## Advanced Computer Graphics (Spring 2013)

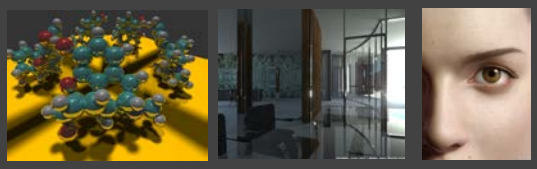
CS 283, Lecture 13: High Quality Real-Time Rendering

Ravi Ramamoorthi

http://inst.eecs.berkeley.edu/~cs283/sp13

## Motivation

- Today, we create photorealistic computer graphics
  - Complex geometry, realistic lighting, materials, shadows
  - Computer-generated movies/special effects (difficult to tell real from rendered…)



- ***But algorithms are very slow (hours to days)***

## Real-Time Rendering

- Goal is interactive rendering. Critical in many apps
  - Games, visualization, computer-aided design, …
- So far, focus on complex geometry

- ***Chasm between interactivity, realism***

## Evolution of 3D graphics rendering

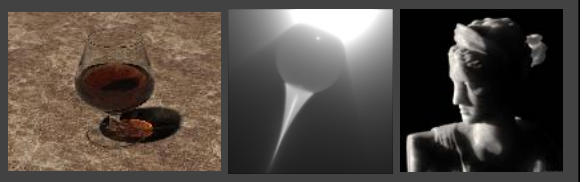
Interactive 3D graphics pipeline as in OpenGL
- Earliest SGI machines (Clark 82) to today
- Most of focus on more geometry, texture mapping
- Some tweaks for realism (shadow mapping, accum. buffer)

SGI Reality Engine 93 (Kurt Akeley)

## Offline 3D Graphics Rendering

Ray tracing, radiosity, global illumination
- High realism (global illum, shadows, refraction, lighting,..)
- But historically very slow techniques

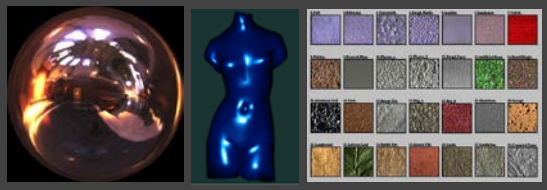
*"So, while you and your children's children are waiting for ray tracing to take over the world, what do you do in the meantime?"* Real-Time Rendering 1st ed.



Pictures courtesy Henrik Wann Jensen

## New Trend: Acquired Data

- Image-Based Rendering: Real/precomputed images as input
- Also, acquire geometry, lighting, materials from real world
- Easy to obtain or precompute lots of high quality data. But how do we represent and reuse this for (real-time) rendering?



## 10 years ago

- High quality rendering: ray tracing, global illum., image-based rendering (This is what a rendering course would cover)
- Real-Time rendering: Interactive 3D geometry with simple texture mapping, maybe fake shadows (OpenGL, DirectX)
- Complex environment lighting, real materials (velvet, satin, paints), soft shadows, caustics often omitted in both
- *Realism, interactivity at cross purposes*

## Today

- Vast increase in CPU power, modern instrs (SSE, Multi-Core)
  - Real-time raytracing techniques are possible (even on hardware: NVIDIA Optix)
- 4th generation of graphics hardware is *programmable*
  - (First 3 gens were wireframe, shaded, textured)
  - Modern nVidia, ATI cards allow vertex, fragment shaders
- Great deal of current work on acquiring and rendering with realistic lighting, materials… [Especially at Berkeley]
- *Focus on quality of rendering, not quantity of polygons, texture*

## Goals

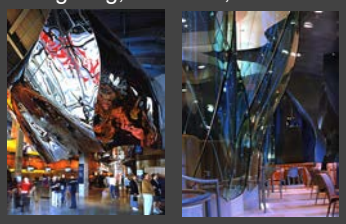
- Overview of basic techniques for high-quality real-time rendering
- Survey of important concepts and ideas, but do not go into details of writing code
- Some pointers to resources, others on web
- One possibility for assignment 3, will need to think about some ideas on your own

## Outline

- *Motivation and Demos*
- Programmable Graphics Pipeline
- Shadow Maps
- Environment Mapping

## High quality real-time rendering

- Photorealism, not just more polygons
- Natural lighting, materials, shadows



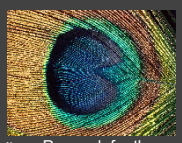
Interiors by architect Frank Gehry. Note rich lighting, ranging from localized sources to reflections off vast sheets of glass.

## High quality real-time rendering

- Photorealism, not just more polygons
- Natural lighting, materials, shadows

Glass Vase

Glass Star (courtesy Intel)



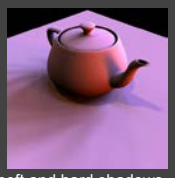
Peacock feather

Real materials diverse and not easy to represent by simple parameteric models. Want to support measured reflectance.

## High quality real-time rendering

- Photorealism, not just more polygons
- Natural lighting, materials, shadows

small area light, sharp shadows
Agrawala et al. 00



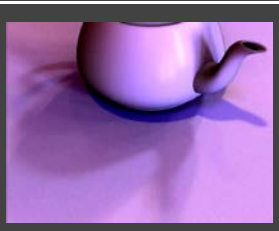
soft and hard shadows
Ng et al. 03

Natural lighting creates a mix of soft diffuse and hard shadows.

## Programmable Graphics Hardware

## Precomputation-Based Methods

- Static geometry
- Precomputation



- Real-Time Rendering (relight all-frequency effects)
- Involves sophisticated representations, algorithms

## Relit Images

Ng, Ramamoorthi, Hanrahan 04

## Video: Real-Time Relighting

## Interactive RayTracing

Advantages
- Very complex scenes relatively easy (hierarchical bbox)
- Complex materials and shading for free
- Easy to add global illumination, specularities etc.

Disadvantages
- Hard to access data in memory-coherent way
- Many samples for complex lighting and materials
- Global illumination possible but expensive

Modern developments: Leverage power of modern CPUs, develop cache-aware, parallel implementations

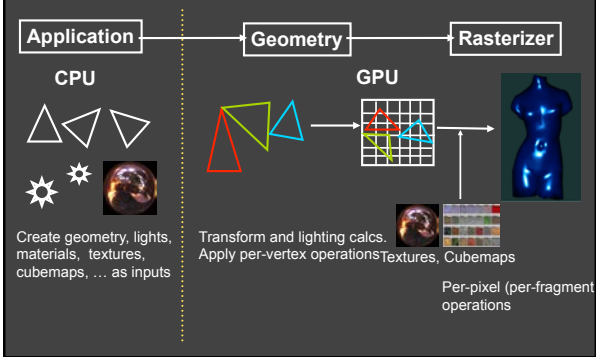
Demo http://www.youtube.com/watch?v=blfxl1cVOzU

Interactive Global Illumination Demo Video

## Outline

- Motivation and Demos
- *Programmable Graphics Pipeline*
- Shadow Maps
- Environment Mapping

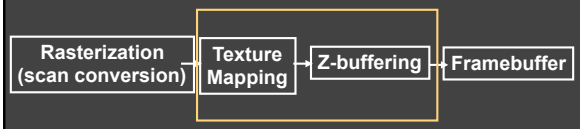
## Basic Hardware Pipeline




## Geometry or Vertex Pipeline


These fixed function stages can be replaced by a general per-vertex calculation using vertex shaders in modern programmable hardware

## Pixel or Fragment Pipeline




These fixed function stages are replaced by a general per-fragment calculation using fragment shaders in programmable hardware

## Shading Languages

- Vertex / Fragment shading described by small program
- Written in language similar to C but with restrictions
- Long history. Cook's paper on Shade Trees, Renderman for offline rendering
- Stanford Real-Time Shading Language, work at SGI
- Cg from NVIDIA, HLSL
- GLSL directly compatible with OpenGL 2.0 (used in 184)

## Outline

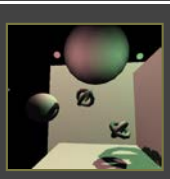
- Motivation and Demos
- Programmable Graphics Pipeline
- *Shadow Maps*
- Environment Mapping

## Shadow and Environment Maps

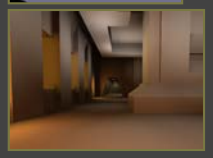
- Basic methods to add realism to interactive rendering
- Shadow maps: image-based way to add hard shadows
  - Very old technique. Originally Williams 78
  - Many recent (and older) extensions
  - Widely used even in software rendering (RenderMan)
  - Simple alternative to raytracing for shadows
- Environment maps: image-based complex lighting
  - Again, very old technique. Blinn and Newell 76
  - Huge amount of recent work
- Together, give many of realistic effects we want
  - **But cannot be easily combined!!**

## Common Real-time Shadow Techniques

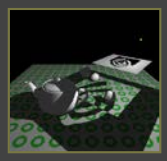
*Projected planar shadows*



*Shadow volumes*



*Light maps*



*Hybrid approaches*

This slide, others courtesy Mark Kilgard

## Problems

Mostly tricks with lots of limitations

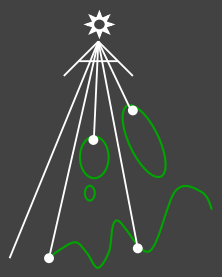
- Projected planar shadows
  works well only on flat surfaces
- Stenciled shadow volumes
  determining the shadow volume is hard work
- Light maps
  totally unsuited for dynamic shadows
- In general, hard to get everything shadowing everything

## Shadow Mapping

- Lance Williams: Brute Force in image space (shadow maps in 1978, but other similar ideas like Z buffer, bump mapping using textures and so on)
- Completely image-space algorithm
  - no knowledge of scene's geometry is required
  - must deal with aliasing artifacts
- Well known software rendering technique
  - Basic shadowing technique for Toy Story, etc.

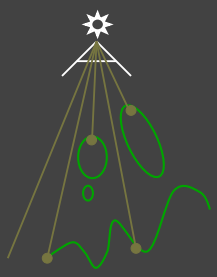
## Phase 1: Render from Light

- Depth image from light source



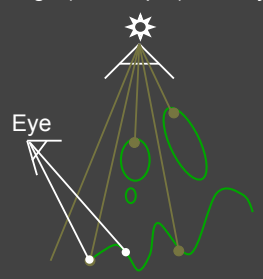
## Phase 1: Render from Light

- Depth image from light source



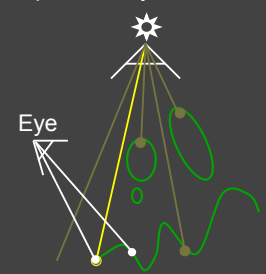
## Phase 2: Render from Eye

- Standard image (with depth) from eye




## Phase 2+: Project to light for shadows

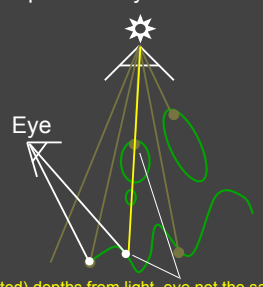
- Project visible points in eye view back to light source




(Reprojected) depths match for light and eye. VISIBLE

## Phase 2+: Project to light for shadows

- Project visible points in eye view back to light source

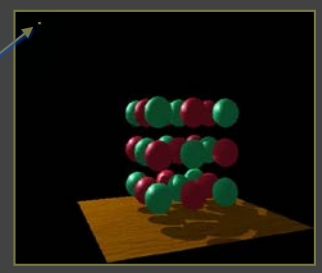



(Reprojected) depths from light, eye not the same. BLOCKED!!

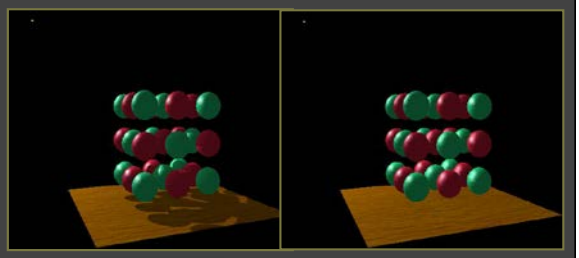
## Visualizing Shadow Mapping

- A fairly complex scene with shadows

***the point light source***



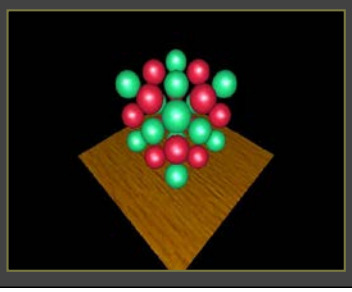
## Visualizing Shadow Mapping

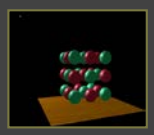


***with shadows*** ***without shadows***

## Visualizing Shadow Mapping

- The scene from the light's point-of-view

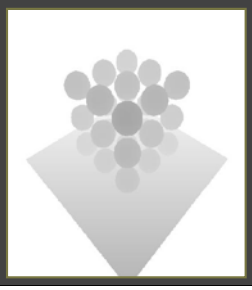




***FYI: from the eye's point-of-view again***

## Visualizing Shadow Mapping

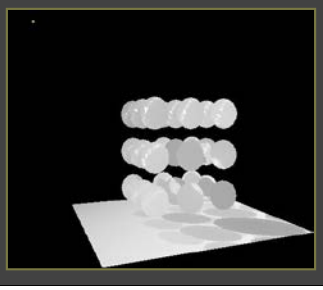
- The depth buffer from the light's point-of-view

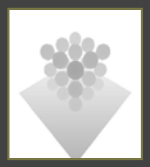


*FYI: from the light's point-of-view again*

## Visualizing Shadow Mapping

- Projecting the depth map onto the eye's view



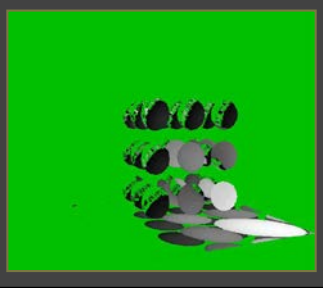


*FYI: depth map for light's point-of-view again*

## Visualizing Shadow Mapping

- Comparing light distance to light depth map

*Green is where the light planar distance and the light depth map are approximately equal*

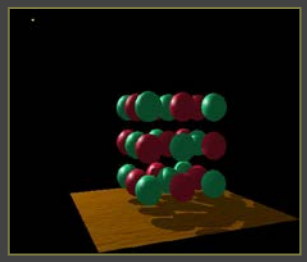


*Non-green is where shadows should be*

## Visualizing Shadow Mapping

- Scene with shadows

*Notice how specular highlights never appear in shadows*



*Notice how curved surfaces cast shadows on each other*

## Hardware Shadow Map Filtering

- Normal texture filtering just averages color components
- Averaging depth values does NOT work
- "Percentage Closer" filtering [Reeves 87]
  - Hardware performs comparison for each sample
  - Then, averages results of comparisons
  - Provides anti-aliasing at shadow map edges
    - Not soft shadows for umbra/penumbra case
    - But often used to fake soft shadows

## Hardware Shadow Map Filtering

*GL_NEAREST: blocky*

*GL_LINEAR: antialiased edges*

*Low shadow map resolution used to heighten filtering artifacts*

## Problems with shadow maps

- Hard shadows (point lights only)
- Quality depends on shadow map resolution (general problem with image-based techniques)
- Involves equality comparison of floating point depth values means issues of scale, bias, tolerance
- Many recent extensions

## Outline

- Motivation and Demos
- Programmable Graphics Pipeline
- Shadow Maps
- *Environment Mapping*

## Reflection Maps

**Blinn and Newell, 1976**

## Environment Maps

Miller and Hoffman, 1984
Later, Greene 86, Cabral et al, Debevec 97, …

## Environment Maps

***Interface*, Chou and Williams (ca. 1985)**

## Environment Maps

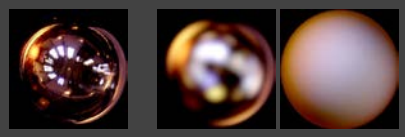
**Cylindrical Panoramas**

**Cubical Environment Map**

**180 degree fisheye**
**Photo by R. Packo**

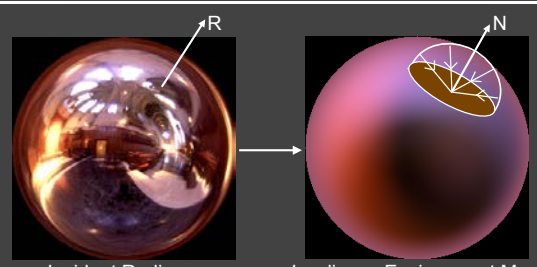
## Reflectance Maps

- Reflectance Maps (Index by N)
- Horn, 1977
- Irradiance (N) and Phong (R) Reflection Maps
- Miller and Hoffman, 1984



Mirror Sphere  Chrome Sphere  Matte Sphere

## Irradiance Environment Maps




Incident Radiance (Illumination Environment Map)  Irradiance Environment Map

## Assumptions

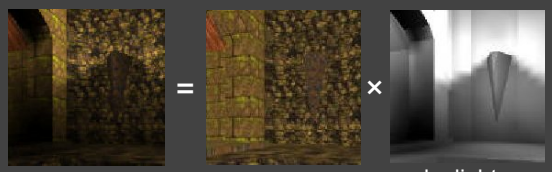
- Diffuse surfaces
- Distant illumination
- No shadowing, interreflection

Hence, Irradiance a function of surface normal

## Diffuse Reflection

$$B = \rho E$$

Radiosity (image intensity)  Reflectance (albedo/texture)  Irradiance (incoming light)

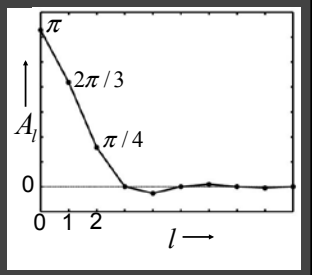



quake light map

## Analytic Irradiance Formula

Lambertian surface acts like low-pass filter

$$E_{lm} = A_l L_{lm}$$




Ramamoorthi and Hanrahan 01
Basri and Jacobs 01

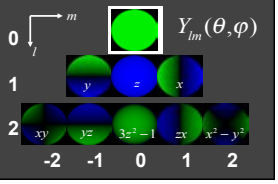
$$A_l = 2\pi \frac{(-1)^{\frac{l}{2}-1}}{(l+2)(l-1)}\left[\frac{l!}{2^l\left(\frac{l}{2}!\right)^2}\right] \quad l \text{ even}$$

## 9 Parameter Approximation

Exact image

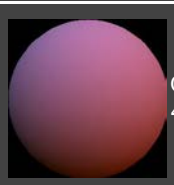
Order 0
1 term

**RMS error = 25 %**



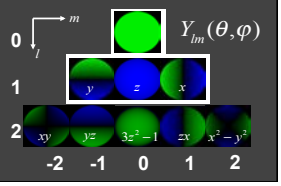
## 9 Parameter Approximation

Exact image



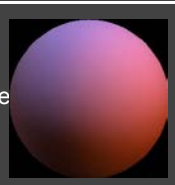
Order 1
4 terms

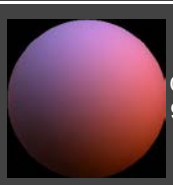
**RMS Error = 8%**




## 9 Parameter Approximation

Exact image

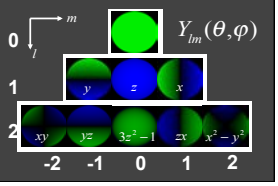




Order 2
9 terms

**RMS Error = 1%**

For any illumination, average error < 3% [Basri Jacobs 01]




## Real-Time Rendering

$$E(n) = n^t M n$$

Simple procedural rendering method (no textures)

- Requires only matrix-vector multiply and dot-product
- In software or NVIDIA vertex programming hardware

Widely used in Games (AMPED for Microsoft Xbox), Movies (Pixar, Framestore CFC, …)

```
surface float1 irradmat (matrix4 M, float3 v) {
        float4 n = {v , 1} ;
        return dot(n , M*n) ;
}
```

## Environment Map Summary

- Very popular for interactive rendering
- Extensions handle complex materials
- Shadows with precomputed transfer

- But cannot directly combine with shadow maps
- Limited to distant lighting assumption

## Resources

- OpenGL red book (latest includes GLSL)
- Web tutorials: http://www.lighthouse3d.com/opengl/glsl/
- Older books: OpenGL Shading Language book (Rost), The Cg Tutorial, …
- http://www.realtimerendering.com
  - Real-Time Rendering by Moller and Haines
- Debevec http://www.debevec.org/ReflectionMapping/
  - Links to Miller and Hoffman original, Haeberli/Segal
- http://www.cs.berkeley.edu/~ravir/papers/envmap
  - Also papers by Heidrich, Cabral, …
- Lots of information available on web…