Other Rendering Techniques

Intro

You have seen
• Scanline converter (+z-buffer)
• Painter’s algorithm
• Radiosity
**Intro**

Some more...
- Raytracing
- Light maps
- Photon-map
- Reyes
- Shadow maps
- Shadow volumes
- PRT
- BSSRF

Raytracing then utilizes this data structure to build the image.
RAYTRACING

• Arthur Appel in 1968 Ray casting
• Turner Whitted
  ■ “An improved illumination model for shaded display” June, 1980
    Communications of the ACM.
  ■ more about illumination model

THAT PINHOLE CAMERA MODEL

After eyespace transformation
Projection Plane at z=-d
Ray Tracing

center of projection through the center of a pixel and off into space...
- What does it hit first?
- If it hits an object, compute the color for that point on the object
- That’s the pixel color

Rays

A Ray is a vector and a point
- Point – The starting point of the ray
- Vector – The ray direction
- This describes an infinite line starting at the point and going in the ray direction

Ray t values
- A distance along the ray

$\text{t}=1.5$
View Plane: Pixel Grid

Visible Surface Raytracing

// width – Screen width, height – Screen height
// fov – Field of view in degrees, aspect – Aspect ratio
double ey = tan(fov / 2. * DEGTORAD);
double ehit=2 * ey;
double ex = ey * aspect;
double ewid = 2 * ex;

for(int r=0;  r<height;  r++)
    for(int c=0;  c<width;  c++)
    {
        ray.start.Set(0,0,0);
        ray.dir.Set(-ex + ewid * (c + 0.5) / width,
                    -ey + ehit * (r + 0.5) / height,
                    -1.);
        ray.dir.Normalize();

        // Compute color in ray direction
        // Save color as image pixel
    }
WHAT WILL WE HIT?

Ray intersection tests
- Reduces to solving for $t$
  - $x = x_s + t x_d$
  - $y = y_s + t y_d$
  - $z = z_s + t z_d$
- Easiest is the sphere
  - $(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2 = r^2$
- Substitute ray equation in and solve for $t$

INTERSECTION W/ A SPHERE

$$t_{1,2} = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

$$A = x_d^2 + y_d^2 + z_d^2$$

$$B = 2(x_d(x_s - x_0) + y_d(y_s - y_0) + z_d(z_s - z_0))$$

$$C = (x_s - x_0)^2 + (y_s - y_0)^2 + (z_s - z_0)^2 - r^2$$

Questions: Why two $t$ values?
How do we tell if there is an intersection?
INTERSECTIONS W/ A PLANE

Plane equation
- $Ax + By + Cz + D = 0$
- $(A, B, C)$ is the plane normal
- We can compute the normal and, given a single point, compute $D$

$$t = \frac{Ax_s + By_s + Cz_s + D}{Ax_d + By_d + Cz_d}$$

Questions: Can $t$ be negative? Zero? Why is this “more difficult” than a sphere?

POLYGON INTERSECTION

Given a plane intersection, the question is:
- Is this point inside the polygon?
**REDUCE TO 2D**

Project the polygon for the maximum area

- What is the largest component of the normal?
- If x: Use a Y/Z projection
- If y: Use an X/Z projection
- If z: Use an X/Y projection
- Why?

**WHY “IMPractical”?**

Suppose you do a ray for every pixel and test against very polygon:

- O(WHP)
- Then if we allow recursion…
Faster Intersection Tests

Grouping objects and using bounding volumes
- Bounding boxes are easier to test for intersections.
- Your chair object could have a box around each polygon and a box around the entire object.
- We can create hierarchical levels of bounding boxes pretty easily (cluster tree)
- Spheres are a common bounding volume

Bounding Box Test

Assumes: Upright box
- Sometimes called “generalized box”

Consider box to have 6 planes
- Planes are orthogonal to some axis
- Box with range: (0.5, 0.7, -1) to (1.3, 2.2, -4)
- Ray: \( R((0,0,0),(1, 1, -1)) \)
- What is “t” for intersections with box planes orthogonal to the X axis?
Ray Walking Algorithms

- subdivide the space into boxes
- trace the ray one box after another

Not the same as line drawing:

- red = line draw
- blue = extras for ray walking
Spatial subdivisions

Uniform divisions
- All boxes are equal sized
- Advantages? Disadvantages?

Non-uniform divisions
- How?
- Why?

Octree-based
**SPEED-UP**

Item buffer
- A z-buffer which holds pointers to objects rather than colors
- How can this speed up ray tracing?
- Also called: First-hit speedup

**COLOR?**

Intersection gives a point on a polygon
Apply the color model
How do we get
- View direction?
- Light direction?
- Normal?
How do we do?

Shadows?
Reflections?

Terms to know

Shadow feeler
- Casting a ray in the direction of the light
- If intersection is less than distance to the light, light is shadowed

Self shadowing
- Roundoff errors can make us intersect ourselves
Recursion

Reflections (THE point of raytracing)
- Simply compute the color in the reflection direction

What does this do to the running time?
- Was $O(WHP)$

Adaptive Depth Control

How deep do you recurse?
- A room with mirrors?
- A room with only specular reflection?

Keep track of how much a ray will contribute to the final color
- Falls below some threshold, don’t recurse anymore…
**How do we do?**

Shadows?
Penumbra?
Depth of field?
Motion blur?
Antialiasing?

---

**Stochastic Ray Tracing**

Problem with ray tracing:
Point sampling

We can introduce randomness all over the place
- Jitter for antialiasing
- Light area for penumbra
- Depth of field and motion blur

[Jensen07]
Minimum Distance
Poisson Distribution

Better matches distribution in the eye

Algorithm
- Generate a random point
- If less than specified distance to nearest point, discard, otherwise keep

Distribution Comparisons

(a) Poisson
(b) Uniform
Path Tracing

1) Sample each pixel with N rays
2) At each intersection, trace only one ray
   proportion of rays in each category
   should match the actual (surface) distribution

Ray Tracing
Path Tracing
Reduces the amount of work deeper in the ray tree

Adaptive Sampling

Determine if number of samples is enough
■ If not, increase the number

Works for uniform and stochastic sampling
**Depth of Field**

- Finite Aperture
- Perturb the rays
- Or jitter eye position

[Jensen07]

**Motion Blur**

- Finite shutter opening time
- Perturb the rays in time

[Jensen07]
Ray Tracing from the Light Sources

In regular ray tracing we don’t see
- Lighting reflected through mirrors
- Surfaces lit by reflection from other surfaces

Large set of rays from each light
- Computes light that hits surfaces
- General solution for diffuse reflection?
- Still requires tracing from the eye as pass 2

Lightmap

- Precomputed lighting
- Implement by multitexture
- Why?
  - Texture can be re-used
  - Lighting unique but low-res

[Chambers01]
**Photon Map**

- Photon by light-based raytracing
- Separating GI from rendering
- Stored in kd-trees
- Good for
  - Caustics, subsurface scattering,
  - ...

**Photon Path**

Start from light sources

Intensity \(\rightarrow\) photon
Final Gathering

- Radiance Computation
- Distributed ray tracing

Another example
Irradiance Caching

- Similar Idea to lightmap
  - Irradiance has lower frequency (than radiance)

Reyes Rendering

Developed by Lucasfilm
In use by Pixar (Renderman)
Now combined with ray-tracing on demand
- One tile at a time
- Subdivide geometry to pixel size
**Design Goals**

Natural Coordinates
- Do calculations in coordinate system most natural for it.

Support for vectorization, parallelism, etc.
All objects work with common representations
- Micropolygons

Locality
- Geometric and Texture locality

**Design Goals (2)**

Linear in model size
Large models (larger than memory)
Textures, textures, textures
- Complex shading
- Bump and displacement maps
Reyes Algorithm

- read model
- dice
- texture → shade
- visibility
- sample
- picture

Micropolygons

Pixel → micropolygon → diced primitive in screen space

Texture map
Sampling

Shadow Maps

Render a depth image from the light position

For each pixel in final image:
1) Transform into light space
2) In shadow iff $z_{\text{inLightSpace}} > z_{\text{shadowMap}}$
**Shadow Maps**

3D texture where each point is how much illumination we receive.

**Deep Shadow Maps**

3D texture where each point is how much illumination we receive.
**Shadow Volumes**

Polygons cast polyhedra of shadow volumes
Intersect polyhedra and object space

---

**Real-time Rendering**

**Lighting**
- full env. map
- single area light
- point lights

**Transport Complexity**
- Precomputed Radiance Transfer
- [Sloan02]
- [Ng03]
- [Zhou05]
- [Ren06]
- [Sun07]

---

CSE 872 Fall 2011
Precomputed Radiance Transfer

\[ L(\vec{s}) \equiv \sum l_i B_i(\vec{s}) \]

\[ R(\vec{v}) \equiv \sum R(\vec{v}) d\vec{s} \]

Preprocess for all \( i \)
**Subsurface Scattering**

Lighting effect
- semi-translucent materials
  - skin, leaves, marble, milk, etc.

**BSSRDF**

Generalization of BRDF
- bidirectional surface scattering distribution function
  - entry and exit not identical
  - total outgoing radiance

\[ dL_o(x_o, \omega_o) = S(x_i, \omega_i; x_o, \omega_o) d\Phi_i(x_i, \omega_i) \]

\[ L_o(x_o, \omega_o) = \int_A \int_{2\pi} S(x_i, \omega_i; x_o, \omega_o) L_i(x_i, \omega_i) (\hat{n} \cdot \omega_i) d\omega_i d\Lambda(x_i) \]
**Subsurface Scattering**

**Big idea**
- break things down into single and multiple (diffuse) scattering events
- how does medium enter?

\[
\langle \vec{\omega}, \vec{V} \rangle L(x, \vec{\omega}) = \sigma_r L(x, \vec{\omega}) + \sigma_s \int_{4\pi} p(\vec{\omega}, \vec{\omega}') L(x, \vec{\omega}') d\omega' + Q(x, \vec{\omega})
\]

\[
\sigma_r = \sigma_a + \sigma_t
\]

red intensity due to scatter

first order scattering as volumetric source

\[
L_{rt}(x_i + s\vec{\omega}_i, \vec{\omega}_i) = e^{-\sigma_s} L_t(x_i, \vec{\omega}_i)
\]

\[
Q(x, \vec{\omega}) = \sigma_s \int_{4\pi} p(\vec{\omega}', \vec{\omega}) L_{rt}(x, \vec{\omega}') d\omega'
\]

---

**Diffusion of Light**

integrate transport equation over all directions

\[
\vec{V} \cdot \vec{E}(x) = -\sigma_t \phi(x) + Q_0(x)
\]

\[
\phi(x) = \int_{4\pi} L(x, \vec{\omega}) d\omega
\]

two term expansion

after substitution into transport eq.

isotropic sources

\[
L(x, \vec{\omega}) = \frac{1}{4\pi} \phi(x) + \frac{3}{4\pi} \vec{\omega} \cdot \vec{E}(x)
\]

\[
\vec{V} \phi(x) = -3\sigma_t \vec{E}(x) + \vec{Q}_1(x)
\]

\[
\vec{E}(x) = -D \vec{V} \phi(x)
\]

\[
D\vec{V}^2 \phi(x) = \sigma_t \phi(x) - Q_0(x) + 3D \vec{V} \cdot \vec{Q}_1(x)
\]

CSE 872 FALL 2011
AND THE WINNER IS?!

You may now pick up your Oscar

- Jensen, Marschner, Levoy, Hanrahan

MAKING IT FASTER