DATA STRUCTURES FOR IRRADIANCE ON THE SPHERE

ARTEFACTS AND REAL-TIME UPDATES

Dr Chris Doran

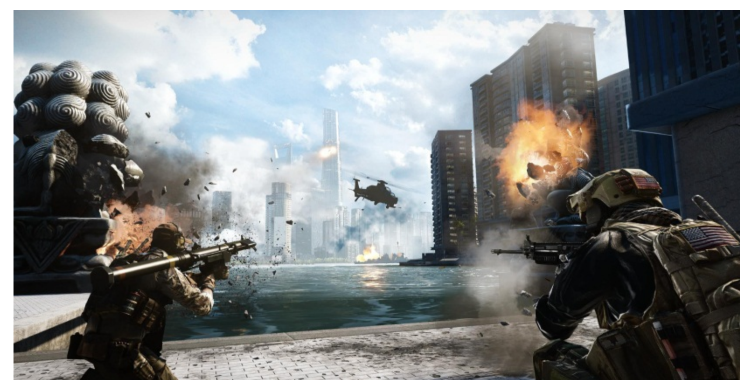
Geomerics / ARM

Outline

- The games industry
- Introduction to game graphics
- Volume re-lighting
- Surface irradiance
- Summary

Games today

- Grand Theft Auto 5 (2013)
- 15.5 GB of data on 2 DVDs
- Team size > 1,000 people (!)
- Majority of these are artists
- Vast majority of the data is content (not code)
- GTA 5 made $1bn in 48 hours (PS3 and Xbox 360 only)



Our target platforms

chrome

android

How do games get made?

- For a 'AAA' game there are three main ingredients:

1. A large capital investment
2. A huge number of artists
3. A game engine

- This is a huge investment in technology

The engine

- Content pre-processor
- Content management
- Level editor
- Rendering
- Animation
- Physics
- Lighting baker
- AI
- Scripting
- ...

} All integrated together

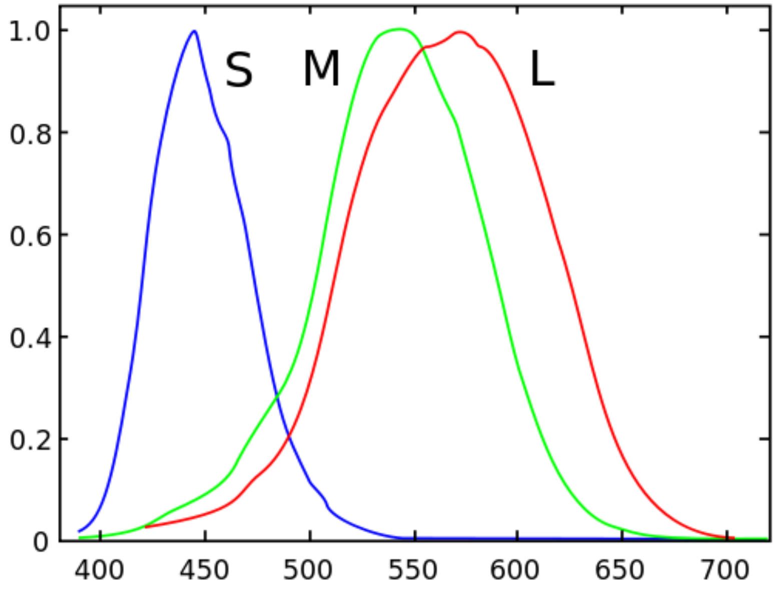
Video

Outline

- The games industry
- Introduction to game graphics
- Volume re-lighting
- Surface irradiance
- Summary

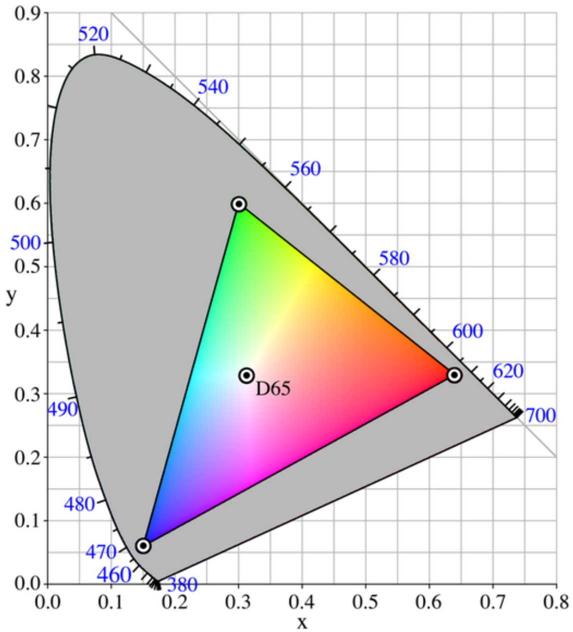
Simplification 1 – RGB Colour

- Restrict to three discrete colour channels
  - Don’t work with entire spectrum – too hard!
- Motivated by physiology of human eye
  - Three types of cone cells: L, M, S [*]

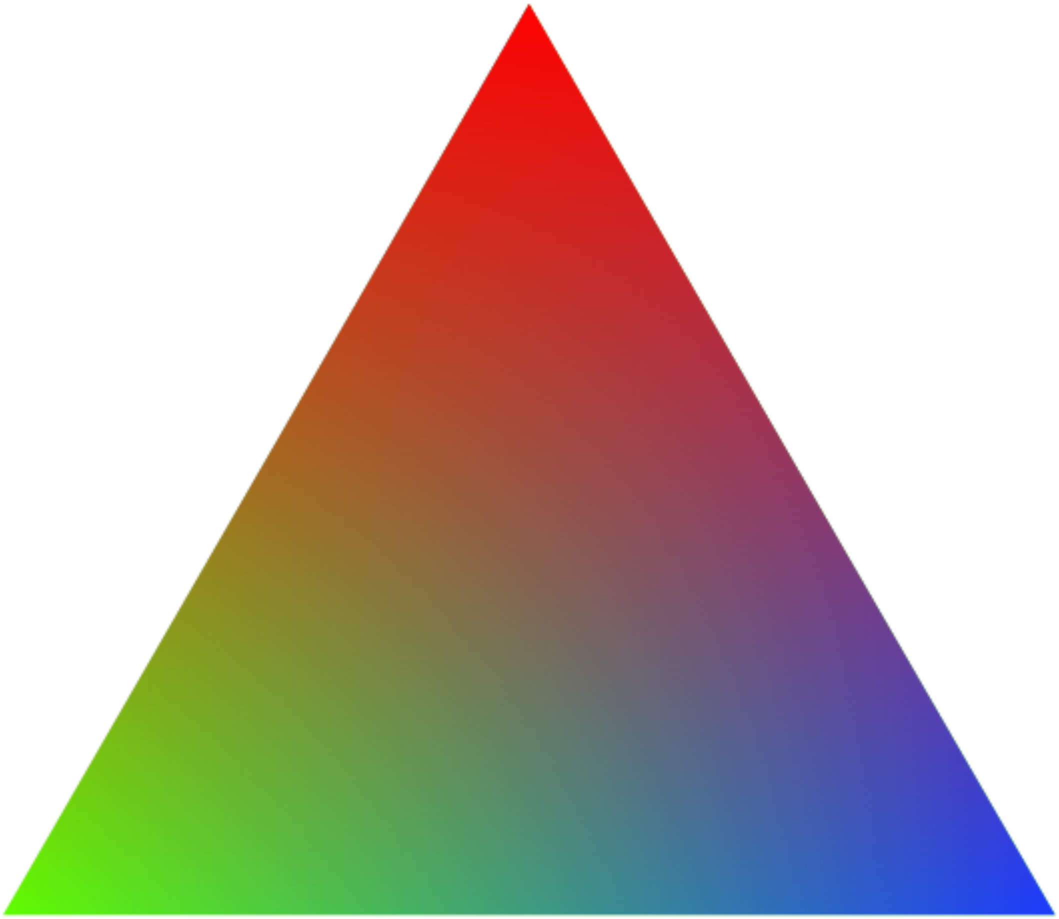


Drawbacks

- Hard to model some physical effects directly
  - Rainbows, polarisation, fluorescence, redshift
- Cannot reproduce all colours
  - Typically displays have this problem too

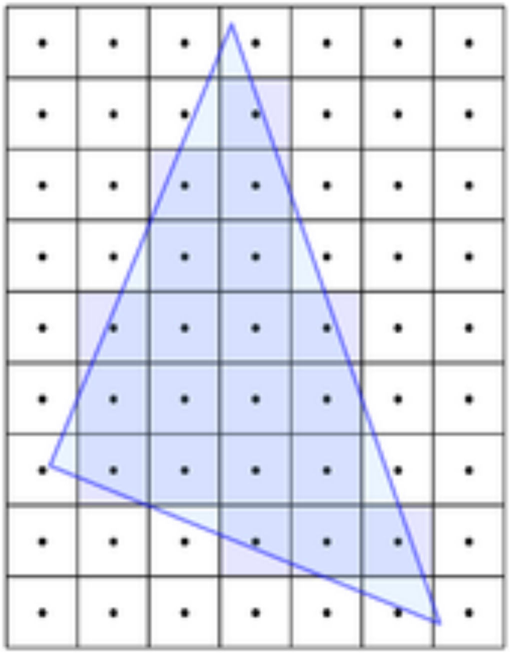


Simplification 2 – Triangles



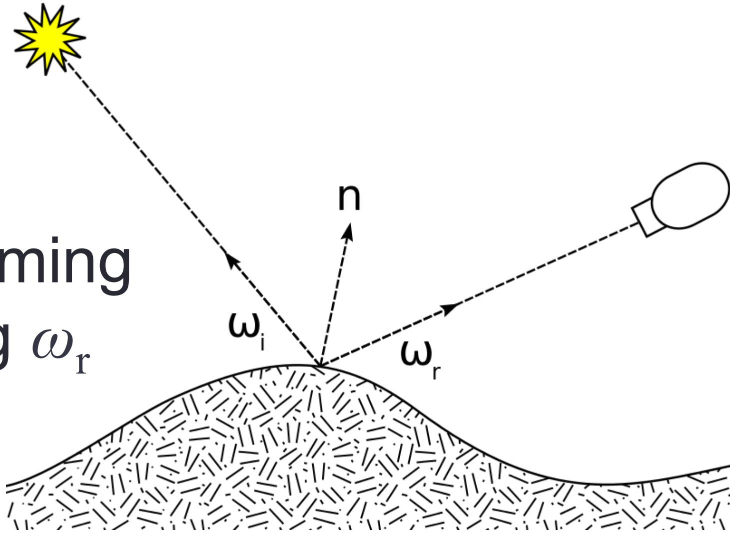
Simplification 3 – Rasterisation

- Single ray per pixel (more with MSAA)
- Give up depth-of-field blur as emergent property
- Add hardware for z-buffer, triangles ☺



Simplification 4 – BRDF

- Assume surfaces infinitesimally thin
- Disregard transmission
- Properties capture by a **Bidirectional Reflectance Distribution Function** for each surface
- Incoming light direction ω_i
- Outgoing light direction ω_r
- BRDF $f(\omega_i, \omega_r)$ is the ratio of the incoming light along ω_i which is reflected along ω_r



The Rendering Equation

$$L(\omega_r) = E(\omega_r) + \int_\Omega f(\omega_i, \omega_r) L_i(\omega_i)(\omega_i \cdot n) d\Omega_i$$

emitted light

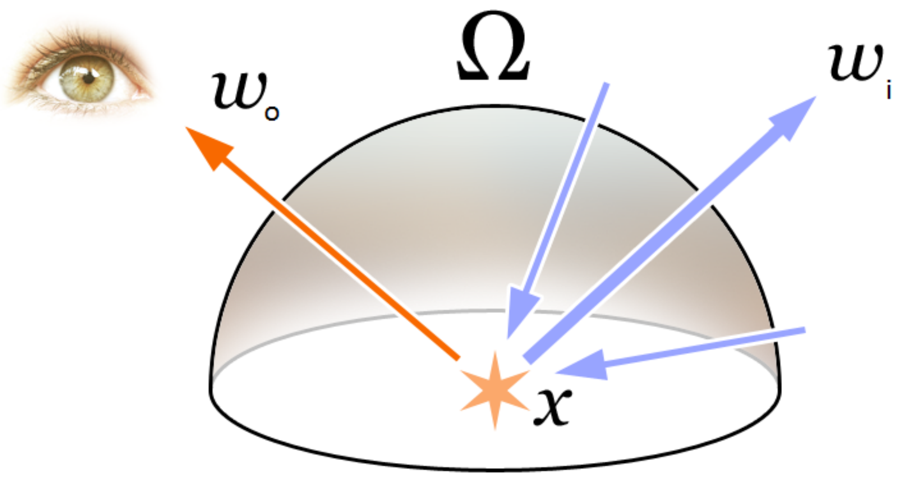
BRDF

incoming light

outgoing light

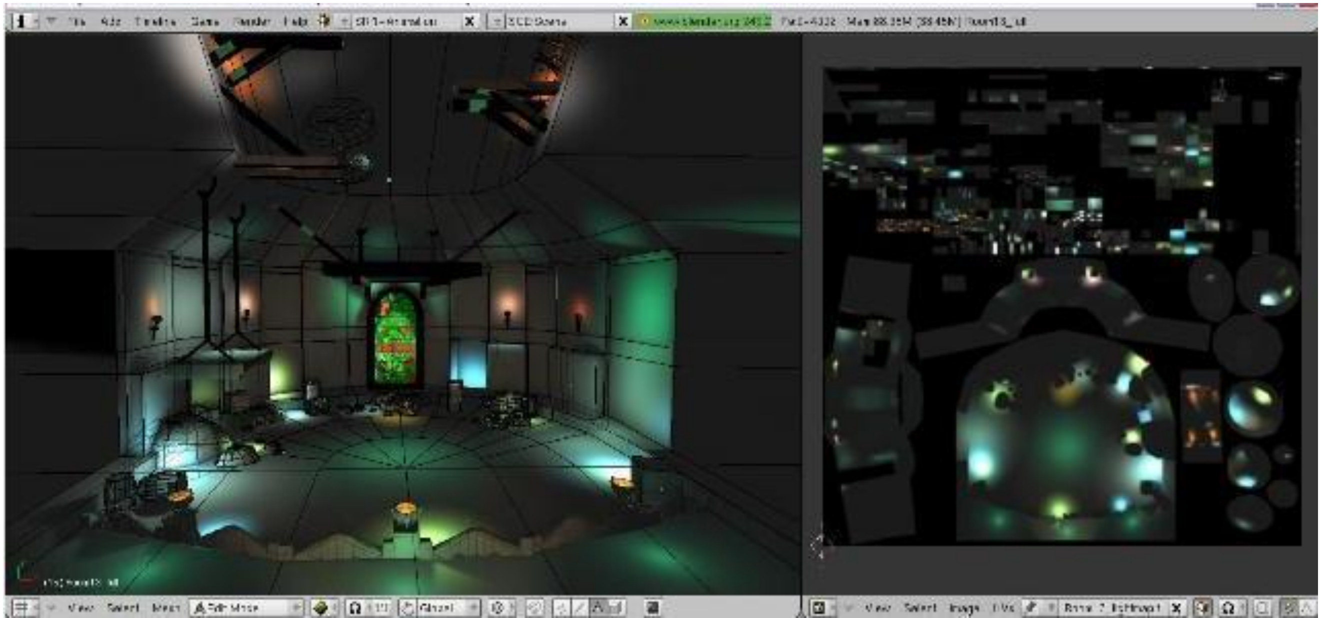
integral over hemisphere

incident angle term



Simplification 5 – Light Maps

- Simplest approach: use offline renderer
- Save result into a texture ("light map")
- Index results with UV coordinates per vertex
- Look up results in shader – very fast
- Usually static – Enlighten computes these dynamically



Simplification 6 – Split BRDF

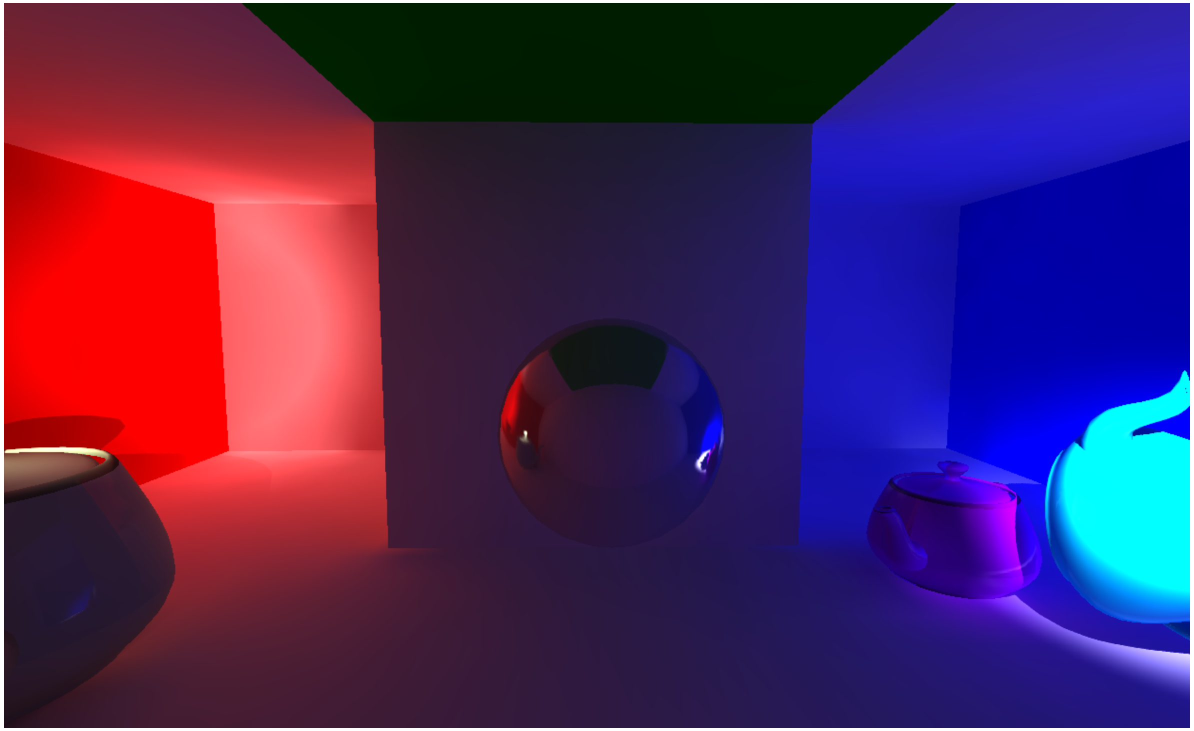
- BRDF = diffuse + specular
- Special case the specular term
- Bounce round the diffuse term - Radiosity
- This is a Lambertian diffuse surface
- Johann Heinrich Lambert, who was first to:
  - Prove pi is irrational
  - Use the term “albedo” in photometry
  - Prove formula for area of hyperbolic triangle

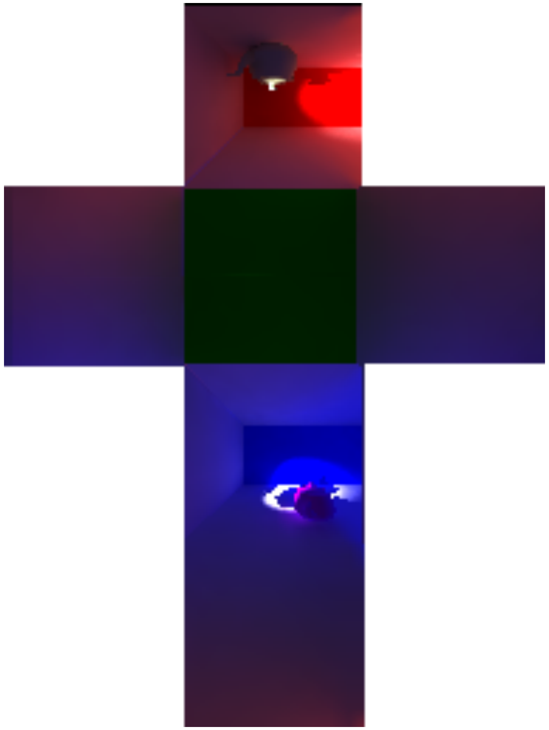
Albedo

$$L(\omega_r) = E(\omega_r) + \int_\Omega \frac{\alpha}{\pi} L_i(\omega_i)(\omega_i \cdot n)\, d\Omega_i$$

Simplification 7 – cube map specular

- Store a view of the scene in a cube map
- Efficient directional look-up
- Use mip-maps to blur based on roughness



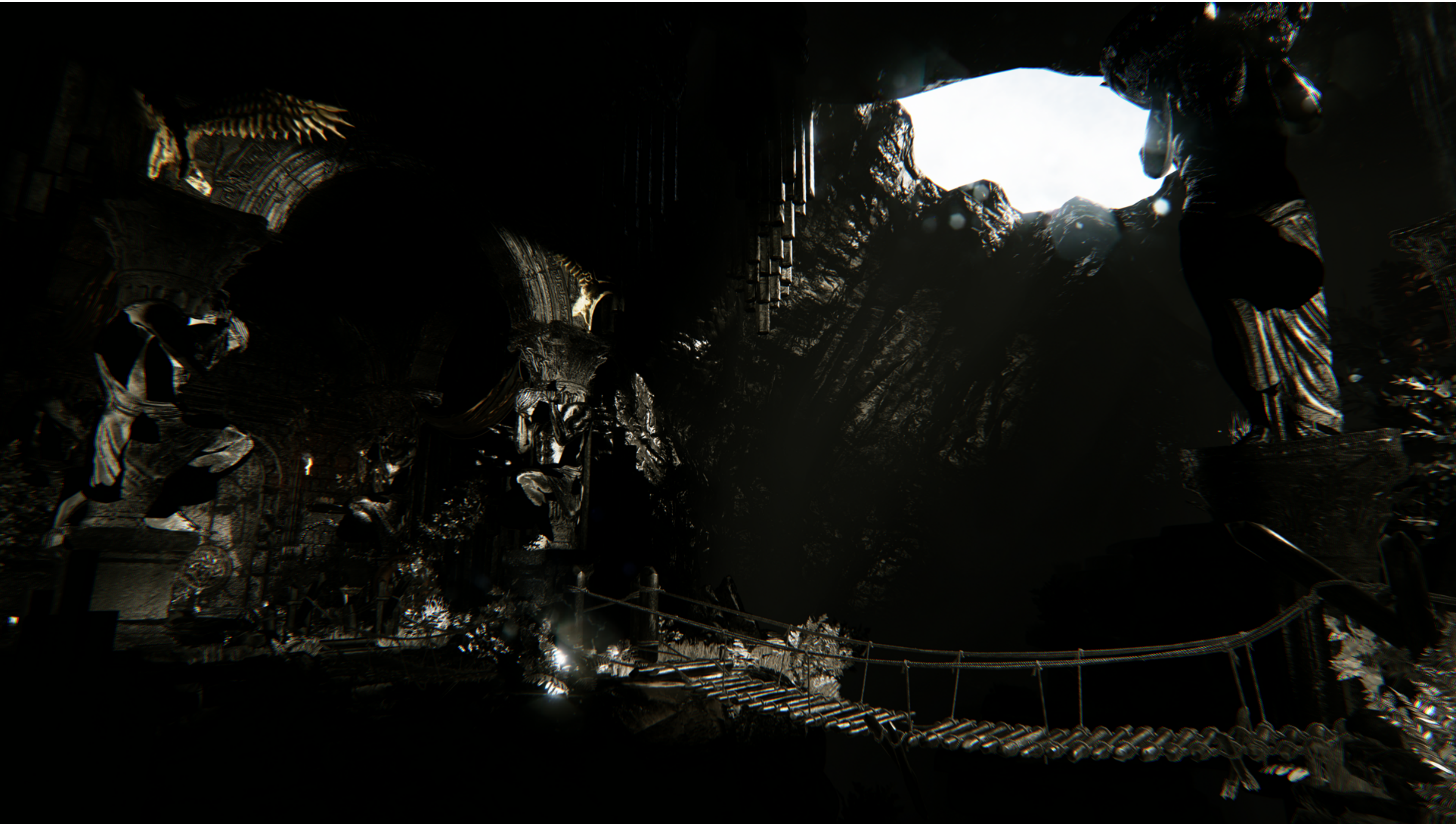


Global illumination – example

Global illumination – direct only

Global illumination radiosity pass

Global illumination – specular pass



Global illumination – final image

Outline

- The games industry
- Introduction to game graphics
- Volume re-lighting
- Surface irradiance
- Summary

Dynamic objects

- Small dynamic objects are lit with ‘light probes’
  - Capture the light field volumetrically
  - Relight geometry with arbitrary position/orientation
  - Use ‘spherical harmonics’

Some notation

- In graphics we rarely go beyond $l=2$
- Don't want to mess around with complex numbers
- Don't bother with factors of $\sqrt{\pi}$
- Normalise once
- Think of first three terms as scalar, vector, symmetric tensor

$$f(\boldsymbol{\omega}) = f_0 + \boldsymbol{f} \cdot \boldsymbol{\omega} + \boldsymbol{\omega}^t F \boldsymbol{\omega}$$

- In some cases makes sense to combine all 9 coefficients into a 3x3 matrix
  - Efficient for rotation

Useful result

- Ramamoorthi + Hanrahan 2001 (JOSA/A)
- At a probe, suppose the incoming light is given by

$$l(\boldsymbol{\omega}) = l_0 + \boldsymbol{l}\cdot\boldsymbol{\omega} + \boldsymbol{\omega}^t L\boldsymbol{\omega} + \cdots$$

- The irradiance in a direction $\boldsymbol{n}$ is given by

$$I(\boldsymbol{n}) = \int_H \frac{d\omega}{\pi}\boldsymbol{\omega}\cdot\boldsymbol{n}\, L(\boldsymbol{\omega})$$

- Which gives

$$I(\boldsymbol{n}) = l_0 + \frac{2}{3}\boldsymbol{l}\cdot\boldsymbol{\omega} + \frac{1}{4}\boldsymbol{n}^t L\boldsymbol{n} + \cdots$$

Useful result

$$l(\boldsymbol{\omega}) = l_0 + \boldsymbol{l}\cdot\boldsymbol{\omega} + \boldsymbol{\omega}^t L \boldsymbol{\omega} + \cdots$$

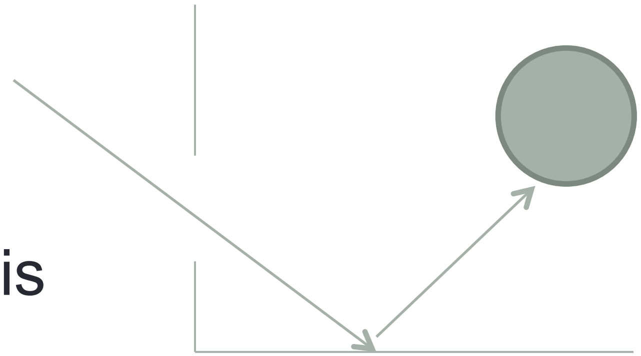
$$I(\boldsymbol{n}) = l_0 + \frac{2}{3}\boldsymbol{l}\cdot\boldsymbol{\omega} + \frac{1}{4}\boldsymbol{n}^t L \boldsymbol{n} + \cdots$$

- All higher-order odd terms vanish from irradiance
- Coefficients of the even terms fall off as $1/l$
- Used to justify terminating at $l=2$
- On last generation consoles we just kept the scalar and vector terms!

Problems

- Low order harmonics produce some unpleasant artefacts
- Pool of light on floor from a window produces a highly directional input
- In this case the SH reconstruction can easily go negative
- In sampling terms

$$\boldsymbol{l} = \frac{3}{n} \sum_a l^a \boldsymbol{\omega}^a$$



- If one direction dominates then $\boldsymbol{l}$ is larger than scalar term

Problems

- Not clear that going to $l=2$ will resolve all problems
- May just introduce unpleasant ringing terms
- On latest games have a lot of objects lit with probes
  - Logic to determine which probes to compute is complex
  - Simpler to just update a voxel grid
  - Interpolate on the GPU
- End up computing quite a dense grid
  - 9 coefficients per colour channel = 27 in total
- Starting to look at multi-resolution techniques
  - Fill in the grid when data slowly changing
- But can we do better?

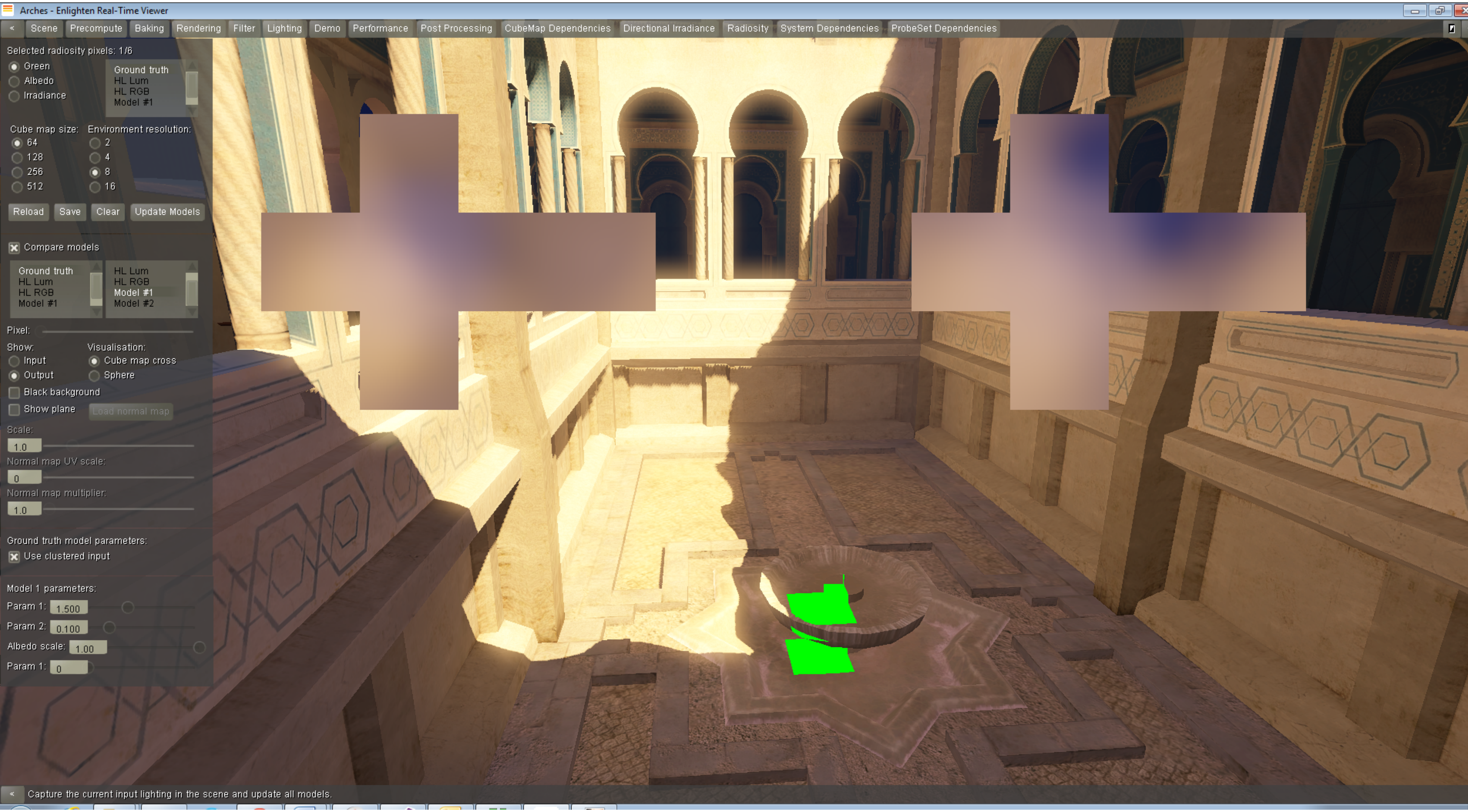
Outline

- The games industry
- Introduction to game graphics
- Volume re-lighting
- Surface irradiance
- Summary

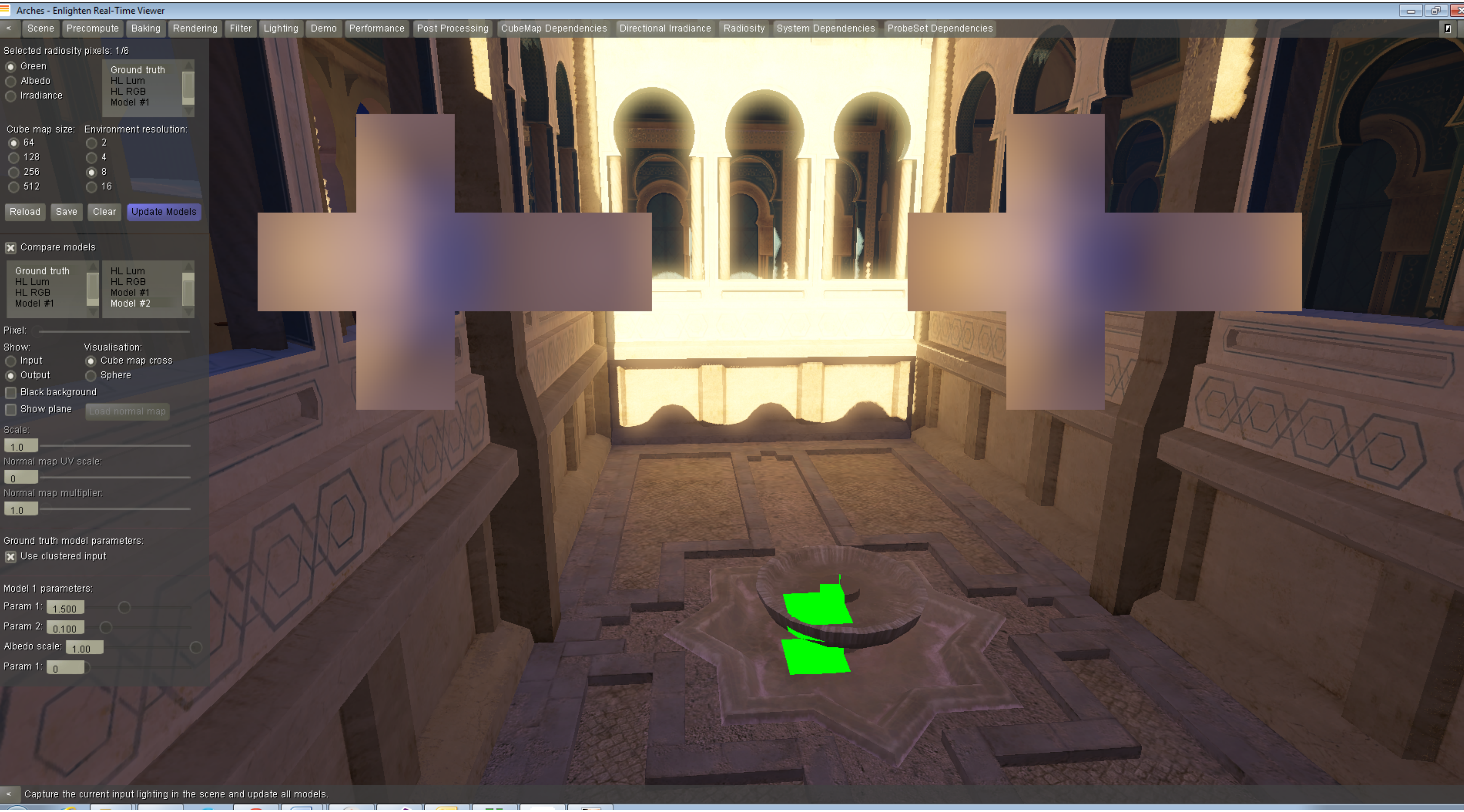
Surface Irradiance

- In Enlighten we compute the irradiance on surfaces at a low resolution
  - Fill in detail by upsampling
- Fine for smooth surfaces
- Need to consider geometric detail
  - Small artefacts on surface
  - Normal maps
- So we also compute an average direction of incoming light
- Do this under cosine distribution for simplicity
- Looking for a better model

Test scene


Arches - Enlighten Real-Time Viewer
Scene Precompute Baking Rendering Filter Lighting Demo Performance Post Processing CubeMap Dependencies Directional Irradiance Radiosity System Dependencies ProbeSet Dependencies
Selected radiosity pixels: 1/6
Green
Albedo
Irradiance
Ground truth
HL Lum
HL RGB
Model #1
Cube map size:
64
128
256
512
Environment resolution:
2
4
8
16
Reload Save Clear Update Models
Compare models
Ground truth
HL Lum
HL RGB
Model #1
HL Lum
HL RGB
Model #1
Model #2
Pixel:
Show:
Input
Output
Visualisation:
Cube map cross
Sphere
Black background
Show plane
Load normal map
Scale:
1.0
Normal map UV scale:
0
Normal map multiplier:
1.0
Ground truth model parameters:
Use clustered input
Model 1 parameters:
Param 1: 1.500
Param 2: 0.100
Albedo scale: 1.00
Param 1: 0
Capture the current input lighting in the scene and update all models.

Test scene


Arches - Enlighten Real-Time Viewer
Scene Precompute Baking Rendering Filter Lighting Demo Performance Post Processing CubeMap Dependencies Directional Irradiance Radiosity System Dependencies ProbeSet Dependencies
Selected radiosity pixels: 1/6
Green
Albedo
Irradiance
Ground truth
HL Lum
HL RGB
Model #1
Cube map size:
Environment resolution:
64
128
256
512
2
4
8
16
Reload
Save
Clear
Update Models
Compare models
Ground truth
HL Lum
HL RGB
Model #1
HL Lum
HL RGB
Model #1
Model #2
Pixel:
Show:
Input
Output
Visualisation:
Cube map cross
Sphere
Black background
Show plane
Load normal map
Scale:
1.0
Normal map UV scale:
0
Normal map multiplier:
1.0
Ground truth model parameters:
Use clustered input
Model 1 parameters:
Param 1:
1.500
Param 2:
0.100
Albedo scale:
1.00
Param 1:
0
Capture the current input lighting in the scene and update all models.

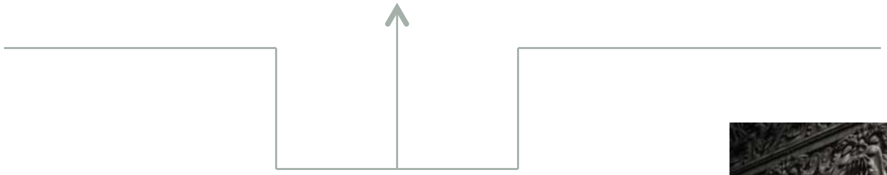
Examples

Problems

- Trying to deal with four types of geometry
- Curved surfaces

- Extrusions

- Intrusions



- Normal maps

Problems

- Can reconstruct a plausible scalar + vector term:

$$I(\boldsymbol{\omega}) = L\frac{1}{2}(1+\cos\theta) + (\boldsymbol{d} \wedge \boldsymbol{nn})\cdot\boldsymbol{\omega}$$

- But only so much you can do with a scalar and a vector
  - Raise a half-Lambert term to a power?
- Have to model the back reflections
- Have surface albedo information here
- Can try half Lambert in back direction

$$I'(\boldsymbol{\omega}) = L\alpha\frac{1}{2}(1-\cos\theta)$$

Problems

- Given the measured data we can reconstruct a ‘best fit’ SH series (up to 2 usually)
- Often collect the directional information at lower resolution
- Want to collect directional data for separate light channels
- Directional data is very susceptible to quantization error
  - See this over a smooth curved surface
- Reconstruction will be done on the GPU in a shader
  - Has to be very simple and hardware friendly
- Can we use the fact the data is hemispherical more intelligently?

Outline

- The games industry
- Introduction to game graphics
- Volume re-lighting
- Surface irradiance
- Summary

Summary: wish list

- Want a spherical basis with
  - Small number of parameters
  - Smooth
  - Local support
  - Positive (functions are all positive)
  - Good rotation properties
  - Fast projection and reconstruction
  - Good interpolation and multi-resolution properties
- Want something similar for the hemisphere
- Also looking for compact representations of BRDF
  - Beyond Lambertian reflection
  - Blinn, Catmull Clark …

Questions

- chris.doran@geomerics.com
- chris.doran@arm.com