Real Numbers, Real Images

Greg Ward
Anyhere Software

Course Outline

I. Introduction
II. Measurement
III. Lighting Simulation
IV. Image Representation
V. Image Display
VI. Image-based Techniques
VII. Conclusions

I. Introduction

- Graphics rendering software & hardware
  - Past
  - Present
  - Future
- Will graphics hardware take over?
- Why "real" numbers are better for rendering and imaging

Rendering Software Past

- Hidden-surface removal in a polygonal environment
  - Optional textures, bump maps, env. maps
- Local illumination
  - Gouraud and Phong shading
  - Shadow maps – some of them analytical!
- Ray-tracing for global illumination
  - Quadric surfaces and specular reflections

Graphics Hardware Past

- Fixed, 8-bit range for lights & materials
- Integer color operations
  - Phong and Gouraud shading hardware
  - Sometimes linear, sometimes pre-gamma
- Limited texture & fragment operations
- Output is 24-bit RGB sent to DAC (digital to analog converter) for analog display

Graphics Hardware Present

- Floating-point (FP) sources and materials
- Mix of integer and FP operations
  - Operations in linear or near-linear color space
- Extensive use of textures and MIP-maps
  - Programmable pixel shaders w/ some FP
- Output converted to 24-bit sRGB
  - Blending usually done in integer space
  - Display via digital video interface (DVI)

Rendering Software Present

- Global illumination (GI) in complex scenes
  - Environments with > 105 primitives common
  - Programmable shaders are the norm
- Micropolygon architectures prevalent
- Radiosity sometimes used for GI
- Ray-tracing (RT) used more and more

Rendering Software Future

- Hyper-complex environments (> 107 primitives)
  - Procedural scene descriptions
  - "Localized" version of global illumination
- Micropolygon architectures hang on
- Radiosity as we know it disappears
- Ray-tracing and Monte Carlo take over
  - Graceful handling of large data sets
  - Ordered rendering improves memory access

Graphics Hardware Future

- Floating-point operations throughout
  - All operations in linear color space
- High-level GPU programming standard
  - Compilers for multipass rendering
- Output converted to 64-bit RGBA
  - Cards output "layers" rather than images
  - Post-card blending on a novel display bus
  - New, high dynamic-range display devices

Will Hardware Take Over?

- No, rendering software will always exist
  - Needed for testing new ideas
  - Ultimately more flexible and controllable
  - Hardware does not address specialty markets
- But, graphics hardware will dominate
  - Programmable GPUs add great flexibility
  - Speed will always be critical to graphics
  - Read-back performance must be improved!

Why Real Numbers Are Better for Rendering & Imaging

- The natural range of light is huge ~1012
  - Humans adjust comfortably over 8 orders
  - Humans see simultaneously over 4 orders
- Color operations, including blending, must reproduce 10000:1 contrasts with final accuracy of 1% or better to fool us
- Human color sensitivity covers about twice the area of an sRGB display gamut

Dynamic Range
CCIR-709 (sRGB) Color Space
HDR Imaging Approach

- Render/Capture floating-point color space
- Store entire perceivable gamut (at least)
- Post-process in extended color space
- Apply tone-mapping for specific display
- HDR used extensively at ILM, Digital Domain, ESC, Rhythm & Hues

HDR Imaging Is Not New

- B&W negative film holds at least 4 orders of magnitude
- Much of the talent of photographers like Ansel Adams was darkroom technique
- “Dodge” and “burn” used to bring out the dynamic range of the scene on paper
- The digital darkroom provides new challenges and opportunities

HDR Tone-mapping
Post-production Possibilities

II. Measurement

- How do we obtain surface reflectances?
- How do we obtain surface textures (and milli geometry)?
- How do we obtain light source distributions?
- What is the best color space to work in?

Macbeth ColorChecker™ Chart

- Digital photo with ColorChecker™ under uniform illumination
- Compare points on image and interpolate
- Best to work with HDR image
- Accurate to ~ 10 ΔE

Radiance macbethcal Program

- Computes grayscale function and 3x3 color transform
- Maintain the same measurement conditions
- Calibrated pattern or uniform color capture
- Accurate to ~ 6 ΔE

Spectrophotometer

- Commercial spectrophotometers run about $5K US
- Measure reflectance spectrum for simulation under any light source
- Accurate to ~ 2 ΔE

BRDF Capture 1

BRDF Capture 2

Combined Capture Method 1

- Pietà Project
  www.research.ibm.com/pieta
  [Rushmeier et al. EGWR '98]
- Multi-baseline stereo camera with 5 lights
- Captured geometry and reflectance
- Sub-millimeter accuracy

Combined Capture Method 2

- CURET database
  - www1.cs.columbia.edu/CAVE/curet/
  - [Dana et al. TOG '99]
- Capture BTF (bidirectional texture function)
- Interpolate BTF during rendering

Combined Capture Method 3

- Lumitexel capture
  - [Lensch et al. EGWR '01]
- Capture 3-D position + normal + color as function of source position
- Fit data locally to BRDF model
- Render from BRDF

Light Source Distributions

- Often ignored, light source distributions are the first order of lighting simulation
- Data is comparatively easy to obtain
  - Luminaire manufacturers provide data files
    - See www.ledalite.com/resources/software
  - American and European standard file formats
  - Hardcopy photometric reports also available

Luminaire Data

- Photometric reports contain candela information per output direction
- All photometric measurements assume a far–field condition
- Interpolate directions and assume uniform over area

Candela Conversion

- A candela equals one lumen/steradian
- A lumen is approximately equal to 0.0056 watts of equal-energy white light
- To render in radiance units of watts/sr-m2
  - Multiply candelas by 0.0056/d**A** where d**A** is projected area in each output direction in m2

What Color Space to Use?

1) How Does *RGB* Rendering Work and When Does It Not?
2) Can *RGB* Accuracy Be Improved?
3) Useful Observations
4) Spectral Prefiltering
5) The von Kries White Point Transform
6) Experimental comparison of 3 spaces

A Brief Comparison of Color Rendering Techniques

- Spectral Rendering
  - ✓ N spectrally pure samples
- Component Rendering
  - ✓ M vector basis functions
- *RGB* (Tristimulus) Rendering
  - ✓ Tristimulus value calculations

Spectral Rendering

1. Divide visible spectrum into N wavelength samples
2. Process spectral samples separately throughout rendering calculation
3. Compute final display color using CIE color matching functions and standard transformations

Component Rendering

[Peercy, Siggraph '93]

1. Divide visible spectrum into M vector bases using component analysis
2. Process colors using MxM matrix multiplication at each interaction
3. Compute final display color with 3xM matrix transform

RGB (Tristimulus) Rendering

1. Precompute tristimulus values
2. Process 3 samples separately throughout rendering calculation
3. Compute final display color with 3x3 matrix transform (if necessary)

Rendering Cost Comparison

Strengths and Weaknesses

Spectral Aliasing

The Data Mixing Problem

- Typical situation:
  - Illuminants known to 5 nm resolution
  - Some reflectances known to 10 nm
  - Other reflectances given as tristimulus
- Two alternatives:
  A. Reduce all spectra to lowest resolution
  B. Interpolate / synthesize spectra [Smits '99]

Status Quo Rendering

- White Light Sources
  - E.g., (R,G,B)=(1,1,1)
- *RGB* material colors obtained by dubious means
  - E.g., "That looks pretty good."
    - ✓ This actually works for fictional scenes!
- Color correction with ICC profile if at all

When Does RGB Rendering Normally Fail?

- When you start with measured colors
- When you want to simulate color appearance under another illuminant
- When your illuminant and surface spectra have sharp peaks and valleys

Can *RGB* Accuracy Be Improved?

- Identify and minimize sources of error
  - Source-surface interactions
  - Choice of rendering primaries
- Overcome ignorance and inertia
  - Many people render in *RGB* without really understanding what it means
  - White-balance problem scares casual users away from colored illuminants

A Few Useful Observations

a) Direct illumination is the first order in any rendering calculation
b) Most scenes contain a single, dominant illuminant spectrum
c) Scenes with mixed illuminants will have a color cast regardless

Picture Perfect *RGB* Rendering

1. Identify dominant illuminant spectrum
   a) Prefilter material spectra to obtain tristimulus colors for rendering
   b) Adjust source colors appropriately
2. Perform tristimulus (*RGB*) rendering
3. Apply white balance transform and convert pixels to display color space

Spectral Prefiltering

Prefiltering vs. Full Spectral Rendering

+ Prefiltering performed once per material vs. every rendering interaction
+ Spectral aliasing and data mixing problems disappear with prefiltering
- However, mixed illuminants and interreflections not computed exactly

Quick Comparison

The von Kries Transform for Chromatic Adaptation

Chromatic Adaptation Matrix

- The matrix **M**C transforms *XYZ* into an "adaptation color space"
- Finding the optimal CAM is an under-constrained problem -- many candidates have been suggested
- "Sharper" color spaces tend to perform better for white balance transforms
  - See [Finlayson & Susstrunk, CIC '00]

Three Tristimulus Spaces for Color Rendering

- CIE *XYZ*
  - Covers visible gamut with positive values
  - Well-tested standard for color-matching
- *sRGB*
  - Common standard for image encoding
  - Matches typical CRT display primaries
- *Sharp RGB*
  - Developed for chromatic adaptation

XYZ Rendering Process

1. Apply prefiltering equation to get absolute *XYZ* colors for each material
   a) Divide materials by illuminant:

   b) Use absolute *XYZ* colors for sources
2. Render using tristimulus method
3. Finish w/ CAM and display conversion

sRGB Rendering Process

1. Perform prefiltering and von Kries transform on material colors
   a) Model dominant light sources as neutral
   b) For spectrally distinct light sources use:

2. Render using tristimulus method
3. Resultant image is *sRGB*

Sharp RGB Rendering Process

1. Prefilter material colors and apply von Kries transform to *Sharp RGB* space:
2. Render using tristimulus method
3. Finish up CAM and convert to display

Our Experimental Test Scene

Experimental Results

- Three lighting conditions
  - Single 2856°K tungsten light source
  - Single cool white fluorescent light source
  - Both light sources (tungsten & fluorescent)
- Three rendering methods
  - Naïve *RGB* (assumes equal-energy white)
  - Picture Perfect *RGB*
  - Full spectral rendering (380 to 720 nm / 69 samp.)
- Three color spaces (*XYZ*, *sRGB*, *Sharp RGB*)

Example Comparison (*sRGB*)

ΔE* Error Percentiles for All Experiments

Results Summary

- Prefiltering has ~1/6 the error of naïve rendering for single dominant illuminant
- Prefiltering errors similar to naïve in scenes with strongly mixed illuminants
- CIE *XYZ* color space has 3 times the rendering errors of *sRGB* on average
- *Sharp RGB* rendering space reduces errors to 1/3 that of *sRGB* on average

III. Lighting Simulation

- Approximating local illumination
- Approximating global illumination
- Dealing with motion
- Exploiting human perception to accelerate rendering

Local Illumination

- Local illumination is the most important part of rendering, and *everyone* gets it wrong (including me)
- Real light-surface interactions are incredibly complex, and humans have evolved to perceive many subtleties
- The better your local illumination models, the more realistic your renderings

LI Advice: Use Physical Range

- Non-metallic surfaces rarely have specular reflectances greater than 7%
  - Determined by the index of refraction, $n < 1.7$
- Physically plausible BRDF models obey energy conservation and reciprocity
  - Phong model often reflects > 100% of incident
- *RGB* reflectances may be slightly out of [0,1] range for highly saturated colors

LI Advice: Add Fresnel Factor

- Specular reflectance goes up near grazing for all polished materials – here is a good approximation for Fresnel reflection:
- Simpler & faster than standard formula
- Improves accuracy and appearance at silhouettes

Fresnel Approximation
LI Advice: Texture Carefully

- Pay attention to exactly how your image textures affect your average and peak reflectances
  - Are they still in a physically valid range?
- Use bump maps sparingly
  - Odd artifacts arise when geometry and surface normals disagree strongly
  - Displacement maps are better

LI Advice: Use BTF Model

- Use CURET data to model view-dependent appearance under different lighting using *TensorTexture* technique
  - See "TensorTextures", M. Alex O. Vasilescu and D. Terzopoulos, Sketch and Applications SIGGRAPH 2003 San Diego, CA, July, 2003. www.cs.toronto.edu/~maov/tensortextures/tensortextures_sigg03.pdf

Global Illumination

- Global illumination will not fix problems caused by poor local illumination, but…
  - GI adds another dimension to realism, and
  - GI gets you absolute answers for lighting
- Radiosity methods compute form factors
  - Says nothing about global illumination
- Ray-tracing methods intersect rays
  - Again, this is not a useful distinction

GI Algorithm Characteristics

- Traces rays
- Subdivides surfaces into quadrilaterals
- Employs form factor matrix
- Deposits information on surfaces
  - Using grid
  - Using auxiliary data structure (e.g., octree)
- Requires multiple passes

GI Example 1: Hemicube Radiosity [Cohen et al. '86]

× Traces rays
✓ Subdivides surfaces into quadrilaterals
✓ Employs form factor matrix
✓ Deposits information on surfaces
    ✓ Using grid
    × Using auxiliary data structure (e.g., octree)
✓ Requires multiple passes

GI Example 2: Particle Tracing [Shirley et al. '95]

- ✓ Traces rays
- × Subdivides surfaces into quadrilaterals
  - ✓ But triangles, yes
- × Employs form factor matrix
- ✓ Deposits information on surfaces
  - × Using grid
  - ✓ Using auxiliary data structure (T-mesh)
- ✓ Requires multiple passes

GI Example 3: Monte Carlo Path Tracing [Kajiya '86]

- ✓ Traces rays
- × Subdivides surfaces into quadrilaterals
- × Employs form factor matrix
- × Deposits information on surfaces
- × Requires multiple passes

GI Example 4: *Radiance*

- ✓ Traces rays
- × Subdivides surfaces into quadrilaterals
- × Employs form factor matrix
- ✓ Deposits information on surfaces
  - × Using grid
  - ✓ Using auxiliary data structure (octree)
- × Requires multiple passes

The Rendering Equation *Radiance* Calculation Methods

- Direct calculation removes large incident
- Indirect calculation handles most of the rest
- Secondary light sources for problem areas
- Participating media (adjunct to equation)

Radiance Direct Calculation

- Selective Shadow Testing
  - Only test significant sources
- Adaptive Source Subdivision
  - Subdivide large or long sources
- Virtual Light Source Calculation
  - Create virtual sources for beam redirection

Selective Shadow Testing

- Sort potential direct contributions
  - Depends on sources and material
- Test shadows from most to least significant
  - Stop when remainder is below error tolerance
- Add in untested remainder
  - Use statistics to estimate visibility

Selective Shadow Testing (2)

Adaptive Source Subdivision

Virtual Light Source Calculation

Indirect Calculation

- Specular Sampling
  - sample rays over scattering distribution
- Indirect Irradiance Caching
  - sample rays over hemisphere
  - cache irradiance values over geometry
  - reuse for other views and runs

Indirect Calculation (2)

Specular Sampling

Energy-preserving Non-linear Filters

Indirect Irradiance Caching

Indirect Irradiance Gradients

- From hemisphere sampling, we can also compute change w.r.t. position and direction
- Effectively introduces higher-order interpolation method, i.e., cubic vs. linear
- See [Ward & Heckbert, EGWR '92] for details

Irradiance Gradients (2)

Secondary Light Sources

- Impostor surfaces around sources
  - decorative luminaires
  - clear windows
  - complex fenestration
- Computing secondary distributions
  - the **mkillum** program

Impostor Source Geometry

- Simplified geometry for shadow testing and illumination computation
  - fits snugly around real geometry, which is left for rendering direct views

Computing Secondary Distributions

- Start with straight scene description
- Use **mkillum** to compute secondary sources
- Result is a more efficient calculation

Using Pure Monte Carlo

Using Secondary Sources

Participating Media

- Single-scatter approximation
- The mist material type
  - light beams
  - constant density regions
- Rendering method

Single-scatter Approximation

- Computes light scattered into path directly from specified light sources
- Includes absorption and ambient scattering

The *Mist* Material Type

- Demark volumes for light beams
- Can change medium density or scattering properties within a volume

Rendering Method

- After standard ray value is computed:
  - compute ambient in-scattering, out-scattering and absorption along ray path
  - compute in-scattering from any sources identified by *mist* volumes ray passes through
    - this step accounts for anisotropic scattering as well

What About Animation?

- Easy: render frames independently
  - What about motion blur?
  - Also, is this the most efficient approach?
- Better: Image-based frame interpolation
  - **Pinterp** program
    - First released in May 1990 (*Radiance* 1.2)
    - Combines pixels with depth for in-between frames
    - Motion-blur capability
    - Moving objects still a problem

Exploit Human Perception

- Video compression community has studied what motions people notice
- In cases where there is an associated task, we can also exploit *inattentional blindness*
- Image-based motion blur can be extended to objects with a little additional work

Perceptual Rendering Framework

- "Just in time" animation system
- Exploits inattentional blindness and IBR
- Generalizes to other rendering techniques
  - Demonstration system uses *Radiance* ray-tracer
  - Potential for real-time applications
- Error visibility tied to attention and motion

Rendering Framework

Example Frame w/ Task Objects

Error Map Estimation

- Stochastic errors may be estimated from neighborhood samples
- Systematic error bounds may be estimated from knowledge of algorithm behavior
- Estimate accuracy is not critical for good performance

Initial Error Estimate

Image-based Refinement Pass

- Since we know exact motion, IBR works very well in this framework
- Select image values from previous frame
  - Criteria include coherence, accuracy, agreement
- Replace current sample and degrade error
  - Error degradation results in sample retirement

Contrast Sensitivity Model

Error Conspicuity Model

Error Conspicuity Map

Final Sample Density

Implementation Example

- Compared to a standard rendering that finished in the same time, our framework produced better quality on task objects
- Rendering the same high quality over the entire frame would take about 7 times longer using the standard method

Example Animation

- The following animation was rendered at two minutes per frame on a 2000 model G3 laptop computer (Apple PowerBook)
- Many artifacts are intentionally visible, but less so if you are performing the task

Algorithm Visualization

IV. Image Representation

- Traditional graphics image formats
  - Associated problems
- High dynamic-range (HDR) formats
  - Standardization efforts

Traditional Graphics Images

- Usually 8-bit integer range per primary

- *sRGB* color space matches CRT monitors, not human vision

Extended Graphics Formats

- 12 or even 16 bits/primary in TIFF

- Photo editors (i.e., Photoshop™) do not respect this range, treating 65535 as white
- Camera raw formats are an archiving disaster, and should be avoided
- RGB still constrains color gamut

The 24-bit Red Green Blues

- Although 24-bit *sRGB* is reasonably matched to CRT displays, it is a poor match to human vision
  - People can see twice as many colors
  - People can see twice the log range

Q: Why did they base a standard on existing display technology?

A: Because signal processing *used* to be expensive…

High Dynamic Range Images

- High Dynamic Range Images have a wider gamut and contrast than 24-bit RGB
  - Preferably, the gamut and dynamic range covered exceed those of human vision

Advantage 1: an image standard based on human vision won't need frequent updates

Advantage 2: floating point pixels open up a vast new world of image processing

Some HDRI Formats

- *Pixar* 33-bit log-encoded TIFF
- *Radiance* 32-bit RGBE and XYZE
- IEEE 96-bit TIFF & Portable FloatMap
- LogLuv TIFF (24-bit and 32-bit)
- *ILM* 48-bit OpenEXR format

Pixar Log TIFF Codec

Purpose: To store film recorder input

- Implemented in Sam Leffler's TIFF library
- 11 bits each of log red, green, and blue
- 3.8 orders of magnitude in 0.4% steps
- ZIP lossless entropy compression
- Does not cover visible gamut
- Dynamic range marginal for image processing

Radiance RGBE & XYZE

Purpose: To store GI renderings

- Simple format with free source code
- 8 bits each for 3 mantissas + 1 exponent
- 76 orders of magnitude in 1% steps
- Run-length encoding (20% avg. compr.)
- RGBE format does not cover visible gamut
- Color quantization not perceptually uniform
- Dynamic range at expense of accuracy

Radiance Format (.pic, .hdr)

IEEE 96-bit TIFF

Purpose: To minimize translation errors

- Most accurate representation
- Files are enormous
  - 32-bit IEEE floats do not compress well

24-bit LogLuv TIFF Codec

Purpose: To match human vision in 24 bits

- Implemented in Leffler's TIFF library
- 10-bit LogL + 14-bit CIE (u',v') lookup
- 4.8 orders of magnitude in 1.1% steps
- Just covers visible gamut and range
- No compression

24 -bit LogLuv Pixel

32-bit LogLuv TIFF Codec

Purpose: To surpass human vision

- Implemented in Leffler's TIFF library
- 16-bit LogL + 8 bits each for CIE (u',v')
- 38 orders of magnitude in 0.3% steps
- Run-length encoding (30% avg. compr.)
- Allows negative luminance values

32-bit LogLuv Pixel

ILM OpenEXR Format

Purpose: HDR lighting and compositing

- 16-bit/primary floating point
  (sign-e5-m10)
- 9.6 orders of magnitude in 0.1% steps
- Wavelet compression of about 40%
- Negative colors and full gamut RGB
- Open Source I/O library released Fall 2002

ILM's OpenEXR (.exr)

HDRI Post-production

Example HDR Post-processing

Image Representation Future

- JPEG and other 24-bit formats here to stay
- Lossless HDRI formats for high-end
- Compressed HDRI formats are desirable for digital camera applications
  - JPEG 2000 seems like a possible option
    - Adobe doesn't like its proprietary inception
  - Others pushing for a "standard raw sensor" format, but I doubt it would work

V. Image Display

- How do we display an HDR image?
- There are really just two options:
  1. Tone-map HDRI to fit in displayable range
  2. View on a high dynamic-range display
- Many tone-mapping algorithms have been proposed for dynamic-range compression
- But, there are no HDR displays!
  (Or are there?)

HDRI Tone-mapping

- Tone-mapping (a.k.a. tone-reproduction) is a well-studied topic in photography
  - Traditional film curves are carefully designed
- Computer imaging offers many new opportunities for dynamic TRC creation
- Additionally, tone reproduction curves may be manipulated locally over an image

Tone-mapping to LDR Display

- A renderer is like an "ideal" camera
- TM is medium-specific and goal-specific
- Need to consider:
  - Display gamut, dynamic range, and surround
  - What do we wish to simulate?
    - Cinematic camera and film?
    - Human visual abilities and disabilities?

TM Goal: Colorimetric

TM Goal: Match Visibility

TM Goal: Optimize Contrast

One Tone-mapping Approach

- Generate histogram of log luminance
- Redistribute luminance to fit output range
- Optionally simulate human visibility
  - match contrast sensitivity
  - scotopic and mesopic color sensitivity
  - disability (veiling) glare
  - loss of visual acuity in dim environments

Histogram Adjustment

Contrast & Color Sensitivity

Veiling Glare Simulation

Other Tone Mapping Methods

- Retinex-based [Jobson et al. IEEE TIP July '97]
- Psychophysical [Pattanaik et al. Siggraph '98]
- Local Contrast [Ashikhmin, EGWR '02]
- Photographic [Reinhard et al. Siggraph '02]
- Bilateral Filtering [Durand & Dorsey, Siggraph '02]
- Gradient Domain [Fattal et al. Siggraph '02]

High Dynamic-range Display

- Early HDR display technology
  - Industrial high luminance displays (e.g., for air traffic control towers) not really HDR
  - Static stereo viewer for evaluating TMO's
- Emerging HDR display devices
  - Collaborative work at the University of British Columbia in Vancouver, Canada

Static HDR Viewer

HDR Viewer Schematic

Viewer Image Preparation

- Two transparency layers yield 1:104 range
  - B&W "scaling" layer
  - Color "detail" layer
- Resolution difference avoids registration (alignment) problems
- 120° hemispherical fisheye perspective
- Correction for chromatic aberration

Example Image Layers

UBC Structured Surface Physics Lab HDR Display

- First generation DLP/LCD prototype
  - 1024x768 resolution
  - 10,000:1 dynamic range
  - 7,000 cd/m2 maximum luminance
- Next generation device w/ LED backlight
  - Flat-panel design presented at SID
  - 10,000:1 DR and 10,000 max. luminance

UBC HDR Display Prototype

VI. Image-based Techniques

- High dynamic-range photography
  - Using *Photosphere*
- Image-based lighting
- Image-based rendering

HDR Photograhy

- Standard digital cameras capture about 2 orders of magnitude in sRGB color space
- Using multiple exposures, we can build up high dynamic range image of static scene
- In the future, manufacturers may build HDR imaging into camera hardware

Hand-held HDR Photography

- Use “auto-bracketing” exposure feature
- Align exposures horizontally and vertically
- Deduce camera response function using [Mitsunaga & Nayar ‘99] polynomial fit
- Recombine images into HDR image
- Optionally remove lens flare

Auto-bracket Exposures

LDR Exposure Alignment

Estimated Camera Response

Combined HDR Image

Tone-mapped Display

Best Single Exposure

Lens Flare Removal

Photosphere HDRI Browser

- Browses High Dynamic Range Images
  - *Radiance* RGBE format
  - TIFF LogLuv and floating point formats
  - OpenEXR short float format
- Makes HDR images from bracketed exposures
- Maintains Catalog Information
  - Subjects, keywords, albums, comments, etc.
- Tracks Image Files
  - Leaves file management & modification to user

Realized Features

- Fast, interactive response
- Thumbnails accessible when images are not
- Interprets Exif header information
- Builds photo albums & web pages
- Displays & edits image information
- Provides drag & drop functionality
- User-defined database fields

Unrealized Features

- Accurate color reproduction on all devices
- Plug-in interface for photo printing services
- Linux and Windows versions
- More supported image formats
  - Currently JPEG, TIFF, *Radiance*, OpenEXR

Browser Layout

Viewer Layout

Info Window Layout

Browser Files

Browser Architecture

Photosphere Demo

Image-based Lighting

- Photograph silver sphere using HDR method
- Place as environment map in scene to render
- Sample map to obtain background values

Image-based Rendering

- Mixed reality is the future for graphics
- High dynamic-range imaging is the key
- Accuracy in rendering is also critical for seamless integration
- A lot of work has been done in the areas of image-based lighting and rendering, but we've only scratched the surface
  - Films like *The Matrix* rely heavily on IBL/IBR

IBR/IBL Example

VII. Conclusions

- Two paths to realism:
  1. Work like nuts until it "looks OK," or
  2. Apply psychophysics of light and vision
- As authors of rendering software, we can save users a lot of (1) with a little of (2)
- Real numbers are needed for physical simulation, as values are unbounded
- The eye and brain are analog devices

Further Reference

- www.anyhere.com/gward
  - publication list with online links
  - LogLuv TIFF pages and images
- www.debevec.org
  - publication list with online links
  - *Radiance* RGBE images and light probes
  - *HDRshop* and related tools
- www.idruna.com
  - *Photogenics* HDR image editor
- radsite.lbl.gov/radiance
  - *Radiance* rendering software and links