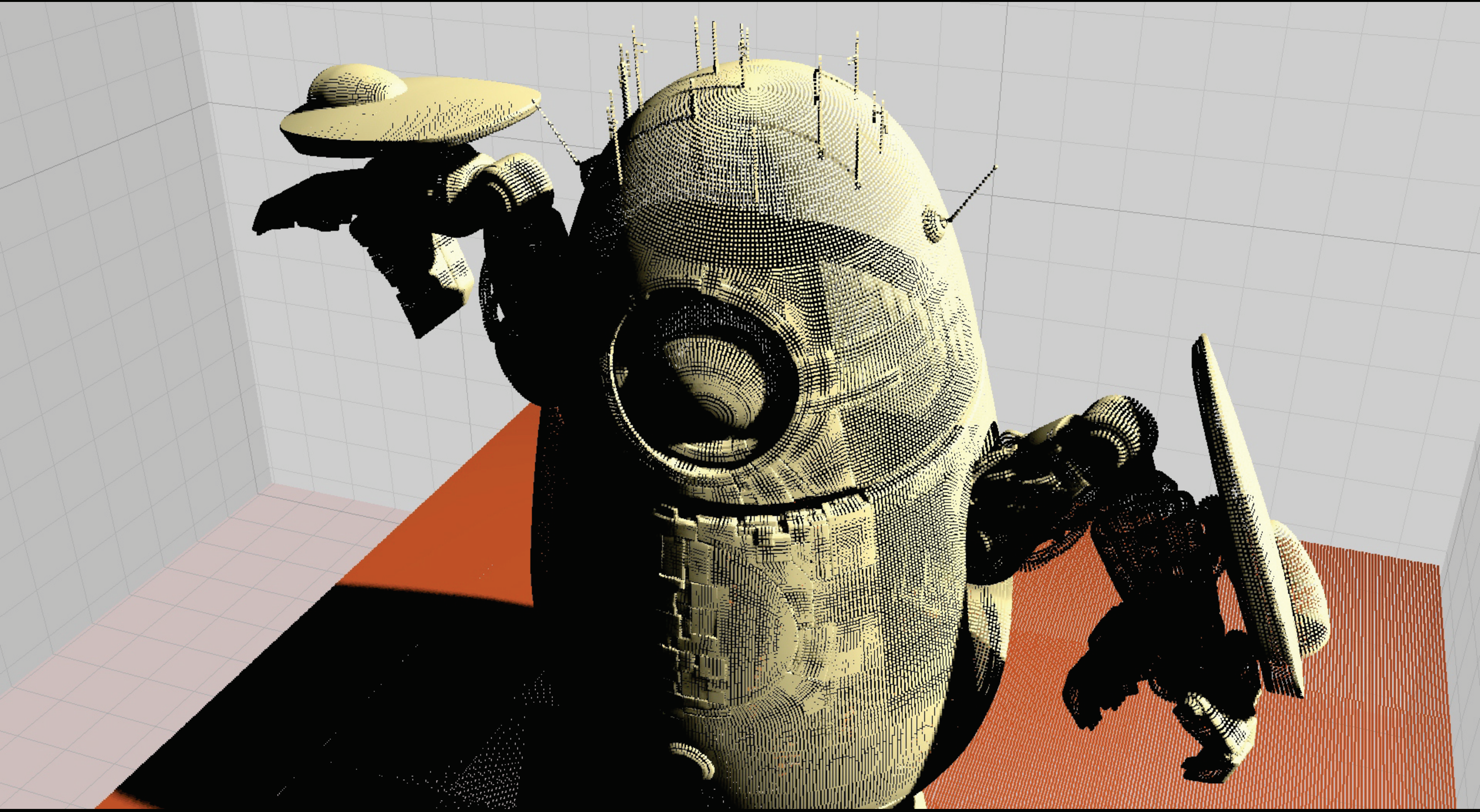


Point-Based GI (PBGI)

- Implementation basics [Christensen 2008]
- Same integration as with Ray Tracing
- Easy to switch Ray Tracing / Points

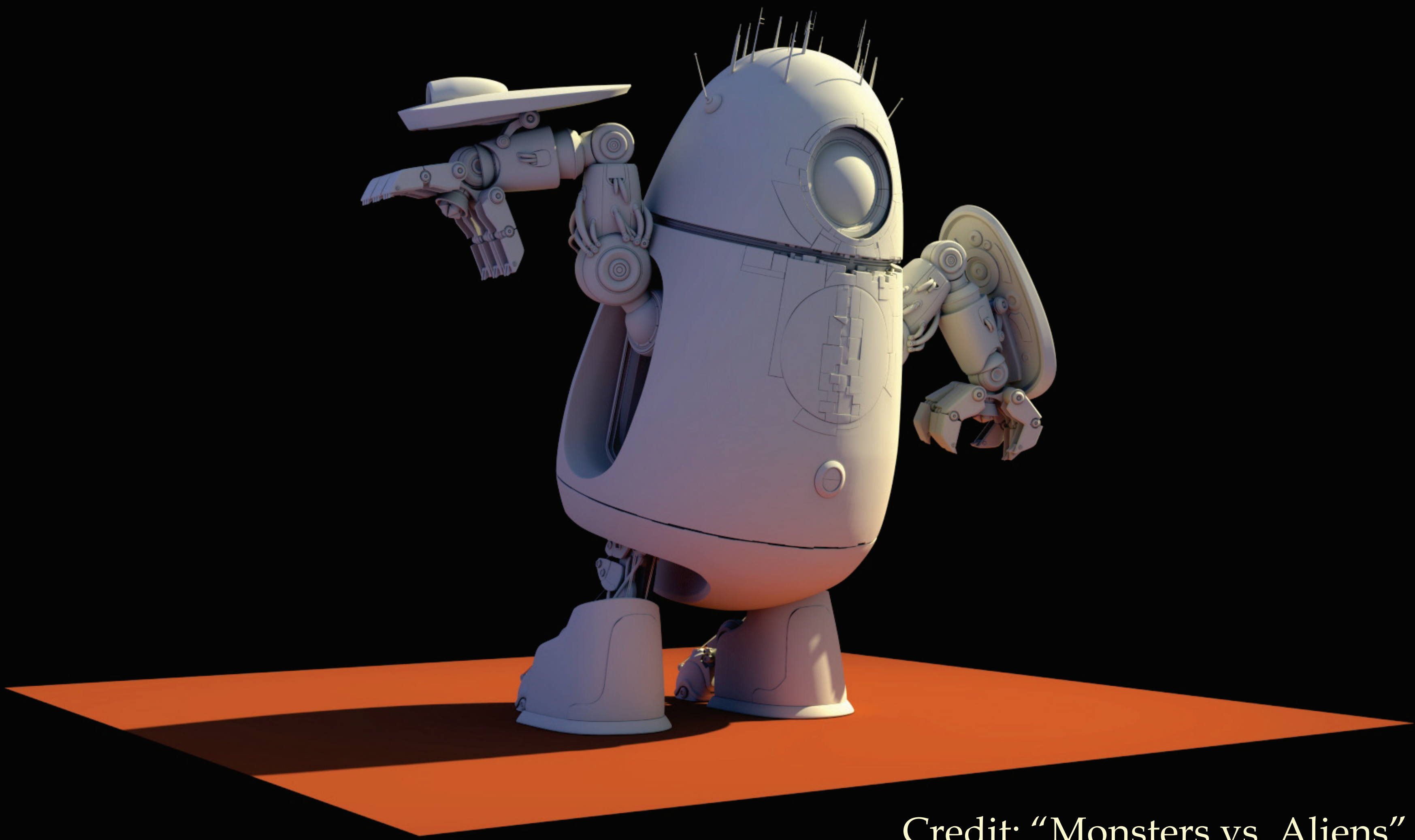


Point-Based GI (PBGI)



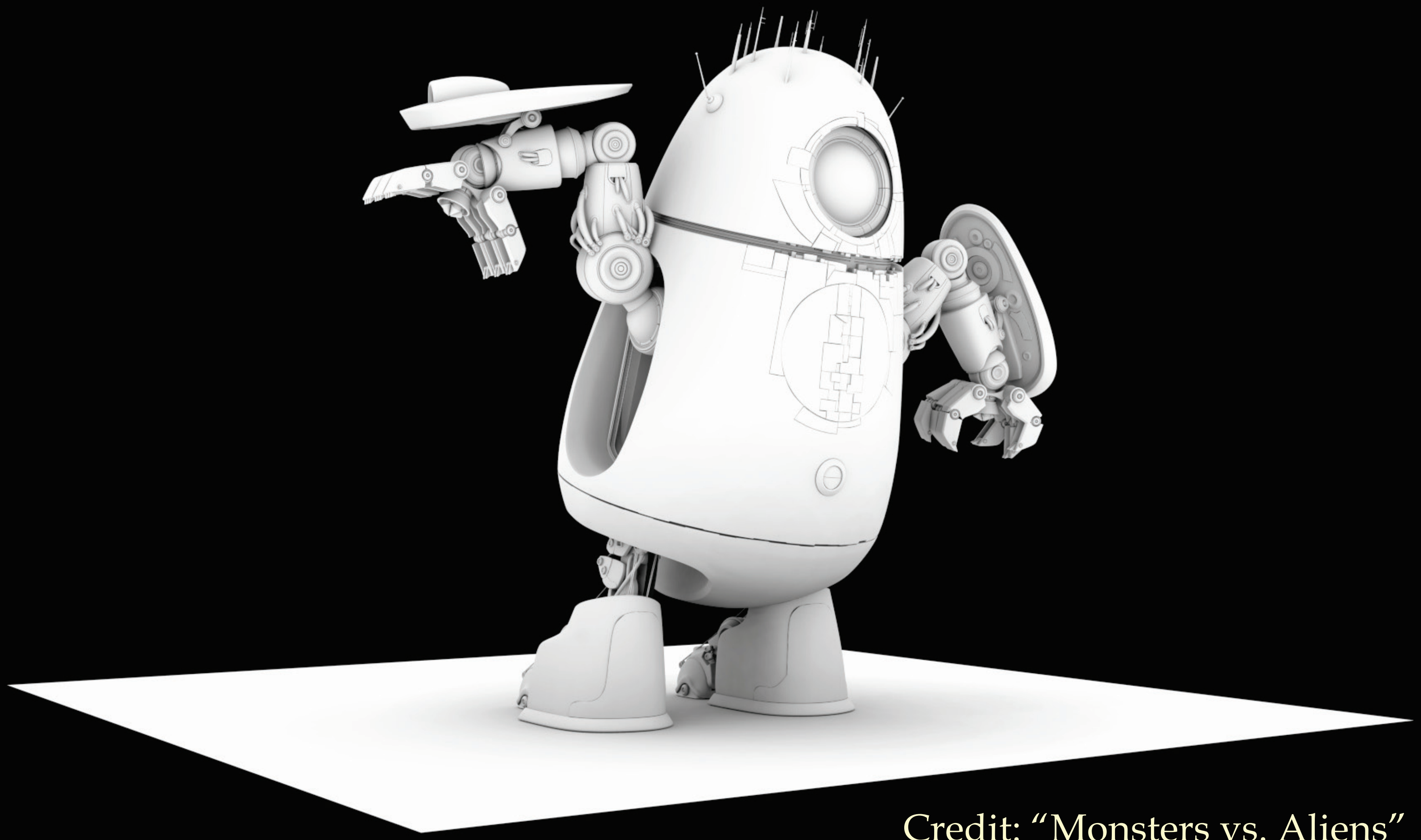
Model credit: "Monsters vs. Aliens"

PBGI Bounce + Environment fill



Credit: “Monsters vs. Aliens”

PBGI Ambient Occlusion



Credit: “Monsters vs. Aliens”

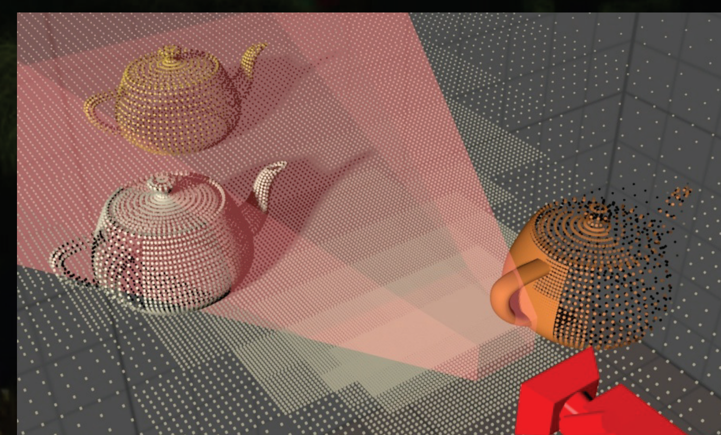
Point Generation 1/5

- Use micropolygon “split & dice” tessellator
- Generate one point per micropolygon
- Need points everywhere
- Include FX point clouds
  - Fire, explosions, etc.
- Optimal point count for maximum LOD
  - In-view: Pixel-size points
  - Out-of-view: Much fewer / bigger points
- Change tessellation
  - Disable culling
  - Cannot use perspective projection
  - Work in camera space

Point Generation 2/5

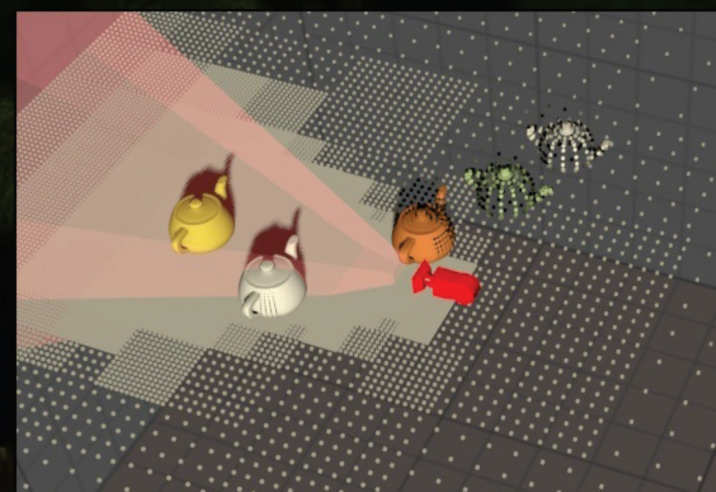
- Solid angle to surface area
- Surface in-view:
  - d = Distance from focal point
  - ω = “center pixel” solid angle
- Solve desired point area
  - Ignore surface orientation
  - A = ω * (d*d)
- Compute surface size
- Derive dice rate

$$\omega = \frac{A \cdot \cos\theta}{d^2}$$



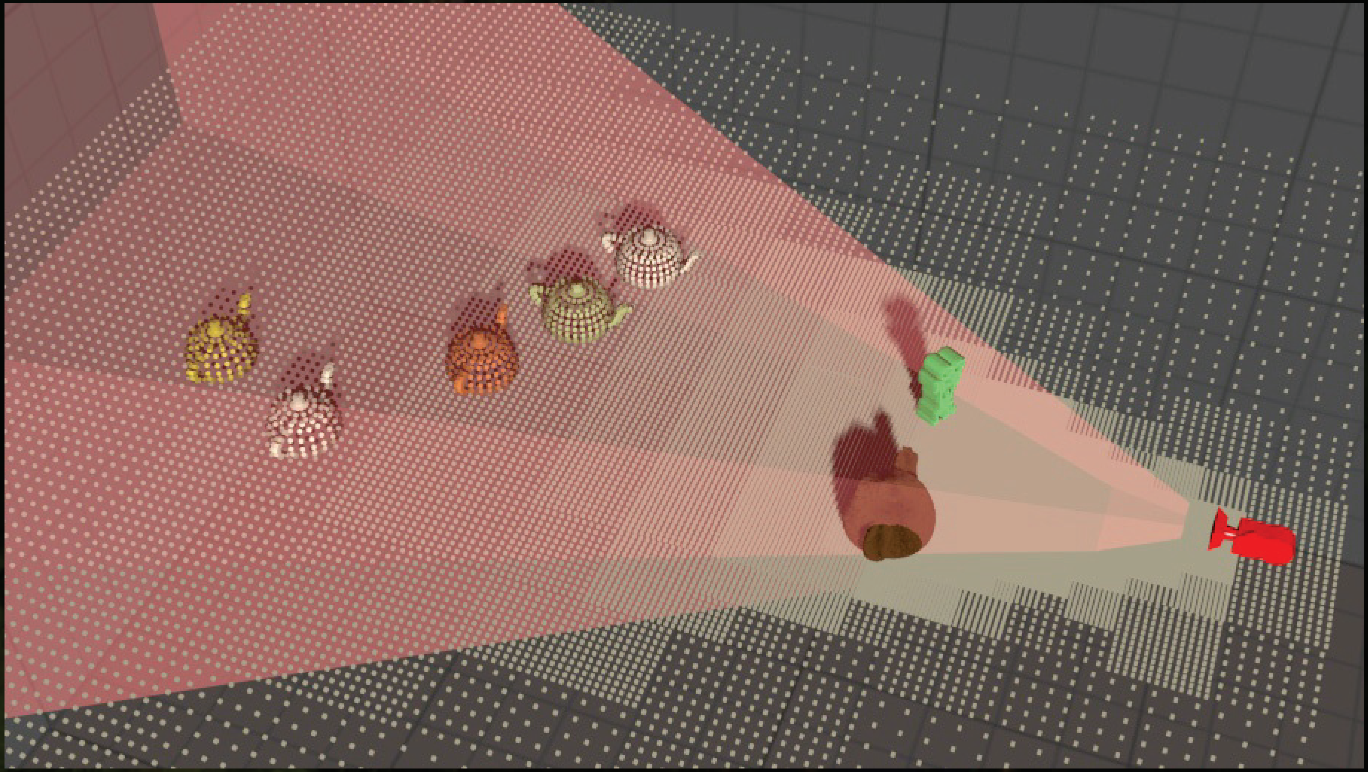
Point Generation 3/5

- Surface Out-of-view:
  - d′ = Distance to closest point in-view
    = Closest point on view frustum polyhedra [Mirtich 1998]
  - ω′ = "max-solid angle" LOD quality
- Solve desired point area
  - Ignore surface orientation
  - $A' = \omega' * (d' * d')$
- Compute surface size
- Derive dice rate



Point Generation 4/5

- Show Video!



Point Generation 5/5

- Do it twice for stereo cameras
- No need to stitch
- Need less points than primitives ?
  - Apply "Russian-roulette"
- Modified geometry generators
  - Fur, Grass, Tree
  - Generate less primitives
  - Generate larger points
- FX particles
  - Fire, explosions, etc.

Point Clustering

- Fast Spherical Harmonics point projection
  - Point = oriented and scaled “cosine lobe”
  - Radially symmetric around Z axis
  - Use Zonal Harmonics rotation [Sloan et al. 2005]
- Use cluster average position
  - More accurate than octree-cell center
  - Less clusters “above the horizon”
  - Smaller cuts

PBGI Bounce: "Shrek Forever After"

PBGI Fur AO Example

Credit: “How To Train Your Dragon”

PBGI Foliage AO Example

Credit: “How To Train Your Dragon”

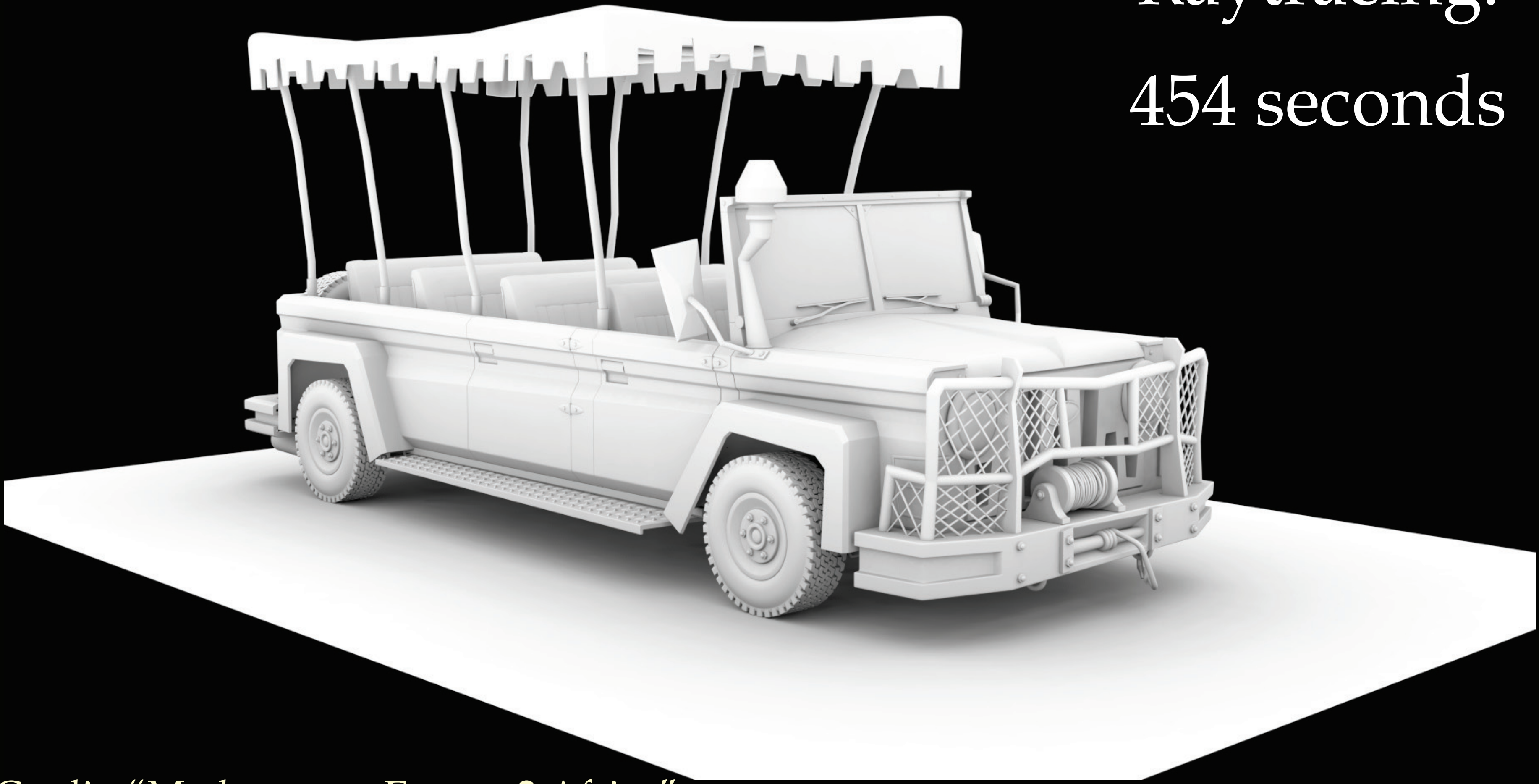
PBGI Fire Lighting Example

Credit: "How To Train Your Dragon"

Dragons Fire Lighting Example

Credit: “How To Train Your Dragon”

PBGI vs. Raytracing



Raytracing:
454 seconds

Credit: “Madagascar: Escape 2 Africa”

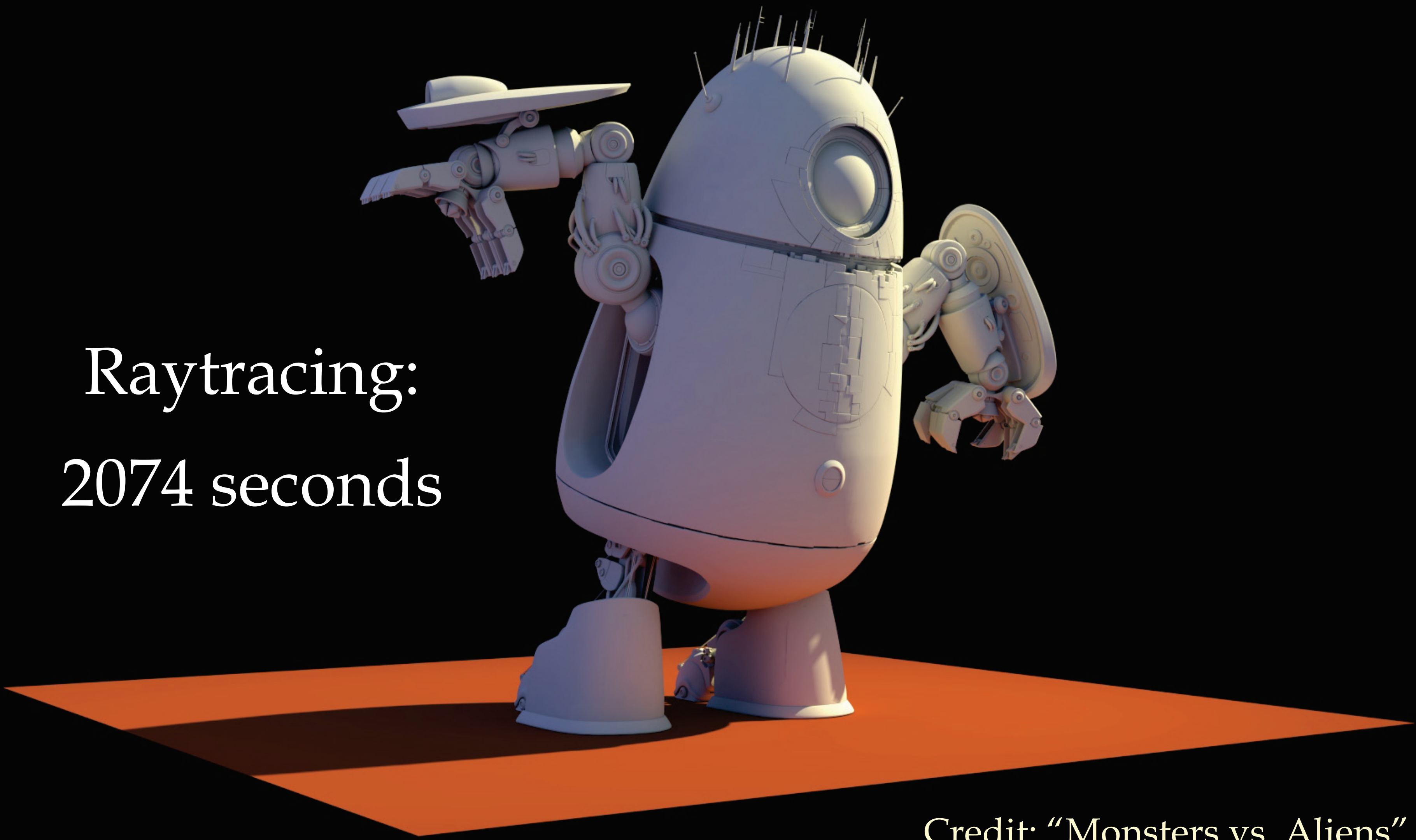
PBGI vs. Raytracing

PBGI:

277 seconds

Credit: "Madagascar: Escape 2 Africa"

PBGI vs. Raytracing



Raytracing:

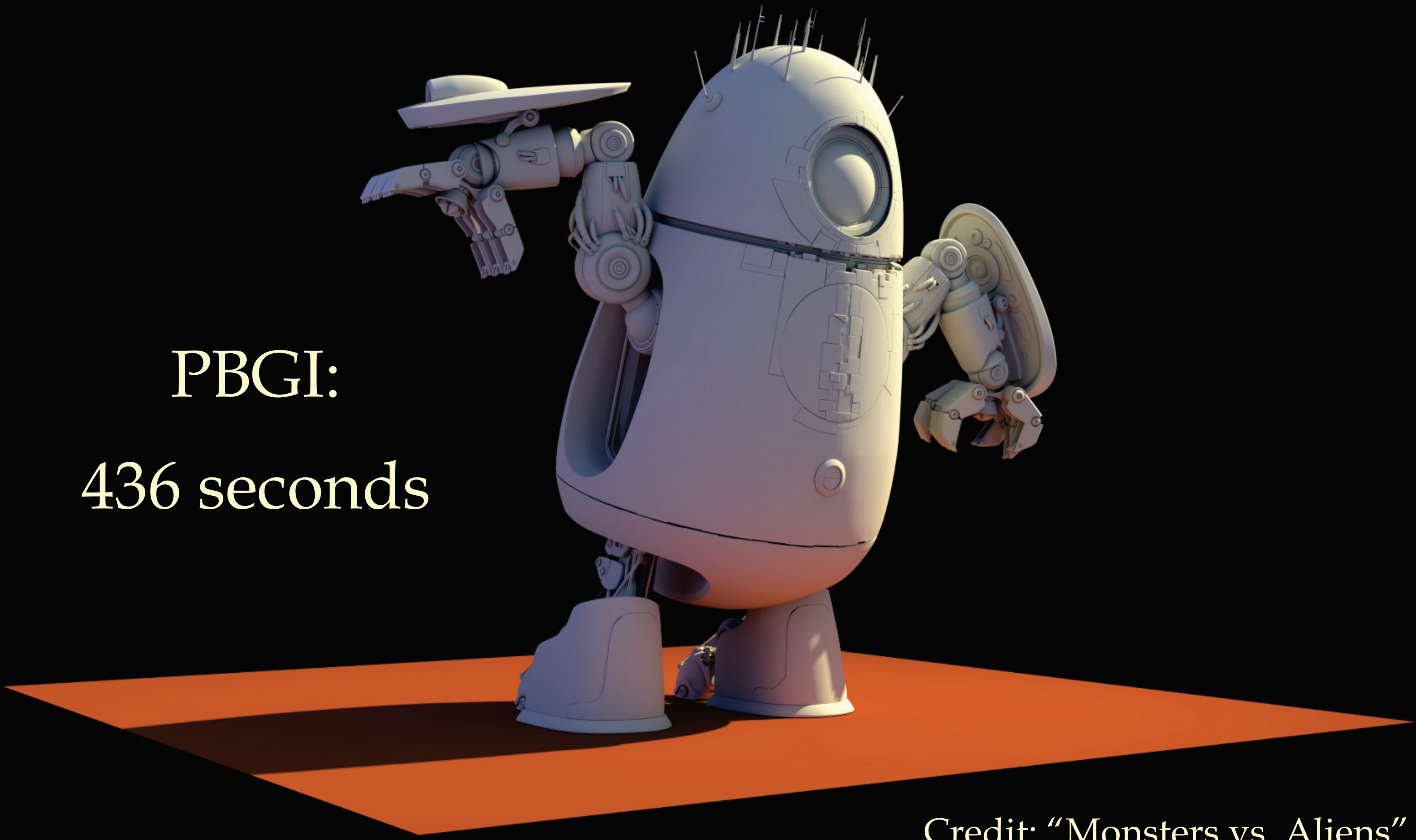
2074 seconds

Credit: "Monsters vs. Aliens"

PBGI vs. Raytracing

PBGI:

436 seconds



Credit: "Monsters vs. Aliens"

PBGI Pros

- Faster render times
  - Up to 4x faster than raytracing in some shots
- Bias vs. noise = image stability
- Unified framework for many effects
- Good at "large ray footprint" effects
  - Softer effects are cheaper
- Geometric detail doesn't increase cost
- Spatial clustering more effective than tessellation LOD
  - Aggregates primitives together
  - Aggregates shading too
- Only deal with points
  - Vs. primitives + triangle meshes + textures
- Can be made to work coherently out-of-core

PBGI Cons

- Bias vs. noise = intensity shift
- Not good at "small ray footprint" effects
  - Sharp shadows, reflections, refractions
- Lazy building not straightforward
  - Rely on point pre-processing
  - Use upper-bound on maximum needed LOD
  - Not optimal everywhere (back-facing, occlusion-culled geometry)
- Clustering limitations
  - Order 3 SH ringing
  - Circular banding artifacts (Cell / LOD jump)
    - Objects or HDR env maps with hot spots
- So many points!
  - Requires out-of-core implementation

Conclusion

- GI hard yet valuable
- Point-Based
  - Performance
  - Image stability
  - Scene complexity
- Ray-Tracing
  - Accuracy
  - Flatter geometry with less detail
  - Needed anyways for sharp reflections, refractions and shadows

References

[Christensen et al. 2003]: Ray Differentials and Multiresolution Geometry Caching for Distribution Ray Tracing in Complex Scenes

[Christensen 2008]: Point-Based Approximate Color Bleeding

[Křivánek et al. 2008]: Practical Global Illumination With Irradiance Caching

[Mirtich 1998]: V-Clip: fast and robust polyhedral collision detection

[Sloan et al. 2005]: Local, deformable precomputed radiance transfer

[Tabellion and Lamorlette 2004]: An approximate global illumination system for computer generated films

[Ward and Heckbert 1992]: Irradiance gradients

Credits

- R&D:
  - Andrew Kunz
  - Bryan Cline
  - Deepak Tolani
  - Janne Kontkanen
  - Jin Liou
  - Robin Green
  - Ryan Overbeck
- Lighting:
  - Archie Donato
  - Benjamin Venancie
  - Dan Levy
  - Igor Lodeiro
  - Jerome Platteaux
  - Jessi Stumpfel
  - Jimmy Maidens
  - Jung Jin Song
  - Liang-Yuan Wang
  - Melva Young
  - Michel Kinfoussia
  - Onny Carr
  - Udai Haraguchi
  - William Arias
- CG-Supervisors:
  - Bert Poole
  - Betsy Nofsinger
  - Mark Edwards
  - Mike McNeill
  - Pablo Valle
  - Ronman Ng