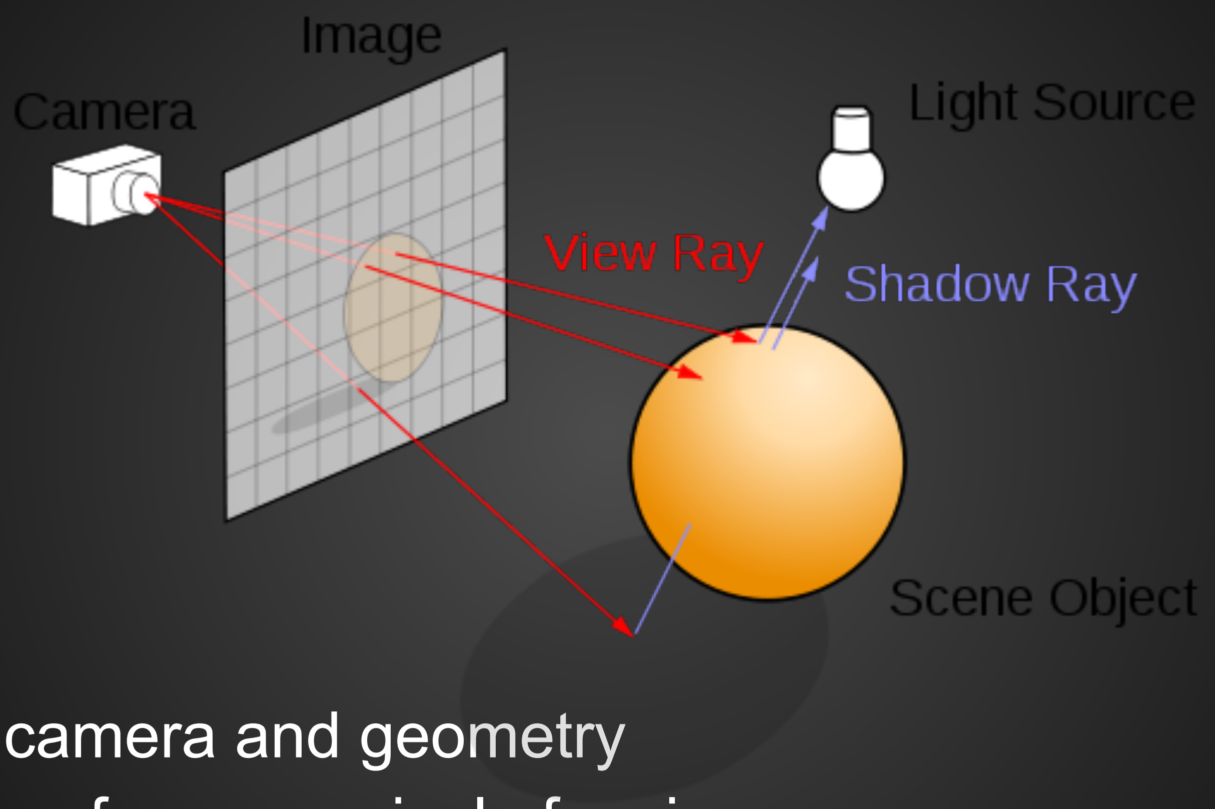
Improved Indirect Illumination

Using Irradiance Caching

Mike Buerli

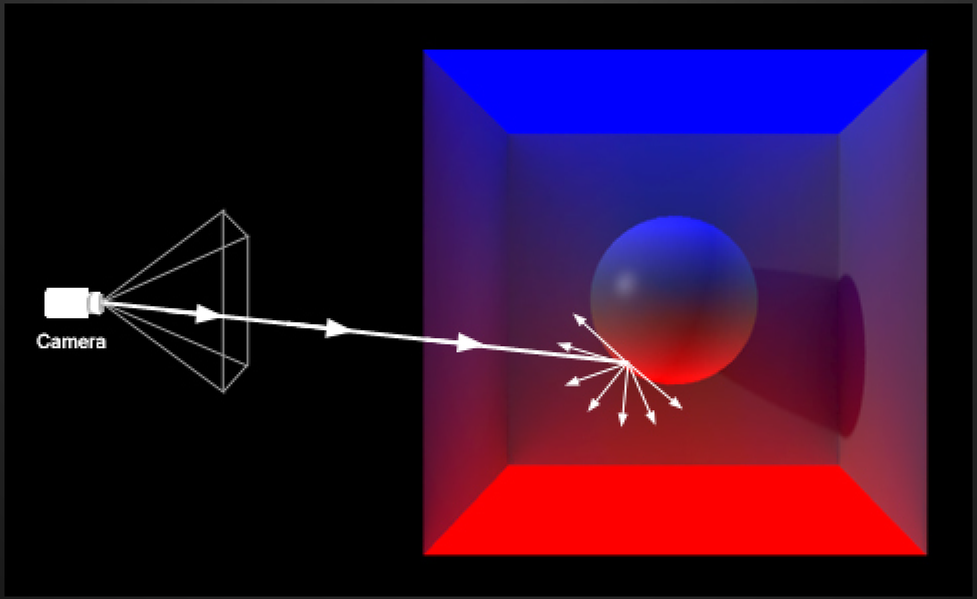
Ray tracing



- Set up camera and geometry
- Cast rays for every pixel of an image
- Cast shadow rays
- Compute lighting

Indirect Illumination

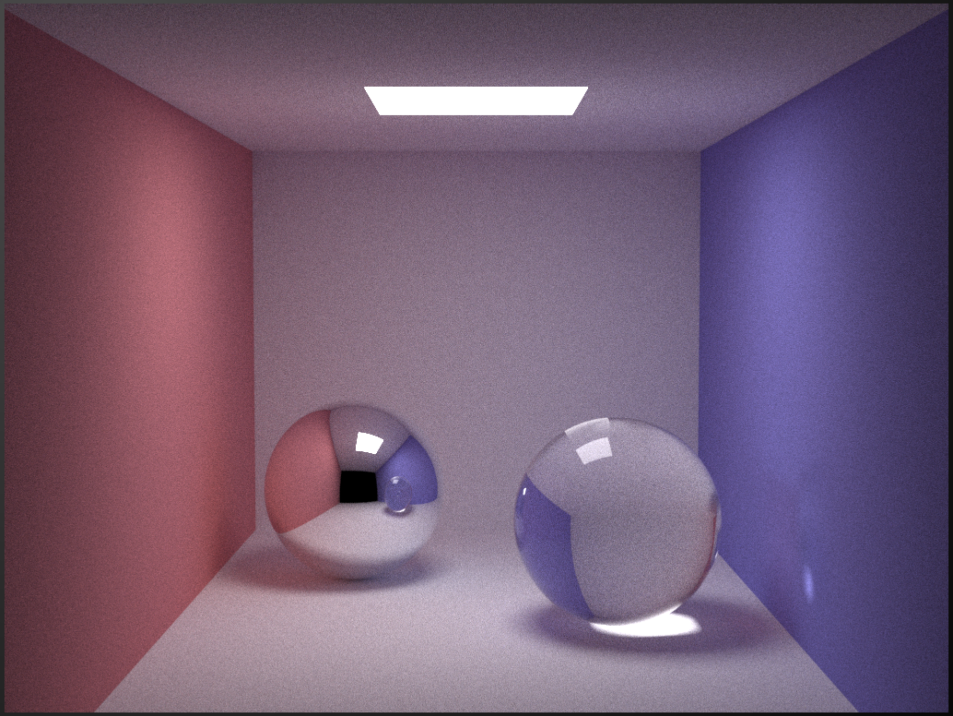
- Light is gathered from surrounding objects
- Allows for color bleeding
- Provides more realistic lighting



Monte Carlo Ray Tracing

- Method of sampling a hemisphere (sending out additional rays) at each intersection

- Includes techniques like light tracing or path tracing



Monte Carlo Ray Tracing

1. Cast rays into a scene

2. For each intersection cast out additional rays to sample a hemisphere

3. Accumulate (integrate) the light from all the samples

Monte Carlo Ray Tracing

Pros

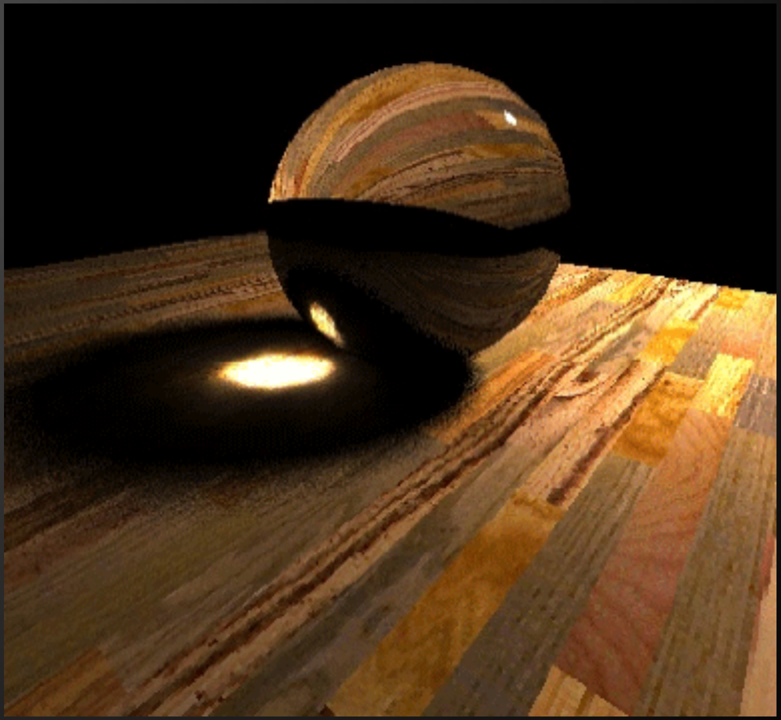
- Realistic lighting
- No setup required

Cons

- Can generate noise
- No data structure
- Number of rays cast greatly increases

Photon Mapping

- Two-pass rendering technique
- Creates light samples cast from the lights and camera
- Uses the neighboring samples to calculate lighting of an intersection



Photon Mapping

1. Create lighting samples (photons), by casting rays from light and camera

2. Cast rays from the camera

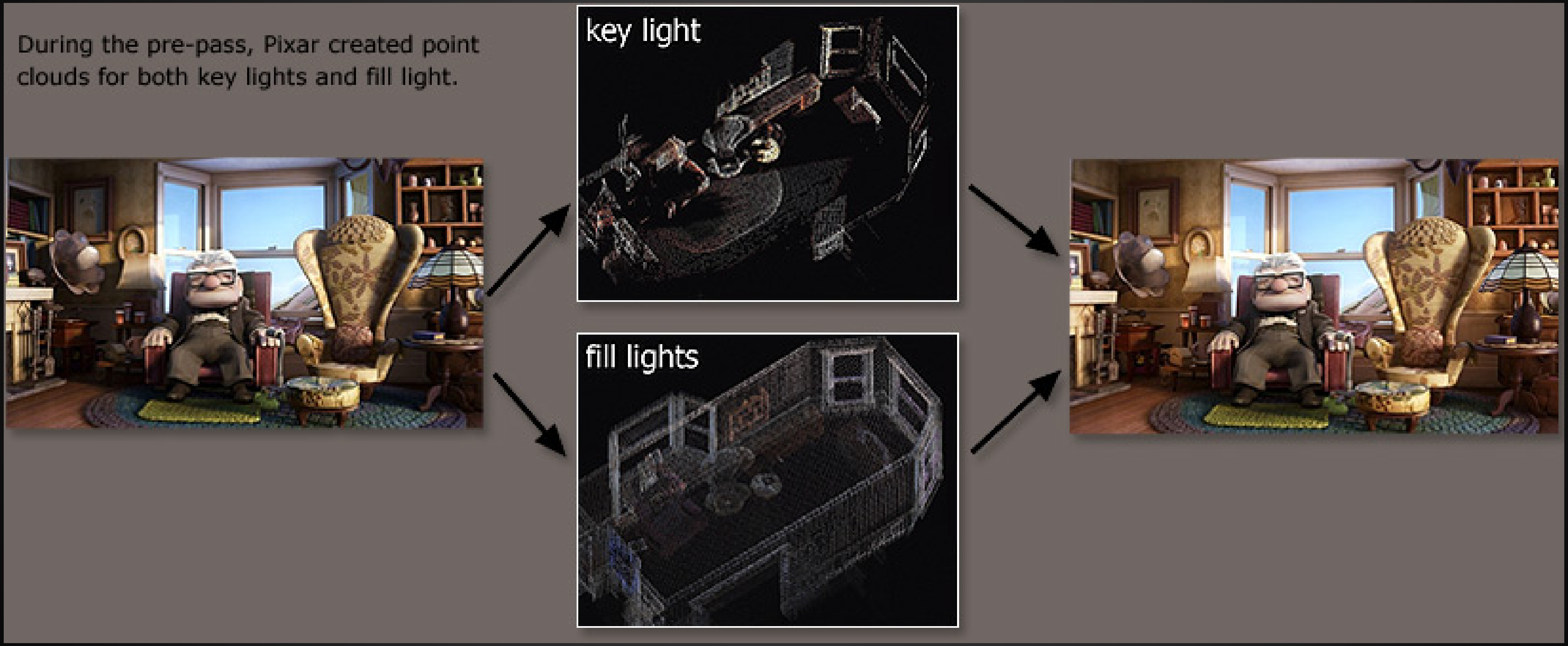
3. For each intersect, use the neighboring photons to calculate the lighting

Photon Mapping

Pros
- Realistic lighting, including caustics
- Uses a data structure to store 'photons'

Cons
- Requires two passes
- Can generate noise

Point Based Color Bleeding


During the pre-pass, Pixar created point clouds for both key lights and fill light.

Point Based Color Bleeding

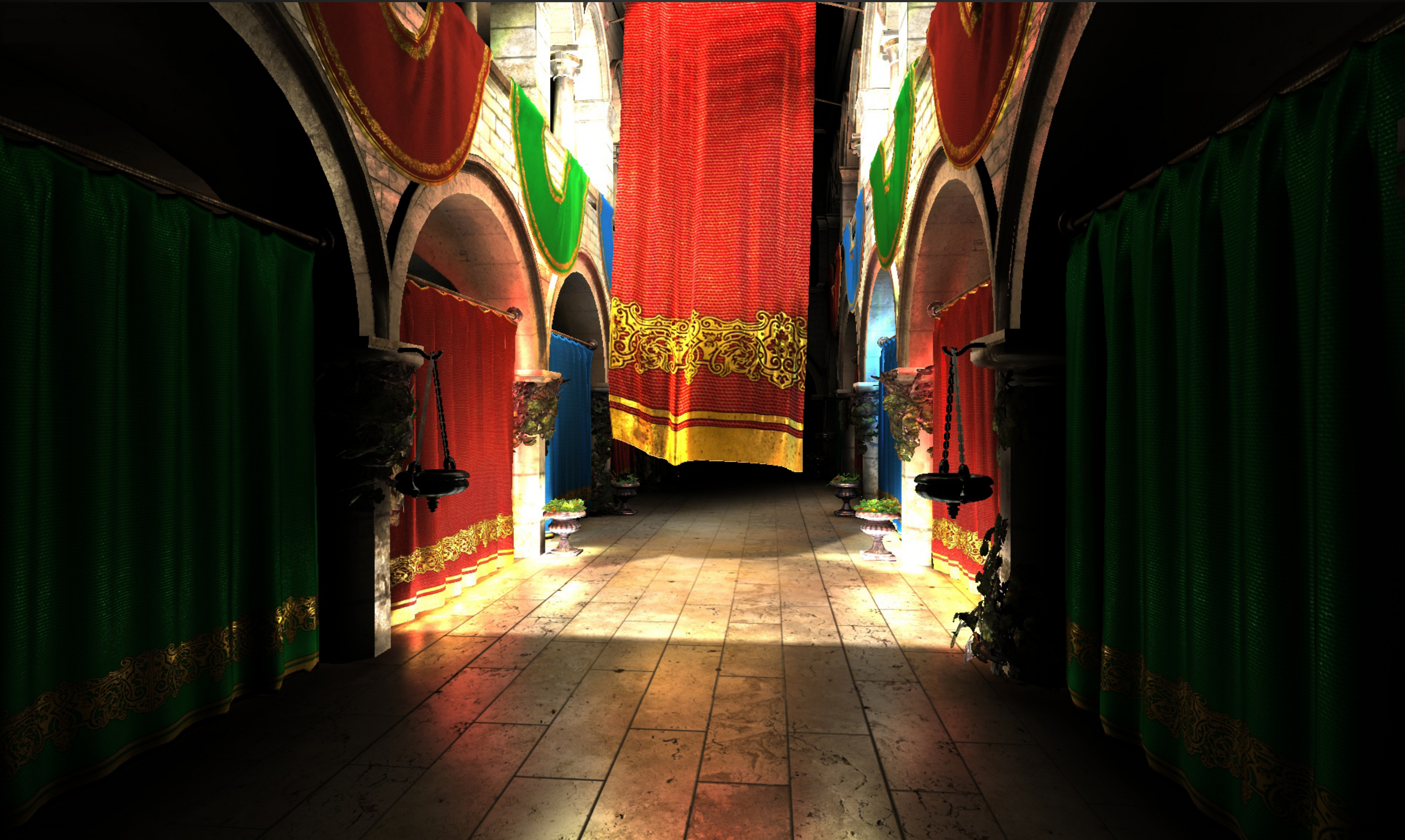
1. Create point cloud of direct illumination
2. Cast rays into the scene
3. Gather cubes of indirect illumination for each intersection from the point cloud
4. Compute the lighting

Point Based Color Bleeding

Pros
- Good approximation of indirect illumination
- Use of point clouds
- Much faster that previous techniques

Cons
- Requires multiple passes

Interactive Indirect Illumination

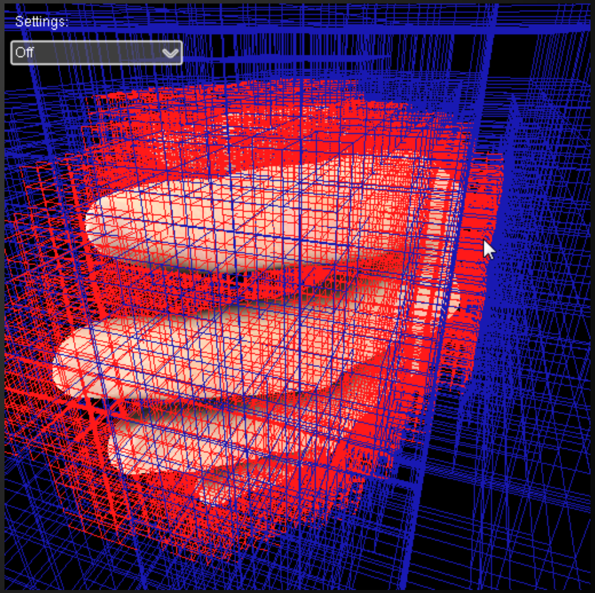


Interactive Indirect Illumination

1. Stores incoming radiance in a sparse voxel octree

2. Filter the higher levels of the octree (mipmap)

3. Render the scene, using voxel cone tracing to gather indirect illumination from the octree.



Interactive Indirect Illumination

Pros:
- Interactive frame rates (25 - 75 fps)
- Use of data structures (octree) to manage light
- Mipmaps to create higher level estimates

Cons:
- Not as accurate
- Uses rasterization for the scene and just overlays lighting

My Approach

- Apply interactive techniques to ray tracing
- Support complex scenes with large amounts of geometry
- Be able to trade off accuracy for render time

My Approach

1. Generate sparse voxel octree of direct illumination

2. Create approximations of homogeneous regions in parent nodes

3. Cast rays into scene

4. Compute lighting, adding in direct illumination from neighboring octree nodes (# dependent upon the LOD)

Validation Framework

- Production level ray tracing software is not freely available
- Production level hardware is also not available

Framework

- Create naive implementations of leading algorithms
- Compare the runtimes of my approach to that of the other algorithms
- Analyze the performance of a variety of scenes both complex and simple

Questions?