Ray Tracing
Global Illumination

Wolfgang Heidrich

Course News

Assignment 3 (project)

- Due April 1
- Demos in labs April 2-7

Reading

- Chapter 10 (ray tracing), except 10.8-10.10
- Chapter 14 (global illumination)

Course Topics for the Rest of the Term

Ray-tracing & Global Illumination

- This week

Parametric Curves/Surfaces

- March 30/April 1
- Taught by Robert Bridson - I will be at a conference

Overview of current research

- April 3/6 (Ivo Ihrke – I am still at conference)

April 8 – Final Q&A (I will be back for that)

Wolfgang Heidrich

Ray-Tracing

Basic Algorithm (Whithead):

for every pixel p_i {
  Generate ray r from camera position through pixel p_i
  p_i= background color
  for every object o in scene {
    if(r intersects o && intersection is closer than previously found intersections)
      Compute lighting at intersection point, using local normal and material properties; store result in p_i
  }
}

Wolfgang Heidrich

Ray-Tracing

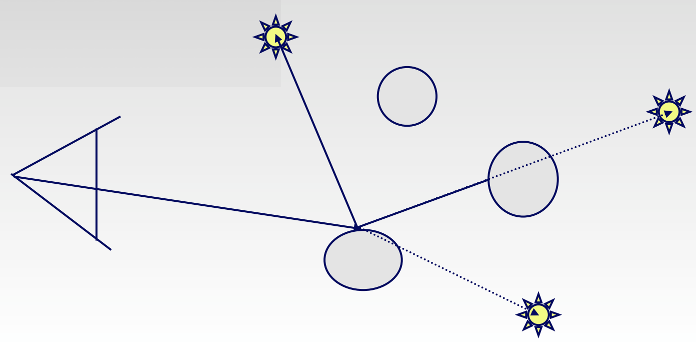
Issues:

- Generation of rays
- Intersection of rays with geometric primitives
- Geometric transformations
- Lighting and shading
- Efficient data structures so we don't have to test intersection with *every* object

Ray-Tracing Shadows

Approach:

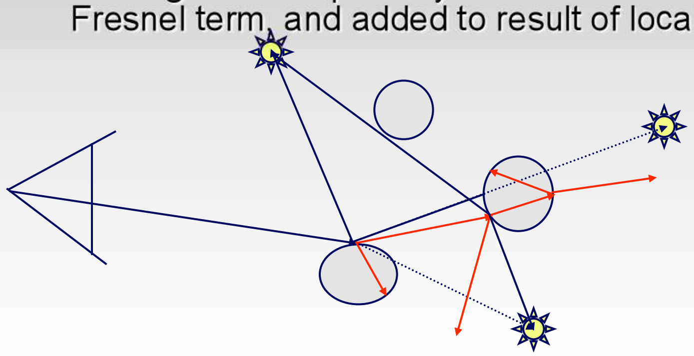
- To test whether point is in shadow, send out <u>*shadow rays*</u> to all light sources
  - *If ray hits another object, the point lies in shadow*



Ray-Tracing Reflections/Refractions

Approach:

- Send rays out in reflected and refracted direction to gather incoming light
- That light is multiplied by local surface color and Fresnel term, and added to result of local shading



Wolfgang Heidrich

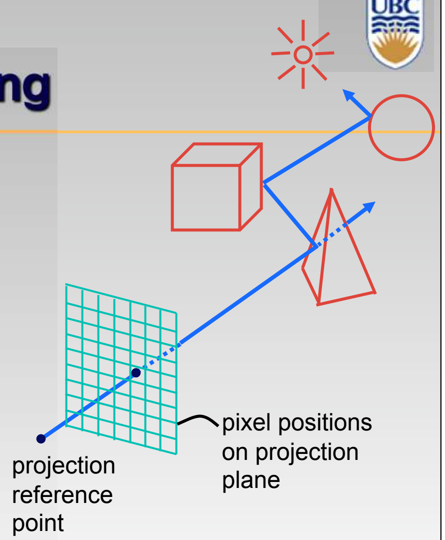
Recursive Ray Tracing

Ray tracing can handle

- Reflection (chrome)
- Refraction (glass)
- Shadows

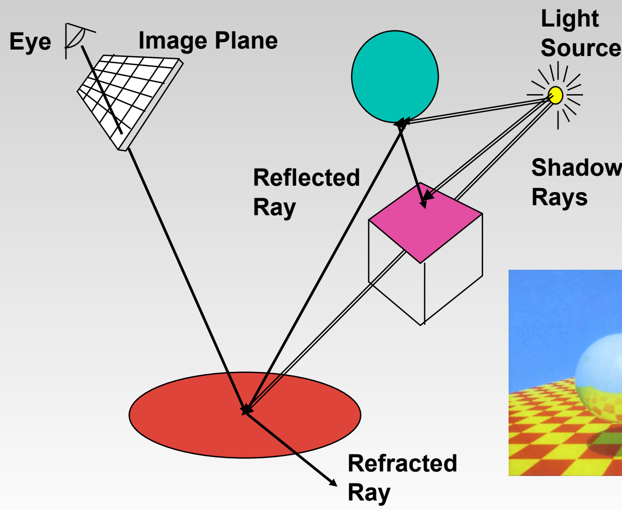
Spawn secondary rays

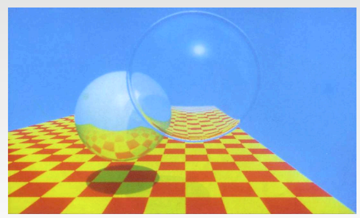
- Reflection, refraction
  - *If another object is hit, recurse to find its color*
- Shadow
  - *Cast ray from intersection point to light source, check if intersects another object*



Wolfgang Heidrich

Recursive Ray-Tracing





Whitted, 1980

Wolfgang Heidrich

Recursive Ray-Tracing Algorithm

```
RayTrace(r,scene)
obj := FirstIntersection(r,scene)
if (no obj)  return BackgroundColor;
else begin
   if ( Reflect(obj) ) then
     reflect_color := RayTrace(ReflectRay(r,obj));
   else
     reflect_color := Black;
  if ( Transparent(obj) ) then
     refract_color := RayTrace(RefractRay(r,obj));
   else
     refract_color := Black;
  return Shade(reflect_color,refract_color,obj);
end;
```

Wolfgang Heidrich

Recursive Ray-Tracing

What rays need to be traced?

- Materials with diffuse/Phong reflection component:
  - *Shadow rays to each light source (binary visibility only)*
- Materials with mirror reflection component:
  - *Reflection ray (i.e. recursion)*
- Materials with transmissive component:
  - *Refraction ray (recursion)*

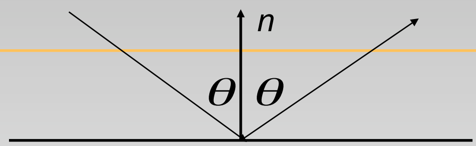
In practice, materials can have any combination of the above components

Algorithm Termination Criteria

Termination criteria

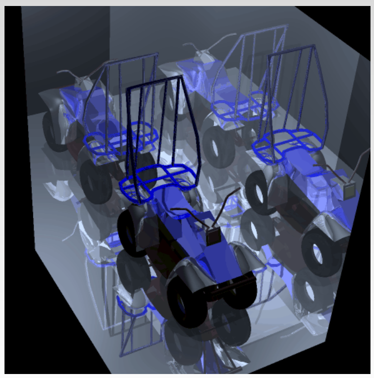
- No intersection
- Reach maximal depth
  - *Number of bounces*
- Contribution of secondary ray attenuated below threshold
  - *Each reflection/refraction attenuates ray*

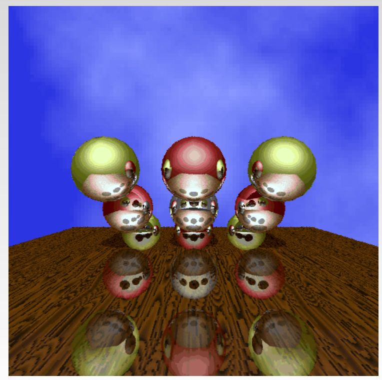
Reflection



Mirror effects

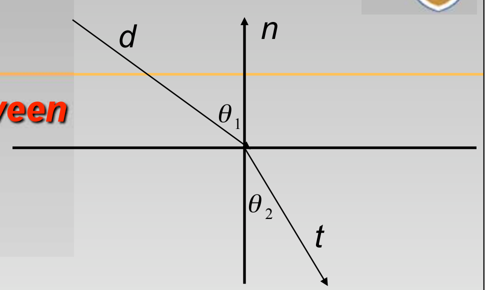
- Perfect specular reflection





Wolfgang Heidrich

Refraction

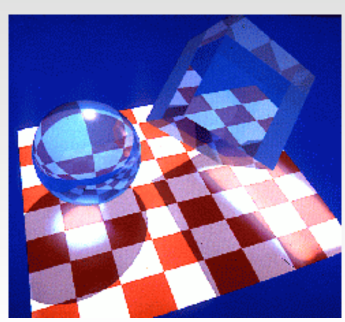


Happens at interface between transparent object and surrounding medium

- E.g. glass/air boundary

Snell's Law

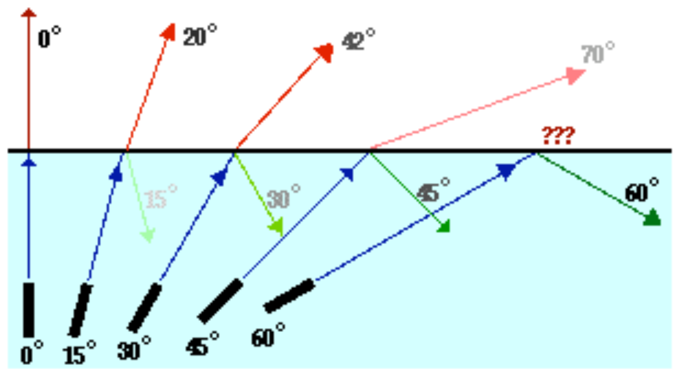
- $c_1 \sin\theta_1 = c_2 \sin\theta_2$
- Light ray bends based on refractive indices c_1, c_2



Wolfgang Heidrich

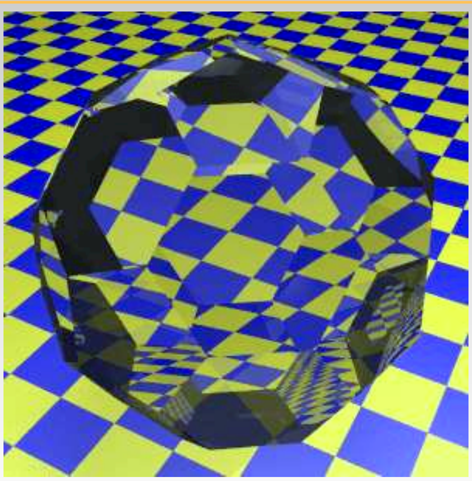
Total Internal Reflection

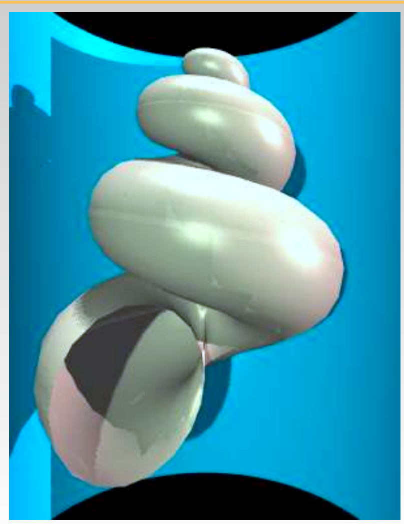
As the angle of incidence increases from 0 to greater angles ...



...the refracted ray becomes dimmer (there is less refraction)
...the reflected ray becomes brighter (there is more reflection)
...the angle of refraction approaches 90 degrees until finally a refracted ray can no longer be seen.

Ray-Tracing Example Images





Wolfgang Heidrich

Ray-Tracing Terminology

Terminology:

- Primary ray: ray starting at camera
- Shadow ray
- Reflected/refracted ray
- Ray tree: all rays directly or indirectly spawned off by a single primary ray

Note:

- Need to limit maximum depth of ray tree to ensure termination of ray-tracing process!

Wolfgang Heidrich

Ray-Tracing

Issues:

- Generation of rays
- Intersection of rays with geometric primitives
- Geometric transformations
- Lighting and shading
- Efficient data structures so we don't have to test intersection with *every* object

Wolfgang Heidrich

Ray Tracing

Data Structures

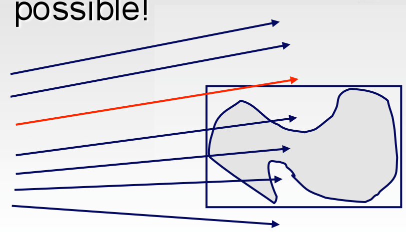
- Goal: reduce number of intersection tests per ray
- Lots of different approaches:
  - *(Hierarchical) bounding volumes*
  - *Hierarchical space subdivision*
    - Oct-tree, k-D tree, BSP tree

Wolfgang Heidrich

Bounding Volumes

Idea:

- Rather than testing every ray against a potentially very complex object (e.g. triangle mesh), do a quick *conservative* test first which eliminates most rays
  - *Surround complex object by simple, easy to test geometry (typically sphere or axis-aligned box)*
    - Want to make bounding volume as tight as possible!

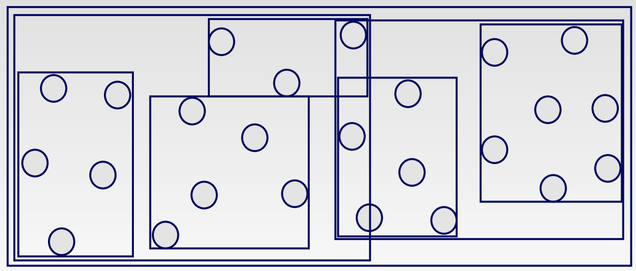


Wolfgang Heidrich

Hierarchical Bounding Volumes

Extension of previous idea:

- Use bounding volumes for groups of objects



Wolfgang Heidrich

Spatial Subdivision Data Structures

Bounding Volumes:

- Find simple object completely enclosing complicated objects
  - *Boxes, spheres*
- Hierarchically combine into larger bounding volumes

Spatial subdivision data structure:

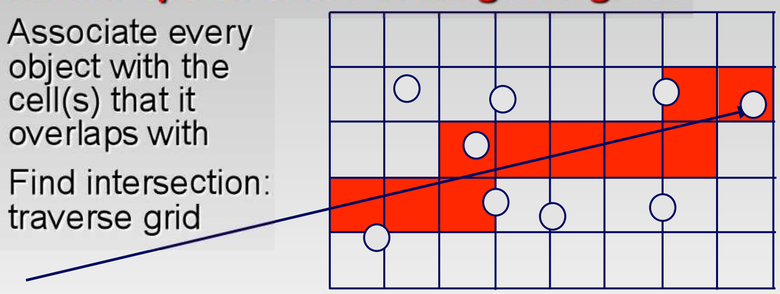
- Partition the whole space into cells
  - *Grids, oct-trees, (BSP trees)*
- Simplifies and accelerates traversal
- Performance less dependent on order in which objects are inserted

Wolfgang Heidrich

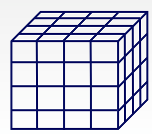
Regular Grid

Subdivide space into rectangular grid:

- Associate every object with the cell(s) that it overlaps with
- Find intersection: traverse grid



In 3D: regular grid of cubes (voxels):

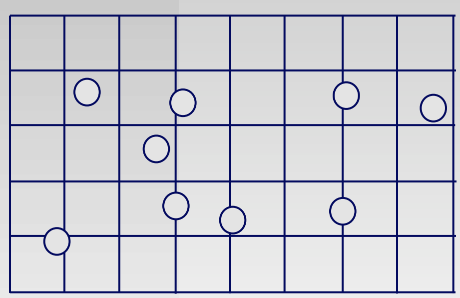


Wolfgang Heidrich

Creating a Regular Grid

Steps:

- Find bounding box of scene
- Choose grid resolution in x, y, z
- Insert objects
- Objects that overlap multiple cells get referenced by all cells they overlap

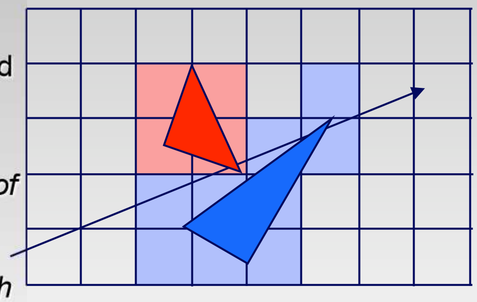


Wolfgang Heidrich

Grid Traversal

Traversal:

- Start at ray origin
- While no intersection found
  - *Go to next grid cell along ray*
  - *Compute intersection of ray with all objects in the cell*
  - *Determine closest such intersection*
  - ***Check if that intersection is inside the cell***
  - *If so, terminate search*



Wolfgang Heidrich

Traversal

Note:

- This algorithm calls for computing the intersection points multiple times (once per grid cell)
- In practice: store intersections for a (ray, object) pair once computed, reuse for future cells

Wolfgang Heidrich

Regular Grid Discussion

Advantages?

- Easy to construct
- Easy to traverse

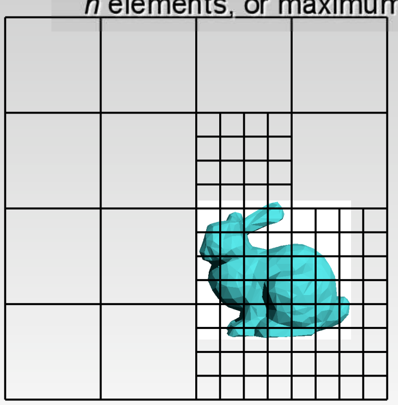
Disadvantages?

- May be only sparsely filled
- Geometry may still be clumped

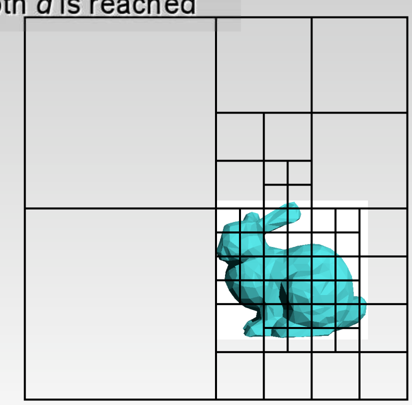
Wolfgang Heidrich

Adaptive Grids

- Subdivide until each cell contains no more than *n* elements, or maximum depth *d* is reached



Nested Grids



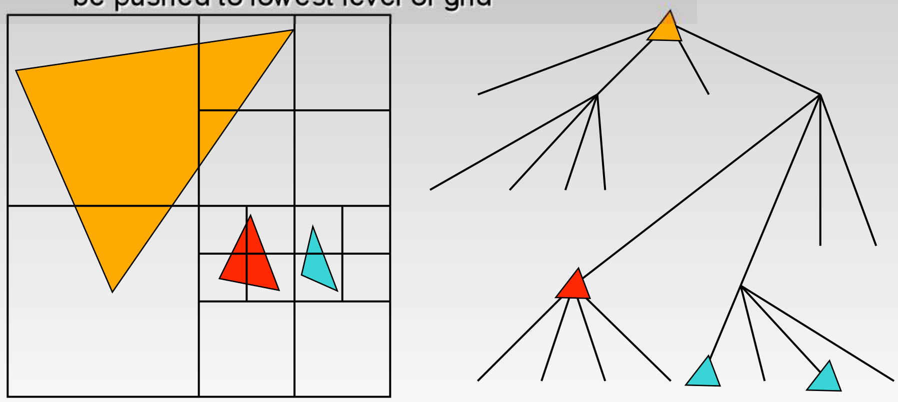
Octree/(Quadtree)

- This slide and the next are curtsey of Fredo Durand at MIT

Wolfgang Heidrich

Primitives in an Adaptive Grid

- Can live at intermediate levels, or be pushed to lowest level of grid



Octree/(Quadtree)

Wolfgang Heidrich

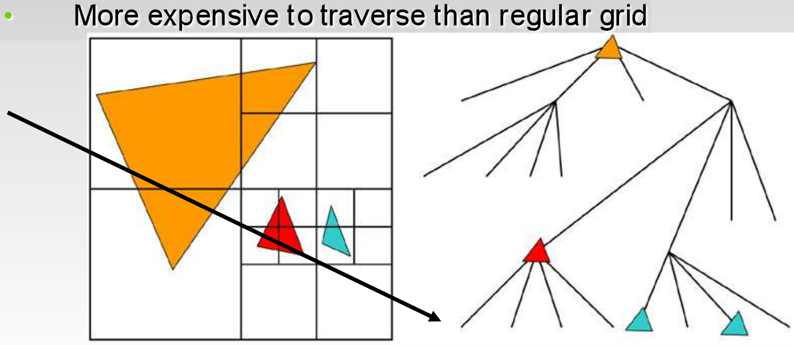
Adaptive Grid Discussion

Advantages

- Grid complexity matches geometric density

Disadvantages

- More expensive to traverse than regular grid



Wolfgang Heidrich

Coming Up...

Friday:

- More global illumination

Monday/Wednesday:

- Curves & surfaces

Wolfgang Heidrich

Soft Shadows & Global Illumination

CPSC 314

Area Light Sources

So far:

- All lights were either point-shaped or directional
  - *Both for ray-tracing and the rendering pipeline*
- Thus, at every point, we only need to compute lighting formula and shadowing for ONE light direction

In reality:

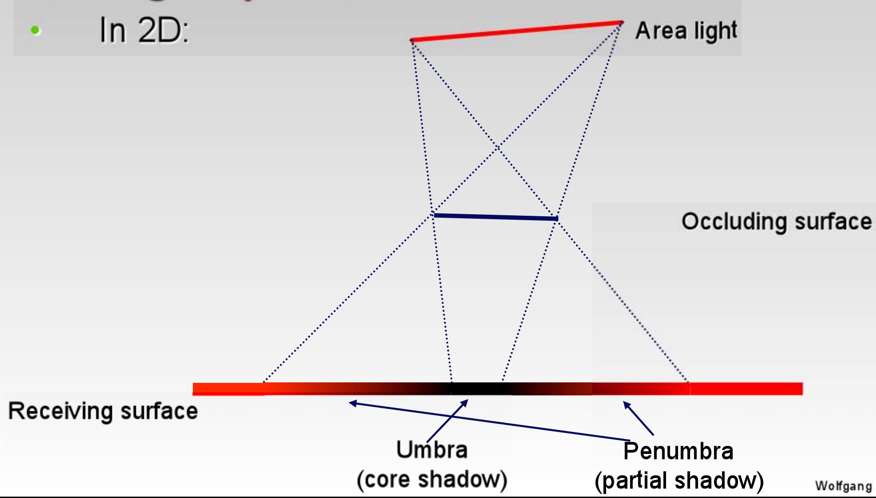
- All lights have a finite area
- Instead of just dealing with one direction, we now have to integrate over all directions that go to the light source

Wolfgang Heidrich

Area Light Sources

Area lights produce soft shadows:

- In 2D:



Wolfgang Heidrich

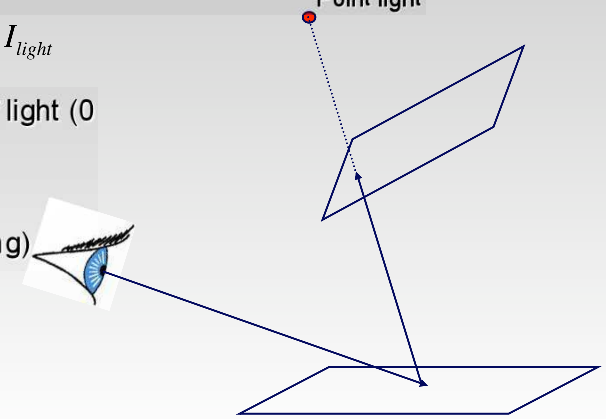
Area Light Sources

Point lights:

- Only one light direction:

$$I_{reflected} = \rho \cdot V \cdot I_{light}$$

- V is visibility of light (0 or 1)
- ρ is lighting model (e.g. diffuse or Phong)



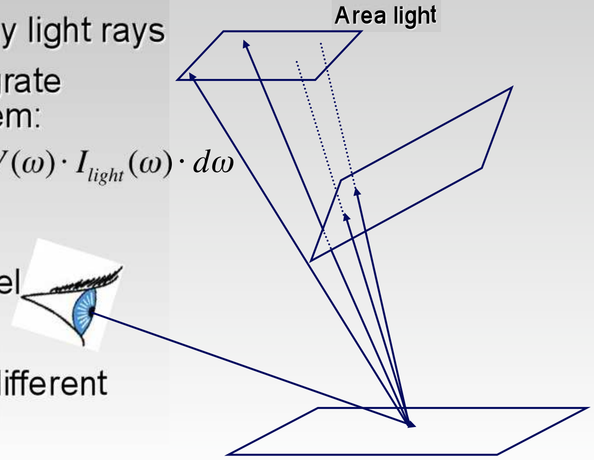
Are Light Sources

Area Lights:

- Infinitely many light rays
- Need to integrate over all of them:

$$I_{reflected} = \int_{\substack{light\\directions}} \rho(\omega) \cdot V(\omega) \cdot I_{light}(\omega) \cdot d\omega$$

- Lighting model visibility and light intensity can now be different for every ray!



Integrating over Light Source

Rewrite the integration

- Instead of integrating over directions

$$I_{reflected} = \int_{\substack{light\\directions}} \rho(\omega) \cdot V(\omega) \cdot I_{light}(\omega) \cdot d\omega$$

we can integrate over points on the light source

$$I_{reflected}(q) = \int_{s,t} \frac{\rho(p-q) \cdot V(p-q)}{|p-q|^2} \cdot I_{light}(p) \cdot ds \cdot dt$$

where q: point on reflecting surface, p= F(s,t) is a point on the area light

- *We are integrating over p*
- *Denominator: quadratic falloff!*

Wolfgang Heidrich

Integration

Problem:

- Except for the simplest of scenes, either integral is not solvable analytically!
- This is mostly due to the visibility term, which could be arbitrarily complex depending on the scene

So:

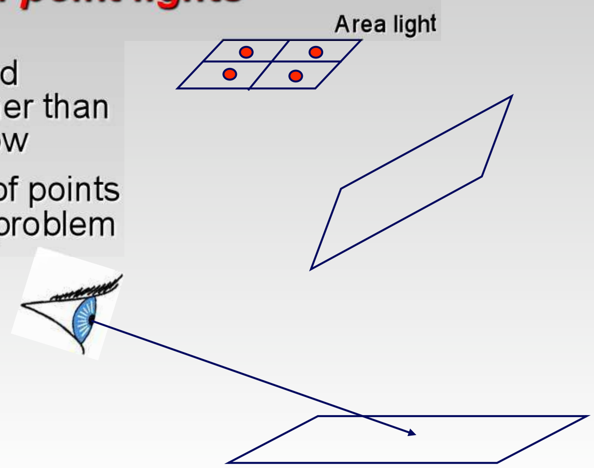
- Use numerical integration
- Effectively: approximate the light with a whole number of point lights

Wolfgang Heidrich

Numerical Integration

Regular grid of point lights

- Problem:
  will see 4 hard shadows rather than as soft shadow
- Need LOTS of points to avoid this problem

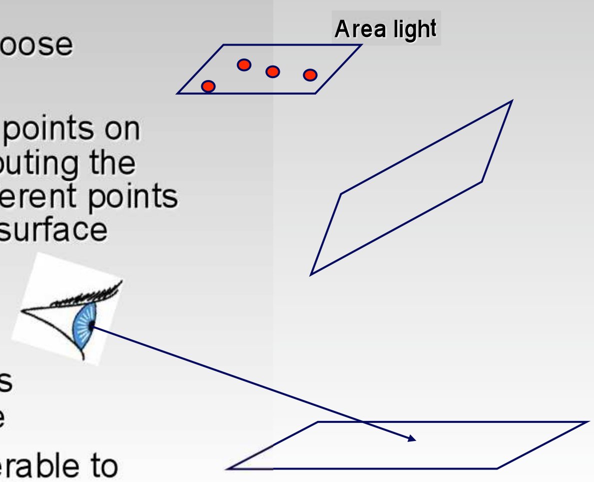


Wolfgang Heidrich

Monte Carlo Integration

Better:

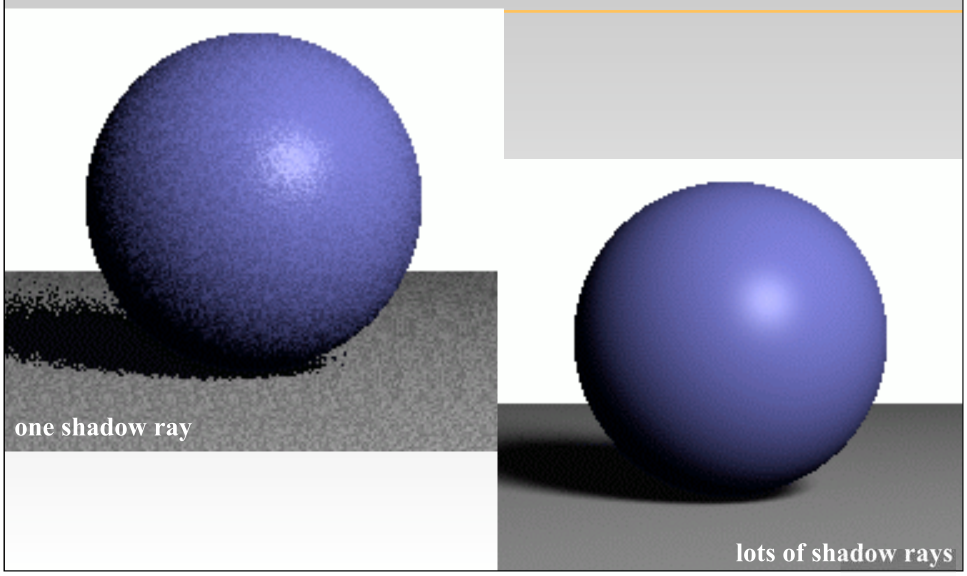
- Randomly choose the points
- Use different points on light for computing the lighting in different points on reflecting surface



- This produces random noise
- Visually preferable to structured artifacts

Wolfgang Heidrich

Monte Carlo Integration



Monte Carlo Integration

Formally:

- Approximate integral with finite sum

$$I_{reflected}(q) = \int_{s,t} \frac{\rho(p-q) \cdot V(p-q)}{|p-q|^2} \cdot I_{light}(p) \cdot ds \cdot dt$$

$$\approx \frac{A}{N} \sum_{i=1}^{N} \frac{\rho(p_i-q) \cdot V(p_i-q)}{|p_i-q|^2} \cdot I_{light}(p_i)$$

where

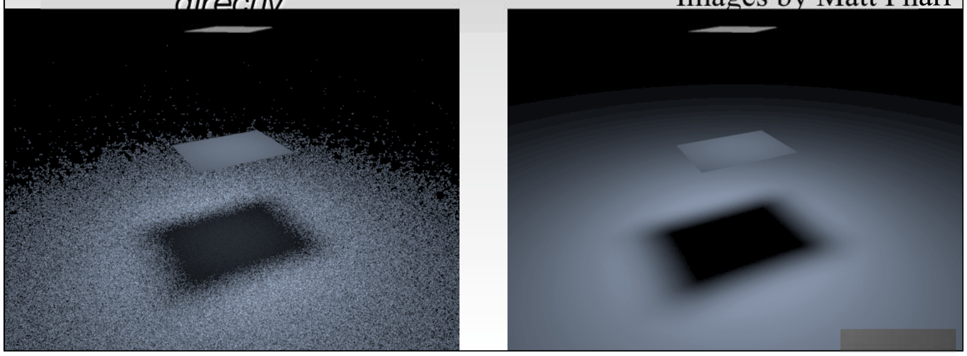
- *The p_i are randomly chosen on the light source*
  - With equal probability!
- *A is the total area of the light*
- *N is the number of samples (rays)*

Sampling

Sample directions vs. sample light source

- Most directions do not correspond to points on the light source
  - *Thus, variance will be higher than sampling light directly*

Images by Matt Pharr



Monte Carlo Integration

Note:

- This approach of approximating lighting integrals with sums over randomly chosen points is much more flexible than this!
- In particular, it can be used for global illumination
  - *Light bouncing off multiple surfaces before hitting the eye*

Wolfgang Heidrich

Global Illumination

So far:

- Have considered only light directly coming form the light sources
  - *As well as mirror reflections, refraction*

In reality:

- Light bouncing off diffuse and/or glossy surfaces also illuminates other surfaces
  - *This is called global illumination*

Wolfgang Heidrich

Direct Illumination

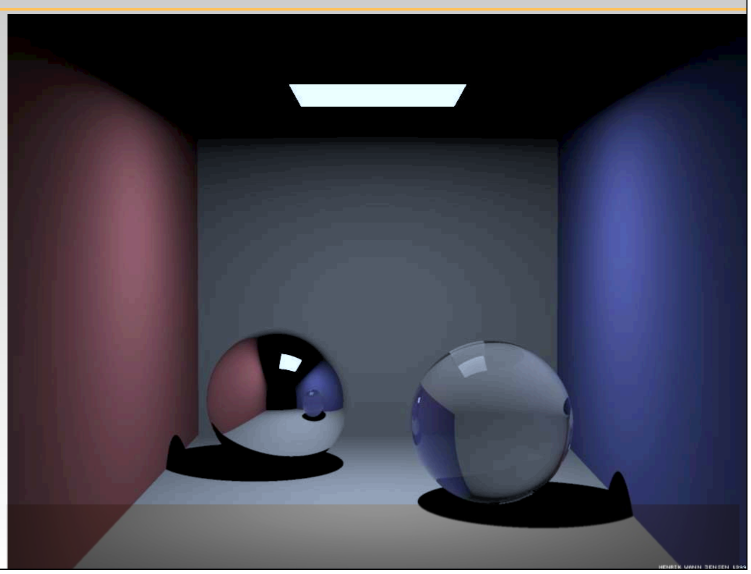


Image by
Henrik Wann Jensen

Global Illumination

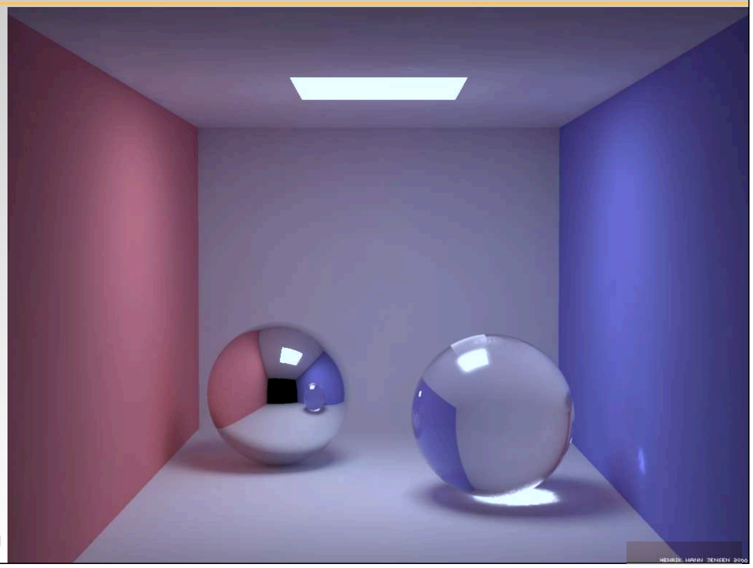


Image by
Henrik Wann Jensen

Rendering Equation

Equation guiding global illumination:

$$L_o(x,\omega_o) = L_e(x,\omega_o) + \int_\Omega \rho(x,\omega_i,\omega_0) L_i(\omega_i) d\omega_i$$

$$= L_e(x,\omega_o) + \int_\Omega \rho(x,\omega_i,\omega_0) L_o(R(x,\omega_i),-\omega_i) d\omega_i$$

Where

- ρ is the reflectance from ω_i to ω_o at point x
- L_ρ is the outgoing (I.e. reflected) **radiance** at point x in direction ω_i
  - *Radiance is a specific physical quantity describing the amount of light along a ray*
  - *Radiance is constant along a ray*
- L_e is the emitted radiance (=0 unless point x is on a light source)
- R is the “ray-tracing function”. It describes what point is

Rendering Equation

Equation guiding global illumination:

$$L_o(x,\omega_o) = L_e(x,\omega_o) + \int_\Omega \rho(x,\omega_i,\omega_0) L_i(\omega_i) d\omega_i$$

$$= L_e(x,\omega_o) + \int_\Omega \rho(x,\omega_i,\omega_0) L_o(R(x,\omega_i),-\omega_i) d\omega_i$$

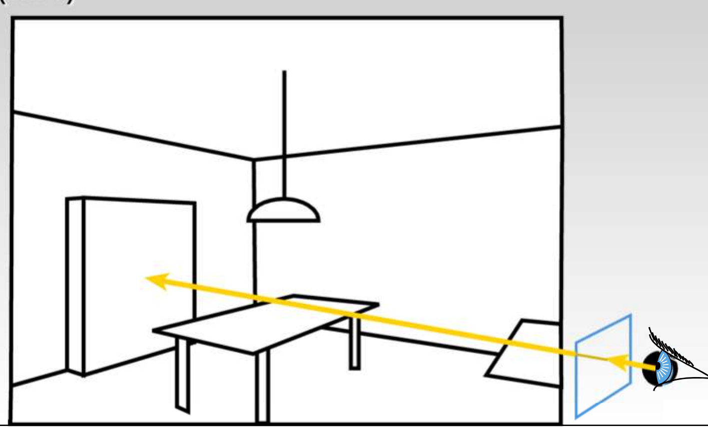
Note:

- The rendering equation is an integral equation
- This equation cannot be solved directly
  - *Ray-tracing function is complicated!*
  - *Similar to the problem we had computing illumination from area light sources!*

Wolfgang Heidrich

Ray Casting

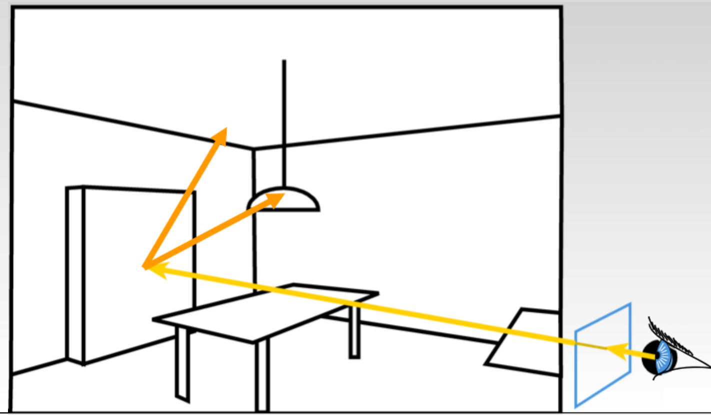
- Cast a ray from the eye through each pixel
- The following few slides are from Fred Durand (MIT)



ch

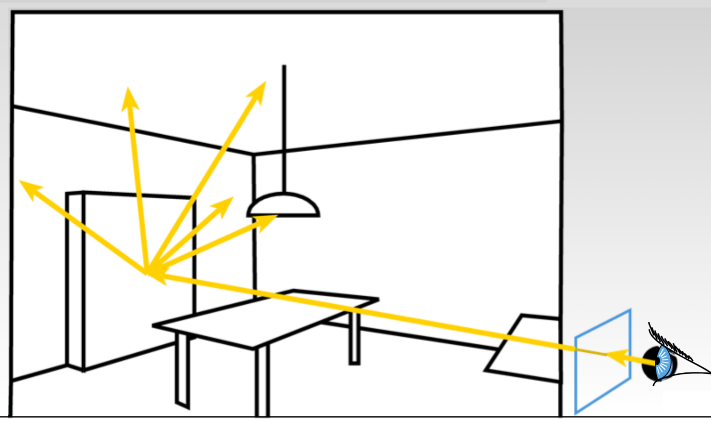
Ray Tracing

- Cast a ray from the eye through each pixel
- Trace secondary rays (light, reflection, refraction)



Monte Carlo Ray Tracing

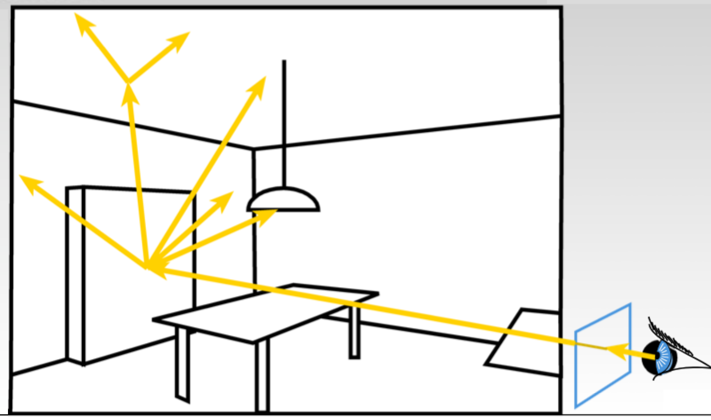
- Cast a ray from the eye through each pixel
- Cast random rays from the visible point
  - *Accumulate radiance contribution*



ch

Monte Carlo Ray Tracing

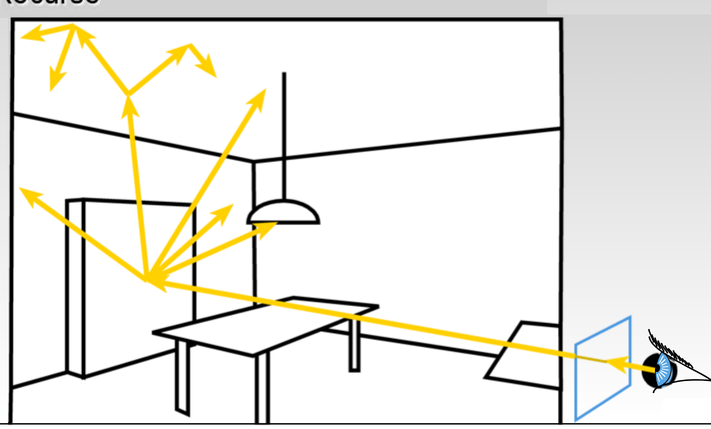
- Cast a ray from the eye through each pixel
- Cast random rays from the visible point
- Recurse



ch

Monte Carlo

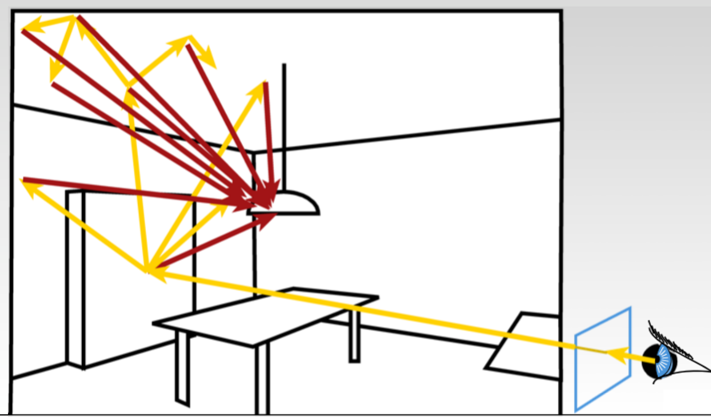
- Cast a ray from the eye through each pixel
- Cast random rays from the visible point
- Recurse



ch

Monte Carlo

- Systematically sample primary light



ch

Monte Carlo Path Tracing

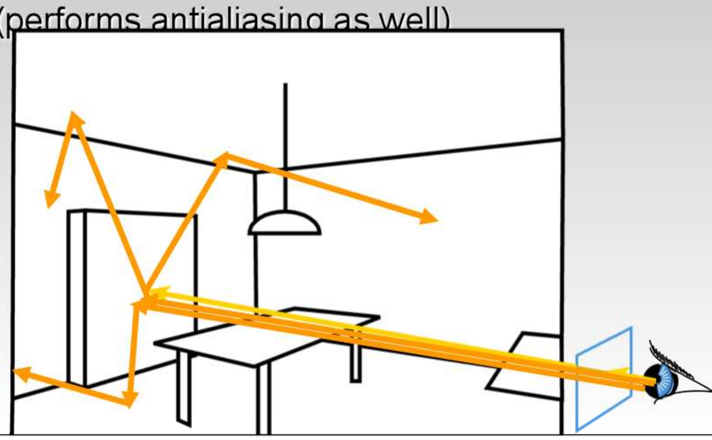
In practice:

- Do not branch at every intersection point
  - *This would have exponential complexity in the ray depth!*
- Instead:
  - *Shoot some number of primary rays through the pixel (10s-1000s, depending on scene!)*
  - *For each pixel and each intersection point, make a **single**, **random** decision in which direction to go next*

Wolfgang Heidrich

Monte Carlo Path Tracing

- Trace only one secondary ray per recursion
- But send many primary rays per pixel
- (performs antialiasing as well)



Wolfgang Heidrich

How to Sample?

Simple sampling strategy:

- At every point, choose between all possible reflection directions with equal probability
- This will produce very high variance/noise if the materials are specular or glossy
- Lots of rays are required to reduce noise!

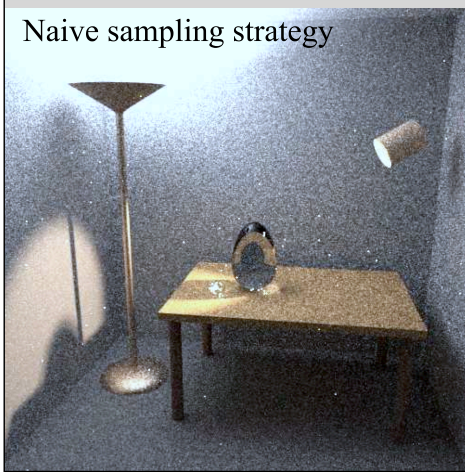
Better strategy: importance sampling

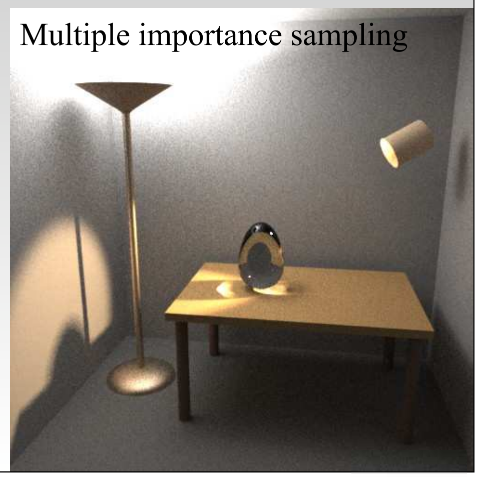
- Focus rays in areas where most of the reflected light contribution will be found
- For example: if the surface is a mirror, then only light from the mirror direction will contribute!
- Glossy materials: prefer rays near the mirror direction

Wolfgang Heidrich

How to Sample?

- Images by Veach & Guibas





How to Sample?

Sampling strategies are still an active research area!

- Recent years have seen drastic advances in performance
- Lots of excellent sampling strategies have been developed in statistics and machine learning
  - *Many are useful for graphics*

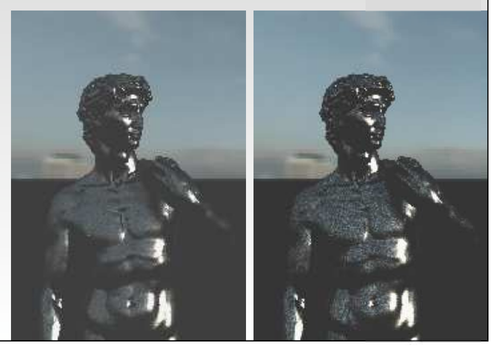
Wolfgang Heidrich

How to Sample?

Objective:

- Compute light transport in scenes using stochastic ray tracing
  - *Monte Carlo, Sequential Monte Carlo*
  - *Metropolis*

[Burke, Ghosh, Heidrich '05]
[Ghosh, Heidrich '06],
[Ghosh, Doucet, Heidrich '06]



How to Sample?

- E.g: importance sampling (left) vs. Sequential Monte Carlo (right)

Wolfgang Heidrich

More on Global Illumination

This was a (very) quick overview

- More details in CPSC 514 (Computer Graphics: Rendering)
- Not offered this year, but in 2008/9

Wolfgang Heidrich

Coming Up

Tuesday:

- Color

Thursday:

- Curves & surfaces

Wolfgang Heidrich