CS348b Review Session

Phaedon Sinis • Alexis Haraux
Friday, 18th April 2012
Today’s topics

- Radiometry
- Assignment 2 (Ray-heightfield intersection)
- VDB (visual debugging)
Radiometry

- Radiant energy
- Radiant power
- Irradiance
- Radiosity
- Radiance
- Radiant intensity

Let’s shed some light!
Radiant Power (flux)

Each photon carries a fixed amount of energy (measured in Joules)

Radiant power only tells us how many photons, each second, are either:
- coming out of a light
- or hitting a surface

- we don’t know size
- we don’t know direction

300 Watts = 300 Joules / sec

• A light source emits radiant power
• Radiant power is incident on a surface
Irradiance

Irradiance (Watts / m^2) is measured at a single point.
It’s the density of radiant power, *incident on a surface*, from ALL directions.

Let’s take another look at the 300-Watt ARRI light:

**Illuminance:**
630 lux per light, at a distance of 3.8 meters
(source: ARRI’s photometric calculator)

Note: We’d need to know the *luminous efficacy of radiation* to convert illuminance to irradiance.

- Irradiance (Watts / m^2) is measured at a single point.
- It’s the density of radiant power, *incident on a surface*, from ALL directions.
Radiosity
(Radiant exitance)

• here, it’s the radiometric quantity, not the algorithm (so not only for diffuse surfaces!)

• radiant exitance (Watts / m\(^2\)) is measured at a single point (function of position)

• density of radiant power reflecting off of a surface, towards \textit{all} directions
Radiance

- Watts / (steradian • m²)
- sensor response (cameras, retinas) is proportional to incident radiance
- radiance is invariant along straight paths
- Q: what about prisms?
- Q: what about participating media?
A few basic relations

Radiance: \[ L(x \rightarrow \Theta) \text{ and } L(x \leftarrow \Theta) \]

Irradiance: \[ E(x) = \int_{\Omega} L(x \leftarrow \Theta) \cos \theta d\omega_\Theta \]

Radiant power: \[ \Phi = \int_{A} E(x) dA_x \]
Example

Uniform diffuser

\[ 3.91 \times 10^{26} \text{ Watts} \]

\[ 1000 \text{ W} / (\text{sr} \cdot \text{m}^2) \]

Source: Dutré et al, Advanced Global Illumination, Chapter 2
Example (cont.)

\[ E = \int L(x \leftarrow \Theta) \cos \theta d\omega \]

\[ = 1000 \int \int \cos \theta \sin \theta d\theta d\phi \]

\[ = 1000 \int_0^{2\pi} d\phi \int_0^{\pi/6} \cos \theta \sin \theta d\theta \]

\[ = (1000)(2\pi) \left[ -\frac{\cos^2 \theta}{2} \right]_0^{\pi/6} \]

\[ = 250\pi W/m^2 \]

Source: Dutré et al, Advanced Global Illumination, Chapter 2
Assignment 2

Accelerating Heightfields
This is not a triangle mesh

Photo credit: http://www.flickr.com/photos/80651083@N00/6910354874/
Heightfields are special

• “Your assignment is to develop a fast intersection routine for heightfields.

• You can use techniques similar to those used for grid- or tree-based acceleration structures, or you can come up with your own design.

• Your technique should in some way exploit the regularity found in heightfields to improve some performance characteristic (e.g. faster intersections, lower memory, faster construction, etc.).”
Tasks

• Render scenes as-is
  • by now you have a good workflow for working in PBRT
• Write acceleration structure
• u,v coordinates for texture-mapping
• Phong interpolation for smooth terrain
• performance writeup!
VDB

- “the printf of visual debugging”
- by Zach DeVito
- https://github.com/zdevito/vdb
VDB setup

• grab the source & build it!
• copy the header vdb.h into pbrt/src/
• In PBRT, edit renderers/samplerrenderer.cpp
• Go to the SamplerRenderer::Li() method and edit the “if” clause as follows:

```cpp
if (scene->Intersect(ray, isect)) {
    Li = surfaceIntegrator->Li(scene, this, ray, *isect, sample, rng, arena);

    float vdbrgb[3];
    Li.ToRGB(&vdbrgb[0]);

    vdb_color(vdbrgb[0], vdbrgb[1], vdbrgb[2]);
    vdb_point(isect->dg.p.x, isect->dg.p.y, isect->dg.p.z);
}
```
VDB, the magic

- `#include "../vdb.h"
- rebuild PBRT
- launch the vdb server
- render your favorite scene and watch what happens!
  - use the `--ncores 1` flag (VDB is not threadsafe)