Lecture 12: Advanced Rendering

CSE 40166 Computer Graphics

Peter Bui

University of Notre Dame, IN, USA

November 30, 2010

Limitations of OpenGL Pipeline Rendering

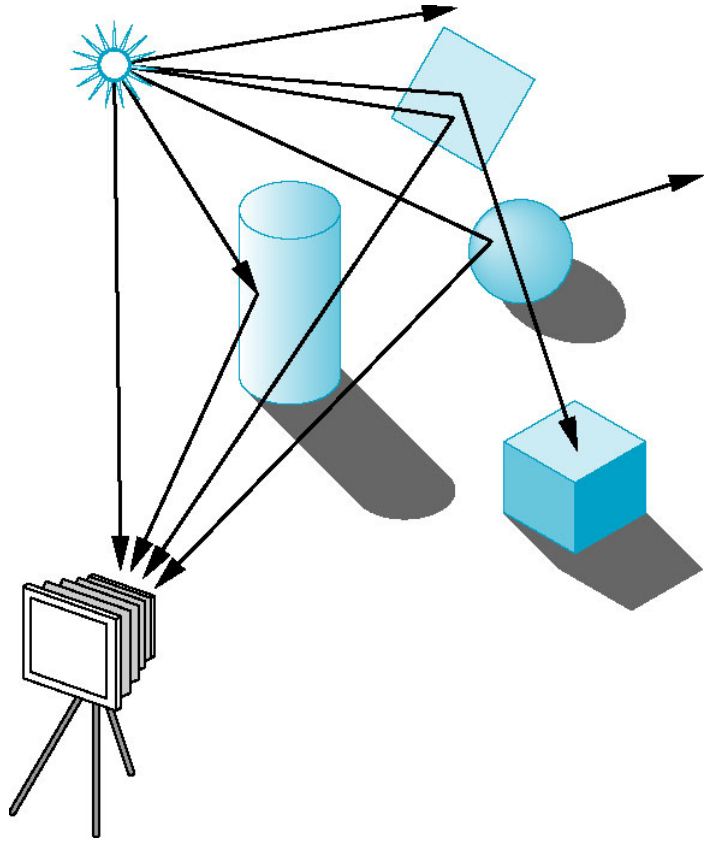
Good

- Fast, real-time graphics rendering.
- Smooth interactive graphics on computer display.

Bad

- Approximate lighting model.
- Restrictive resolution.

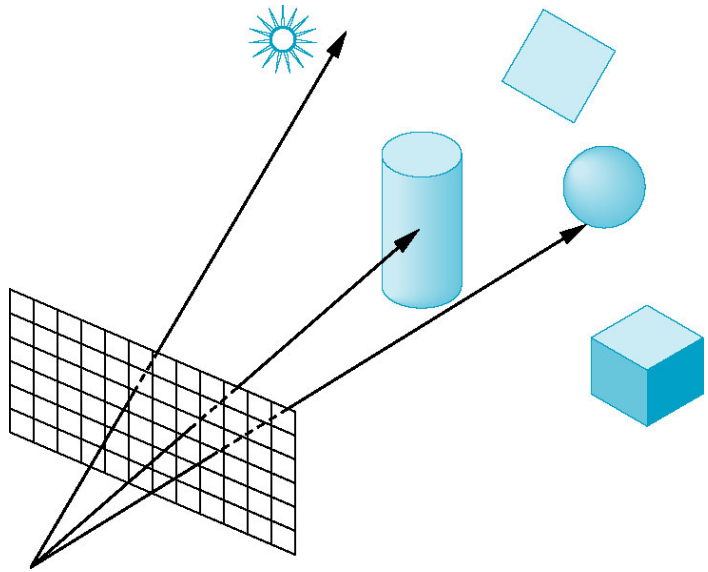
Ray Tracing



Core Concept

Light source emits rays that interact with objects, and eventually enter the synthetic camera.

Ray Tracing (Casting Model)

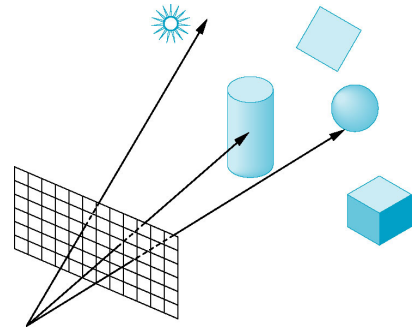


Core Concept

Reverse the direction of the rays, and only consider the rays that start at the center of projection, since we know these rays must contribute to the image.

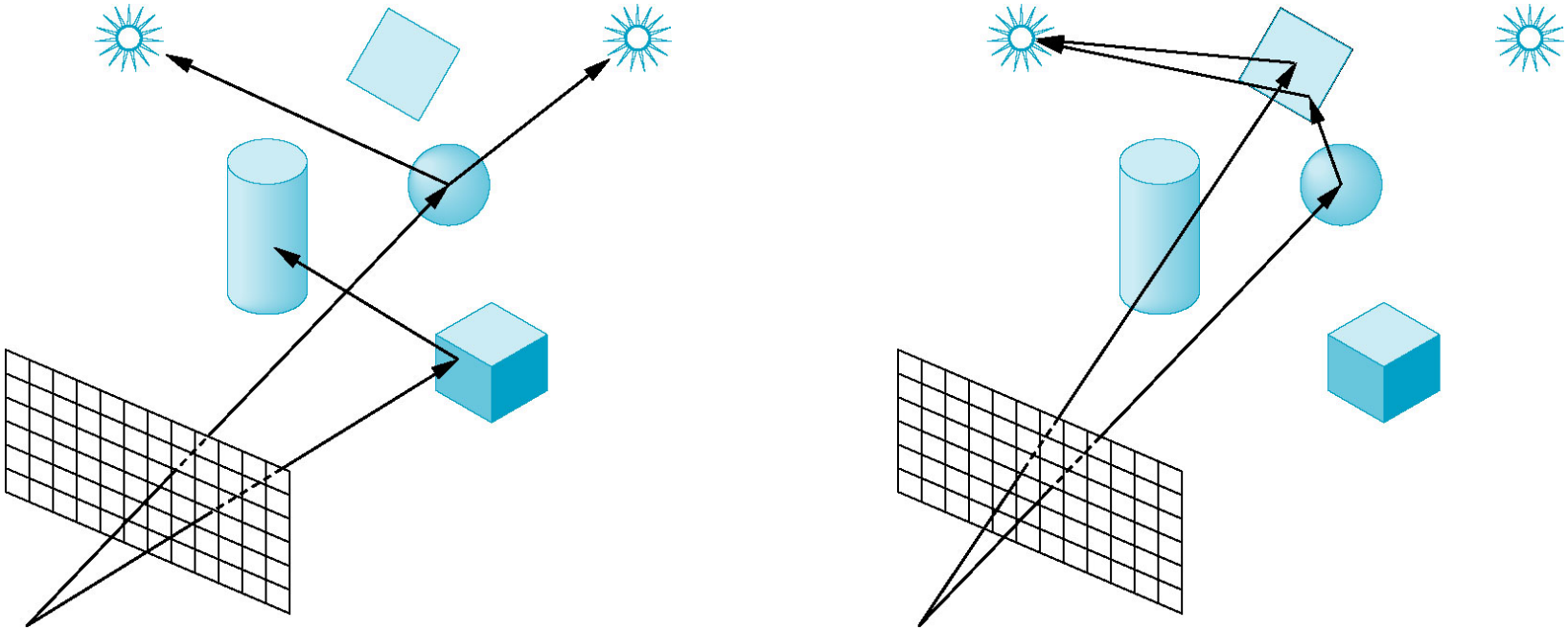
Ray Tracing (Process)

1. Start with image plane, ruled into pixel-sized areas.
2. For each pixel, cast at least one ray.
3. Each ray either intersects a surface or light source, or goes off to infinity.
4. If ray intersects, then use a lighting model to compute the intensity, otherwise, set pixel value to background color.



Ray tracer works on a pixel-by-pixel basis.

Ray Tracing (Shadows and Reflections)



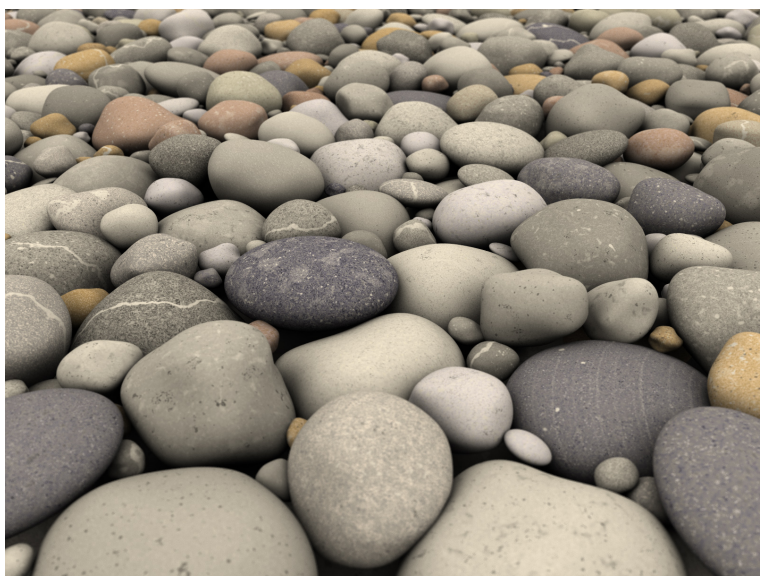
- Compute shadow or feeler rays from point on source to each source to see if surface is illuminated.
- If we have reflective surfaces, we can follow shadow rays from surface to surface by applying process recursively (each surface contributes part of the output intensity).

Ray Tracing (Pseudo-Code)

```
1  color trace (point p, vector dir, int step) {
2      color local, reflected, transmitted;
3      point q;
4      normal n;
5
6      if (step > max) return(background_color);
7
8      q = intersect(p, dir, status); // closest in scene
9
10     if (status == light_source)    return(light_source_color);
11     if (status == no_intersection) return(background_color);
12
13     n = normal(q);       // at intersection with surface
14     r = reflect(q, n);   // find reflected ray
15     t = transmit(q, n);  // find transmitted ray
16
17     local = phong(q, n, r);                  // local color
18     reflected = trace(q, r, step+1);         // color from r
19     transmitted = trace(q, t, step+1);       // color from t
20
21     return(local + reflected + transmitted); // sum colors
22 }
```

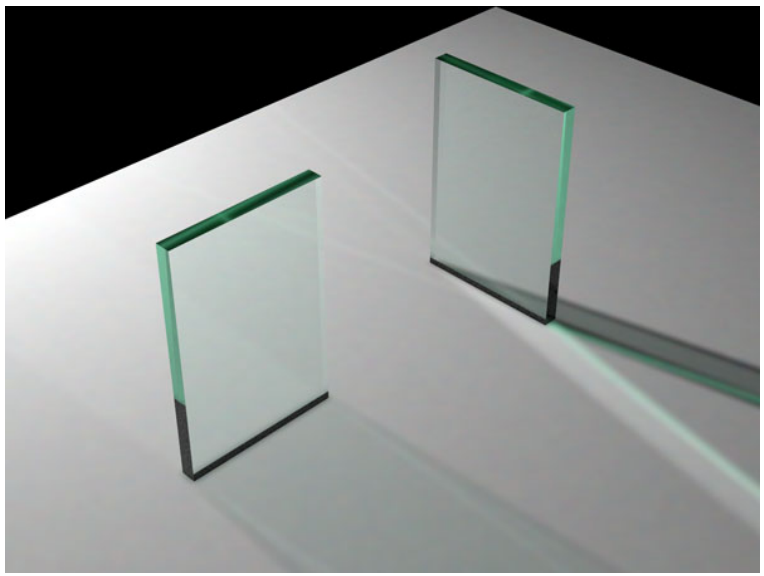
Most of the work in ray tracing goes into calculation of intersections between rays and surfaces; consequently, most basic ray tracers only support flat and quadric surfaces.

Ray Casting Model (Examples)



Ray Tracing (Odds and Ends)

- Use bounding boxes/volumes to aid in intersection.
- Ray tracing is a *sampling method*, so is subject to aliasing errors.
- Use stochastic sampling method to direct newly casted rays.
- Ray tracing is inherently parallel, but each trace requires access to every object; thus we need a shared-memory architecture.



RenderMan

Modeling-rendering paradigm

- **Modeling:**Creator manipulates objects, lights, cameras, materials interactively and stores data in file.
- **Rendering:**An offline renderer is used to produce output image (usually a render farm).

RenderMan (Features)

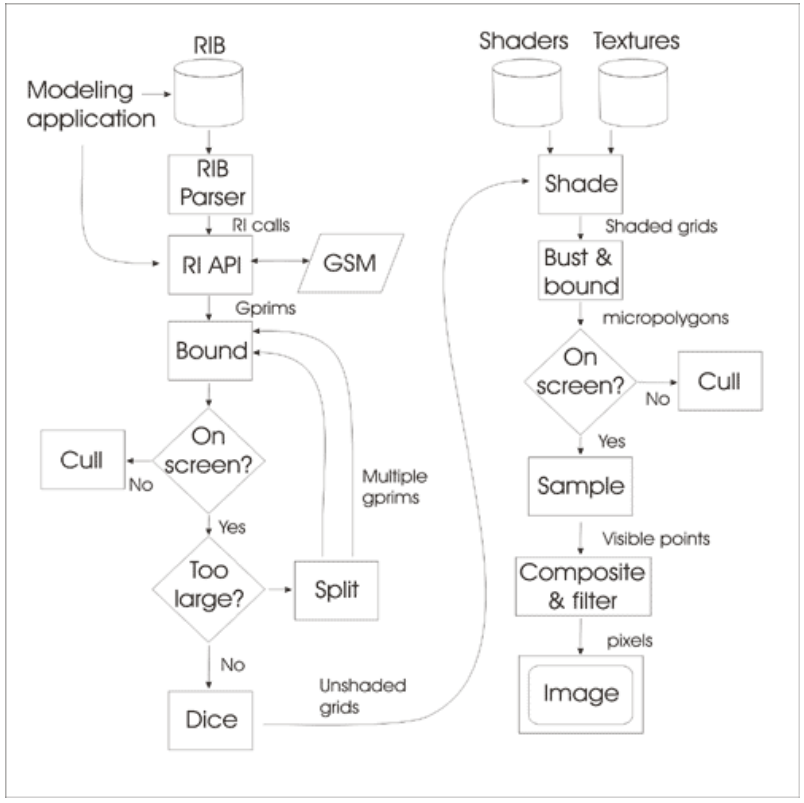
- **Particles:** Points, Spheres, Implicit surfaces.
- **Global Illumination:** Color bleeding, Ambient Occlusion, Image Based Lighting
- **Ray Tracing**
- **Point Clouds**
- **Hair and Fur**
- **Deep Shadows**
- **And much more!**

https://renderman.pixar.com/products/whats_renderman/features.html

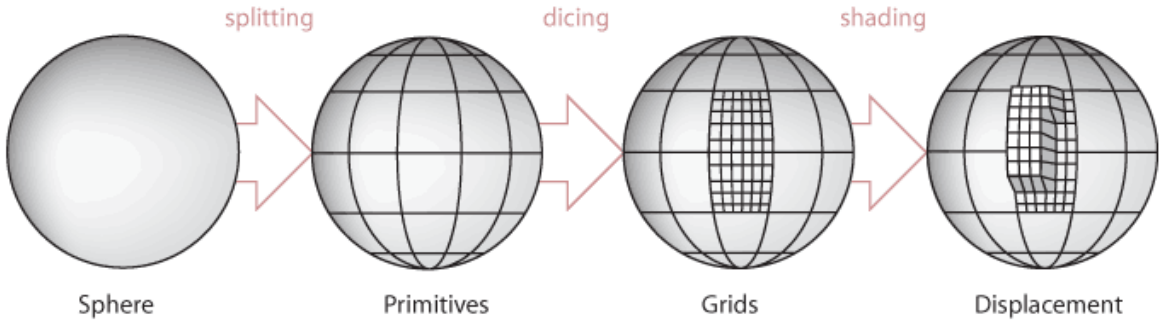
RenderMan (REYES)

RenderMan uses the REYES (*Renders Everything You Ever Saw*) algorithm to generate scene.

- REYES is a geometry-processing engine.
- Geometry can be in nurbs, polygons, or a subdivision surface.
- Eventually it will form shaded micropolygons.

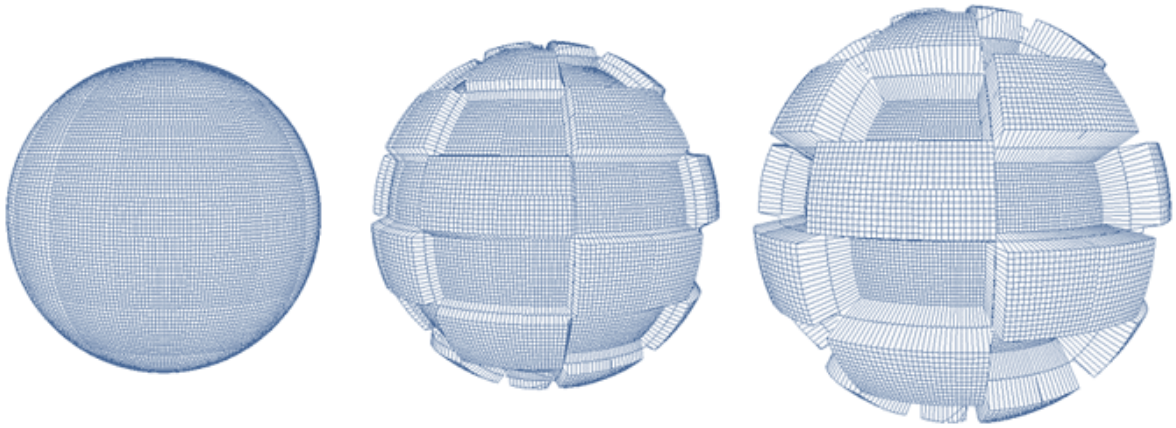


RenderMan (REYES Pipeline)



1. **Bound:** Calculate the bounding volume of each geometric primitive, and discard any objects outside viewing volume.
2. **Split:** Split large primitives into smaller, diceable primitives.
3. **Dice:** Convert primitive into grid of micropolygons, each approximately the size of a pixel.

RenderMan (REYES Pipeline continued)

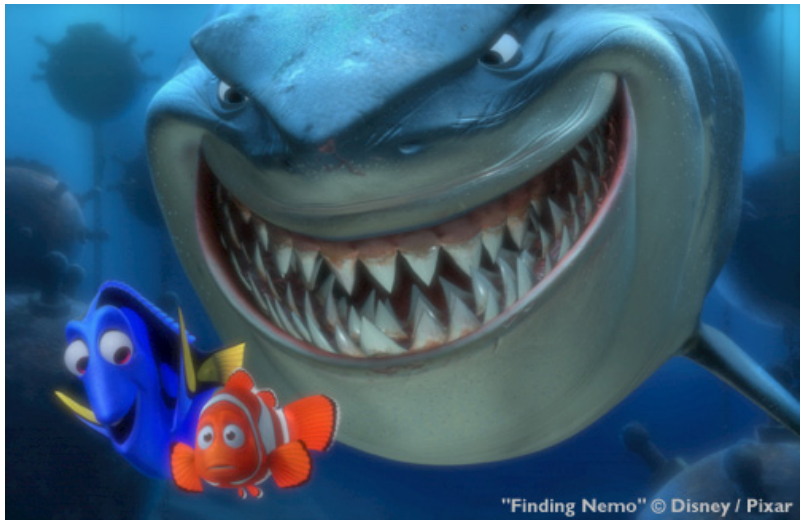


4. **Shade:** Calculate lighting and shading at each vertex of the micropolygon grid.
5. **Bust:** Bound each individual micropolygon and check for visibility.
6. **Hide:** Stocastically sample the micropolygons and produce final 2D image.

RenderMan (REYES Buckets)

A common memory optimization technique is to introduce a step called *bucketing* before the dicing step.

- Output image is divided into course grid of *buckets*.
- Objects are then split along bucket boundaries and placed into buckets based on their location.
- Each bucket is diced and drawn individually, and the data from the previous bucket is discarded before the next bucket is processed.



Thoughts

Perspective

- **Ray tracing:** 14-29 frames per second in ET: Quake Wars using 16-core system running at 2.93 GHz.
- **RenderMan:** Avatar used 40,000 processors with 104 TB RAM; each frame took hours to generate.

What is your goal?

- **Interactive:** OpenGL + Shader tricks = Good enough?
- **Photorealistic:** Ray tracing, RenderMan.