CS559: Computer Graphics

Lecture 28: Ray Tracing

Li Zhang
Spring 2010

Slides are from Ravi Ramamoorthi's graphics class at Columbia U

Effects needed for Realism

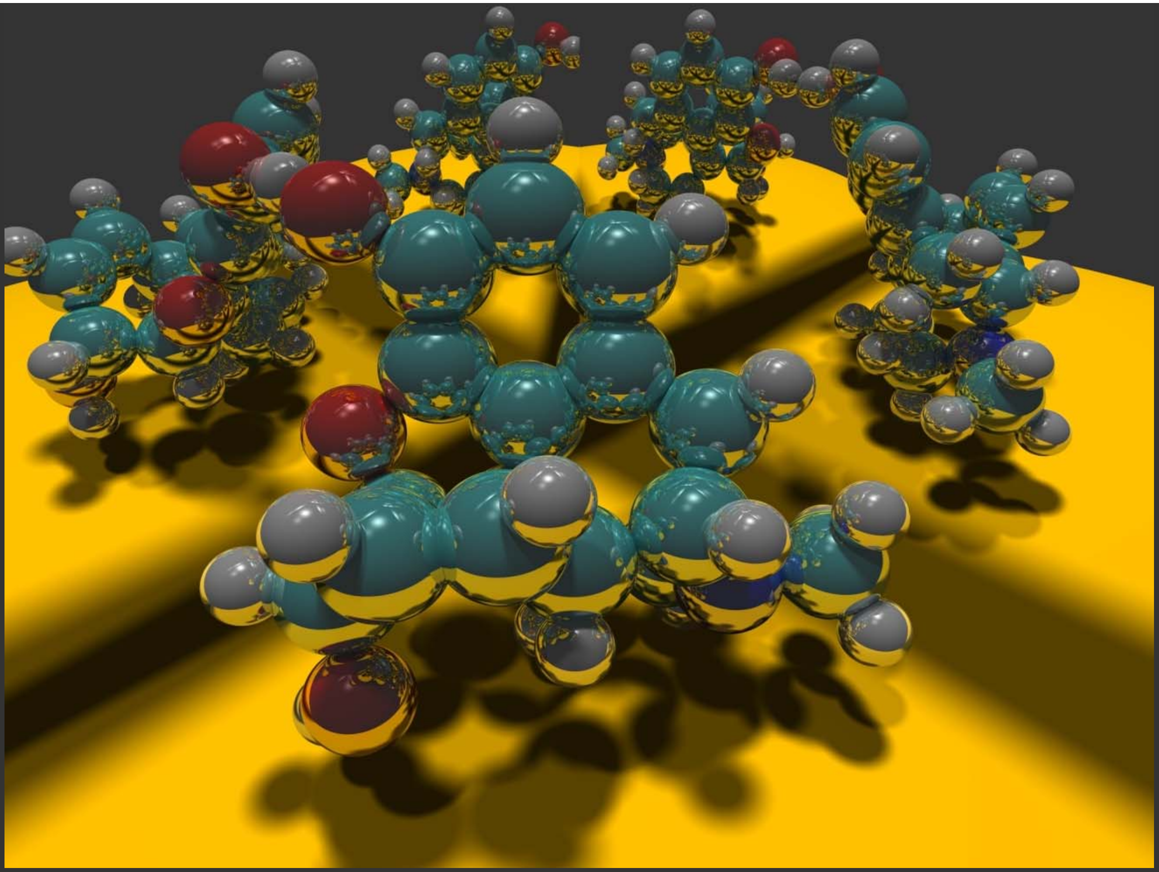


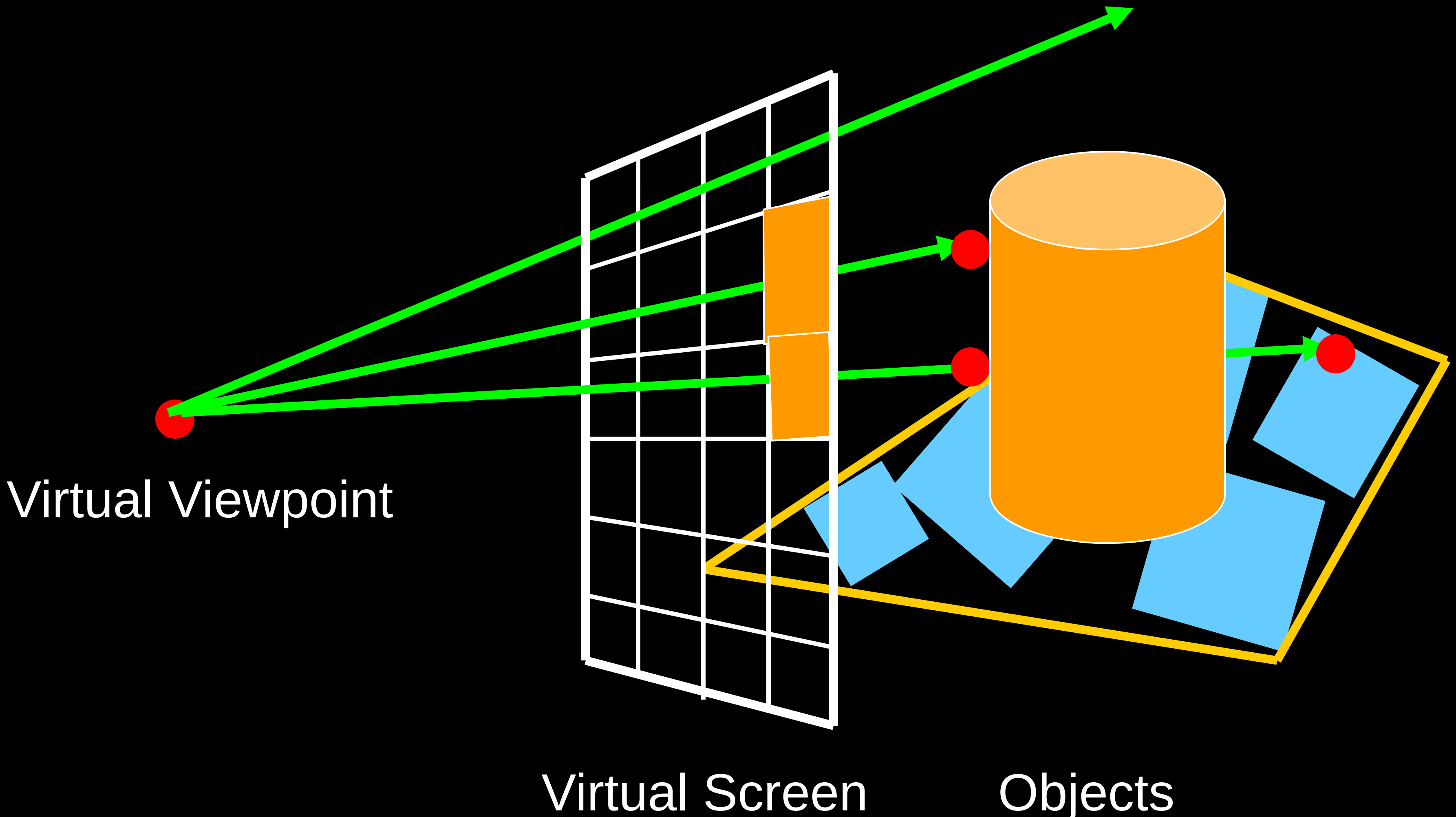
Image courtesy
Paul Heckbert 1983

- Reflections (Mirrors and Glossy)
- Transparency (Water, Glass)
- Interreflections (Color Bleeding)
- (Soft) Shadows
- Complex Illumination (Natural, Area Light)
- Realistic Materials (Velvet, Paints, Glass)
- And many more

Ray Tracing

- Different Approach to Image Synthesis as compared to Hardware pipeline (OpenGL)
  - OpenGL : Object by Object
  - Ray Tracing : Pixel by Pixel
- Advantage:
  - Easy to compute shadows/transparency/etc
- Disadvantage:
  - Slow (in early days)

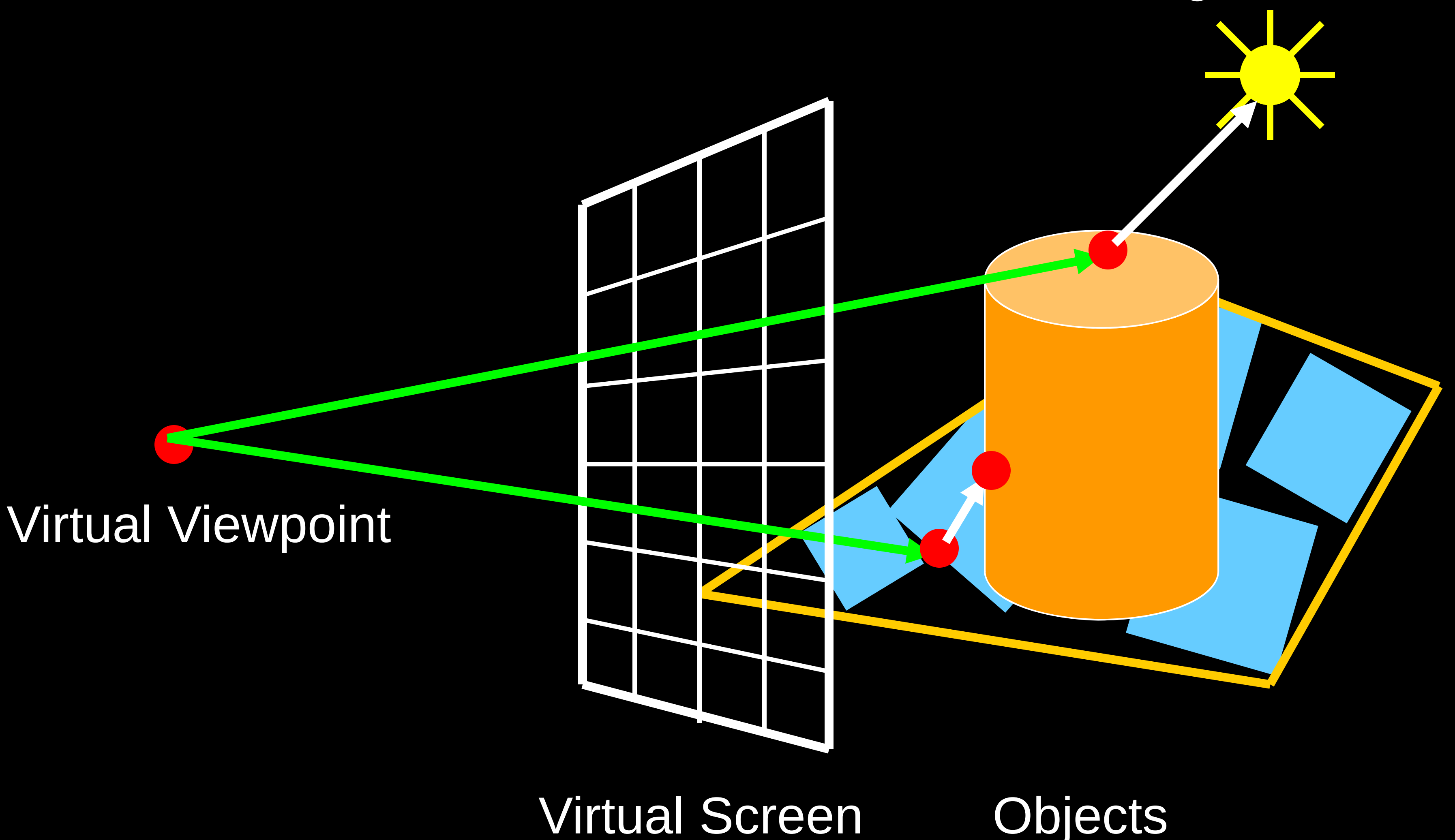
Basic Version: Ray Casting



Ray intersects object: shade using color, lights, materials

Multiple intersections: Use closest one (as does OpenGL)

Shadows

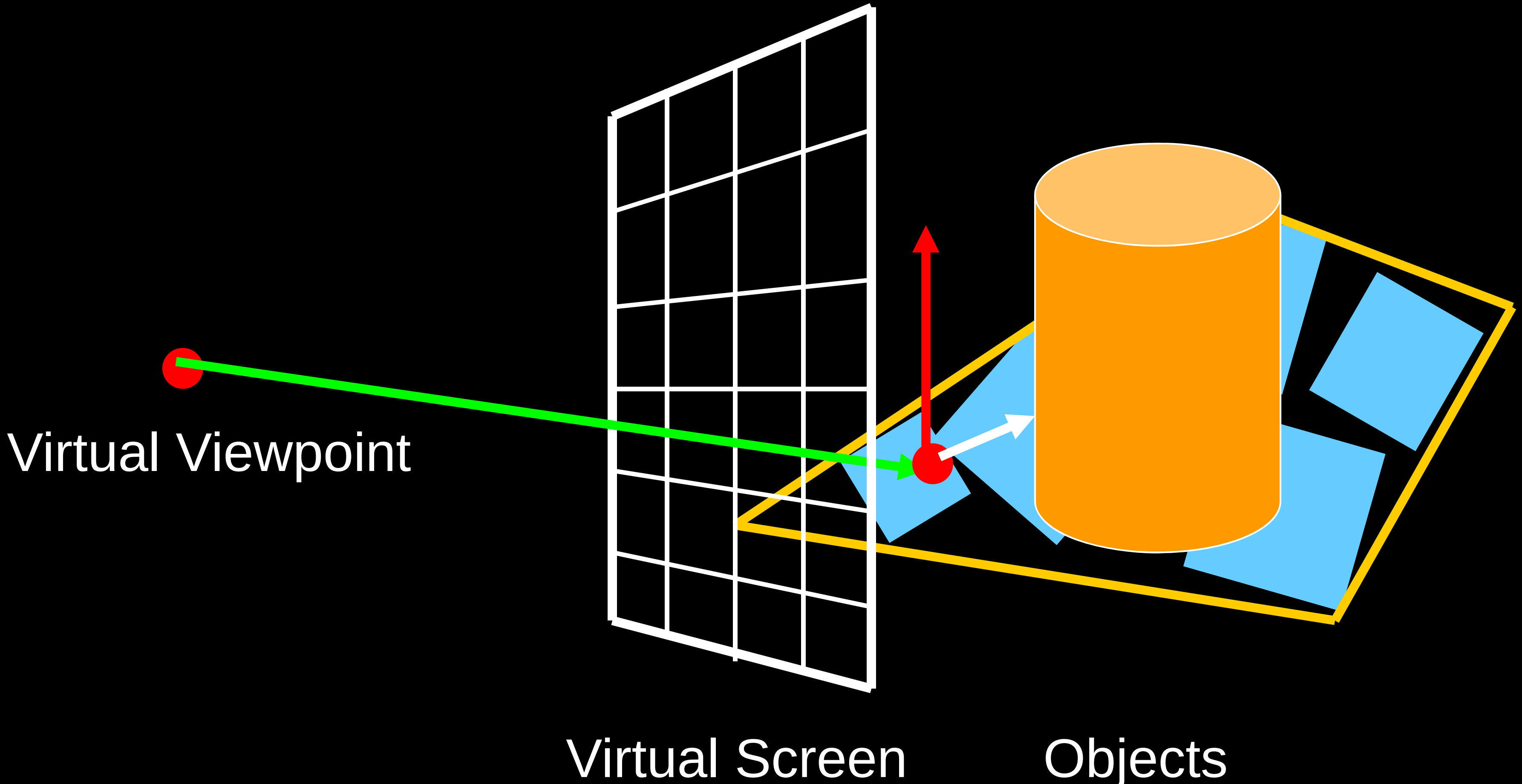


Shadow ray to light is unblocked: object visible

Shadow ray to light is blocked: object in shadow

10.5 in textbook

Mirror Reflections/Refractions



Generate reflected ray in mirror direction,
Get reflections and refractions of objects

Recursive Ray Tracing (Core Idea)

For each pixel

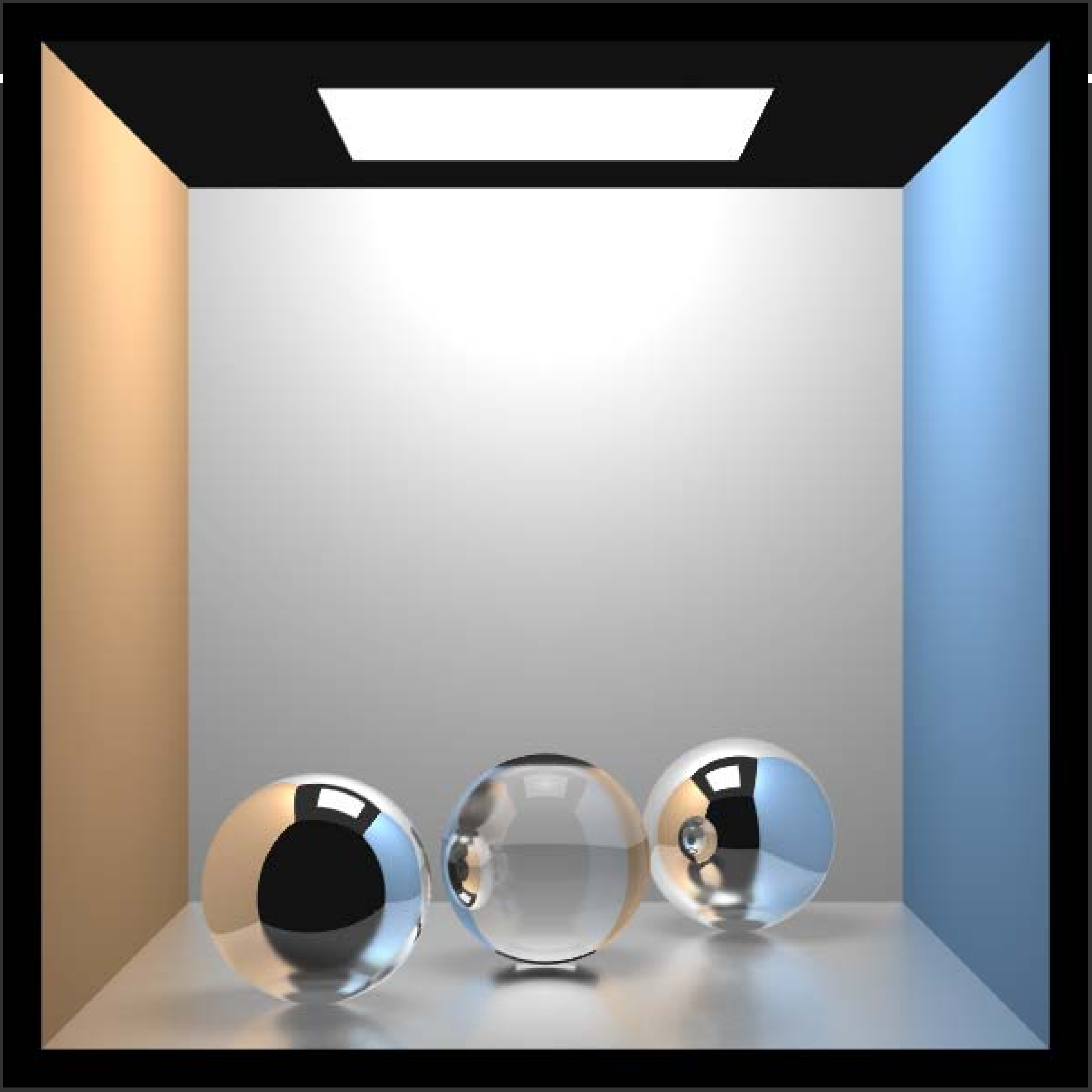
- Trace Primary Eye Ray, find intersection

- Trace Secondary Shadow Ray(s) to all light(s)
  - Color = Visible1 ? Illumination Model(light1) : 0 ;
  - Color += Visible2 ? Illumination Model(light2) : 0 ;
  - …

Recursive function Calls

- Trace Reflected Ray
  - Color += reflectivity * Color of reflected ray

- Trace Refracted Ray
  - Color += transparency * Color of refracted ray



Example

- Sphere
  - How to decide there is an intersection?
- Triangle
  - How to decide the intersection is inside?
- Polygon
  - How to decide the intersection is inside?

- How about an ellipsoid?

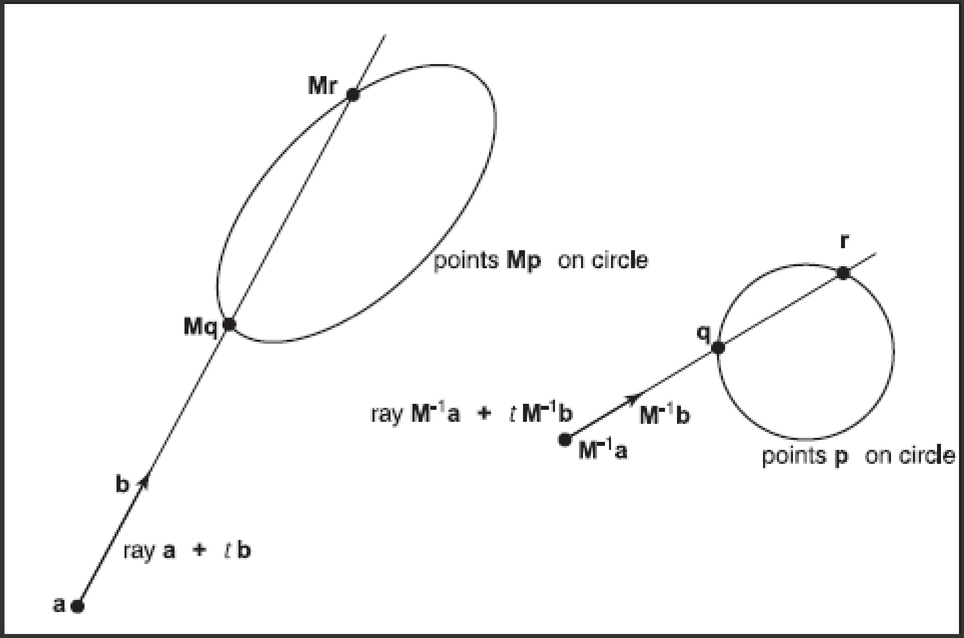
Ray-Tracing Transformed Objects

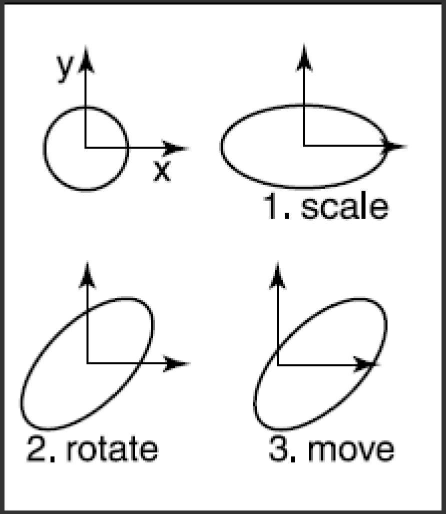
We have an optimized ray-sphere test

- But we want to ray trace an ellipsoid…

Solution: Ellipsoid transforms sphere

- Apply inverse transform to ray, use ray-sphere





Acceleration

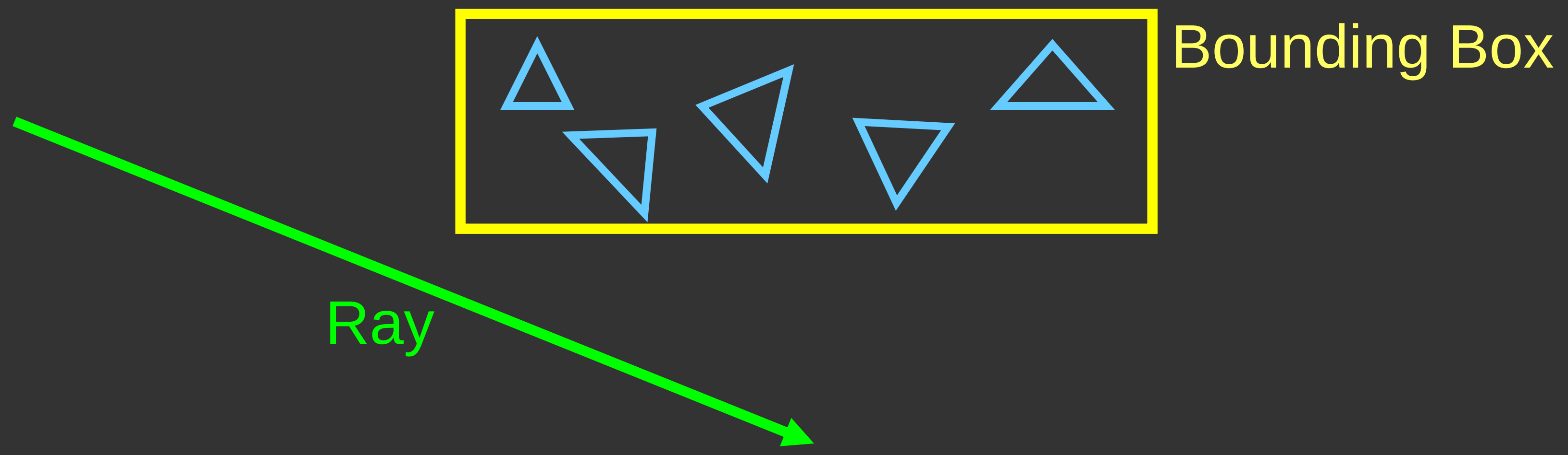
Testing each object for each ray is slow

- Faster Intersections
  - Optimized Ray-Object Intersections
  - ***Fewer Intersections***

Acceleration Structures

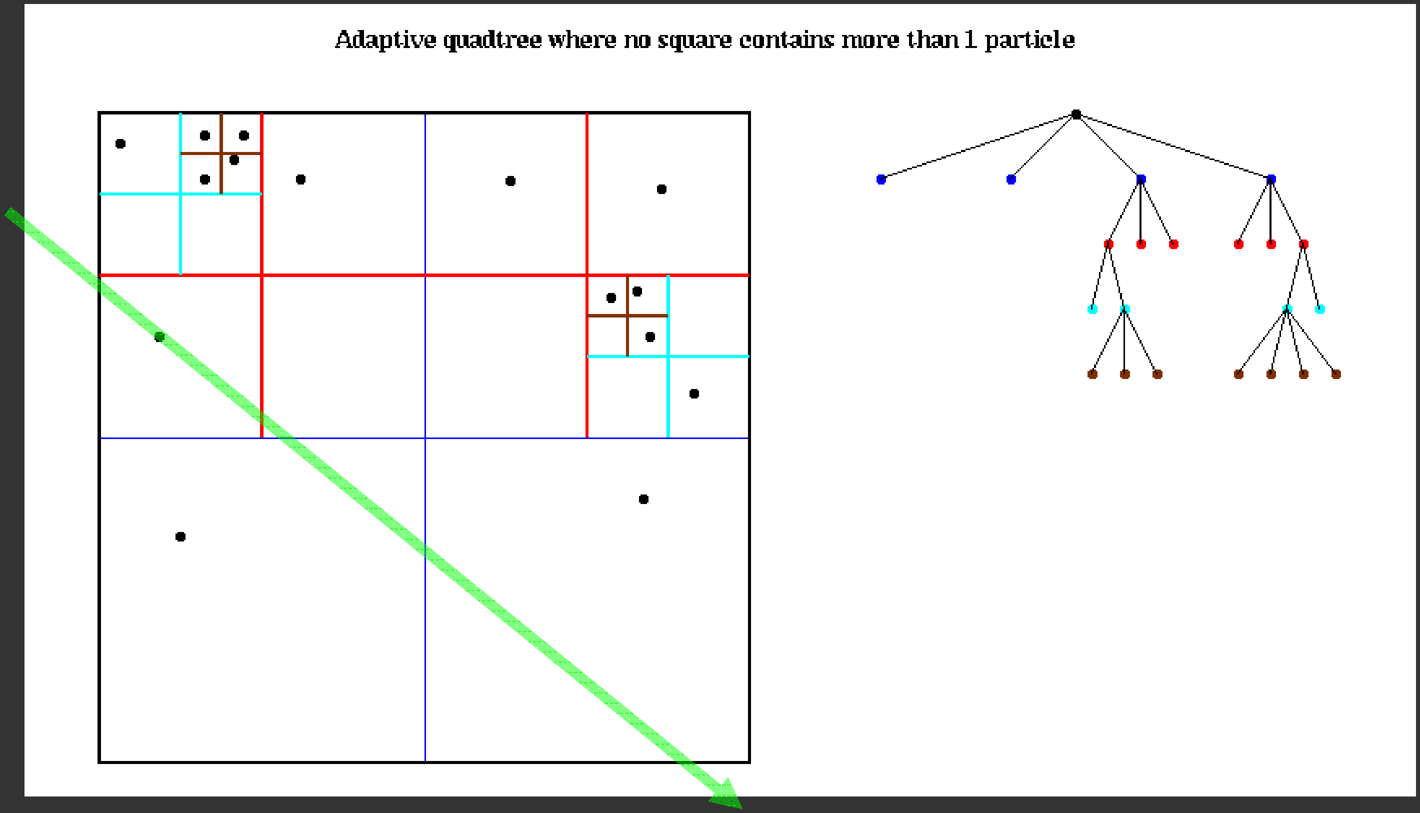
Bounding boxes (possibly hierarchical)

If no intersection bounding box, needn't check objects

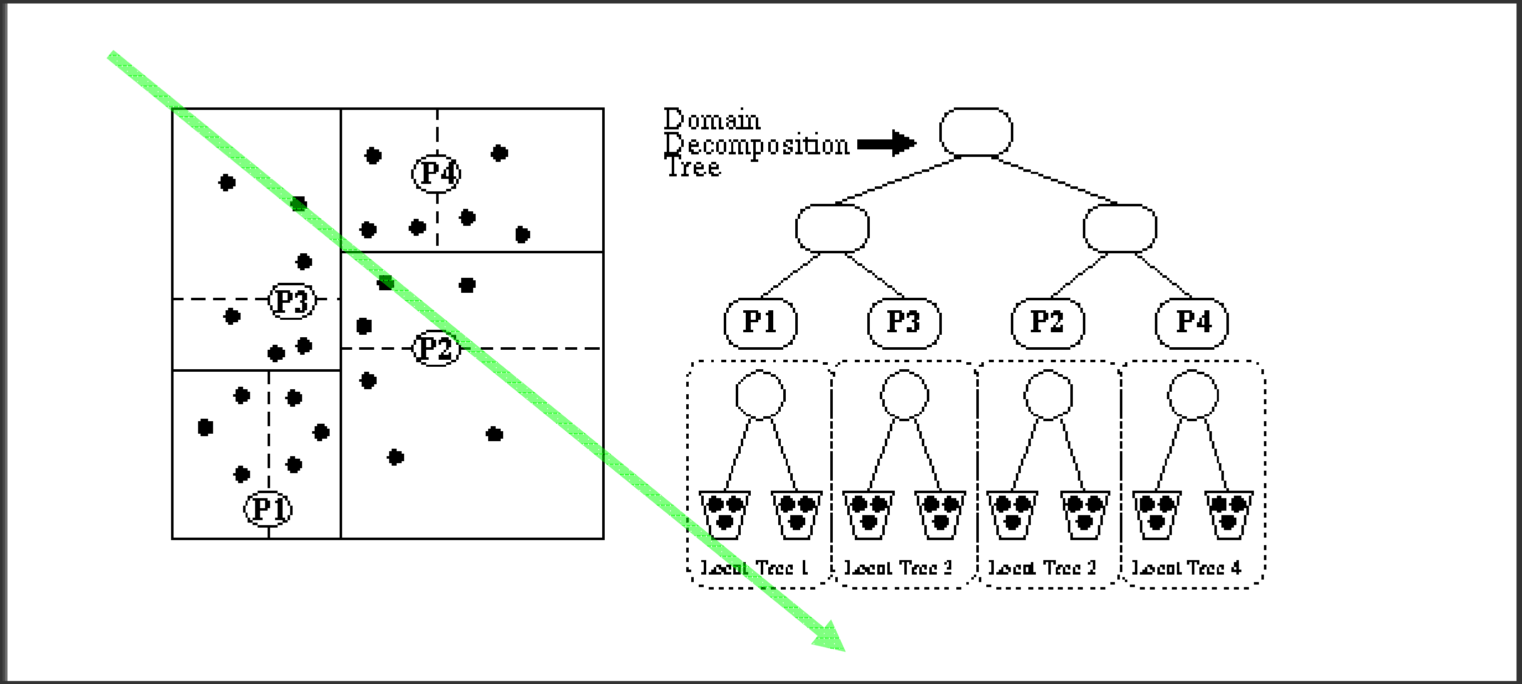


Different Spatial Hierarchies (Oct-trees, kd trees, BSP trees)

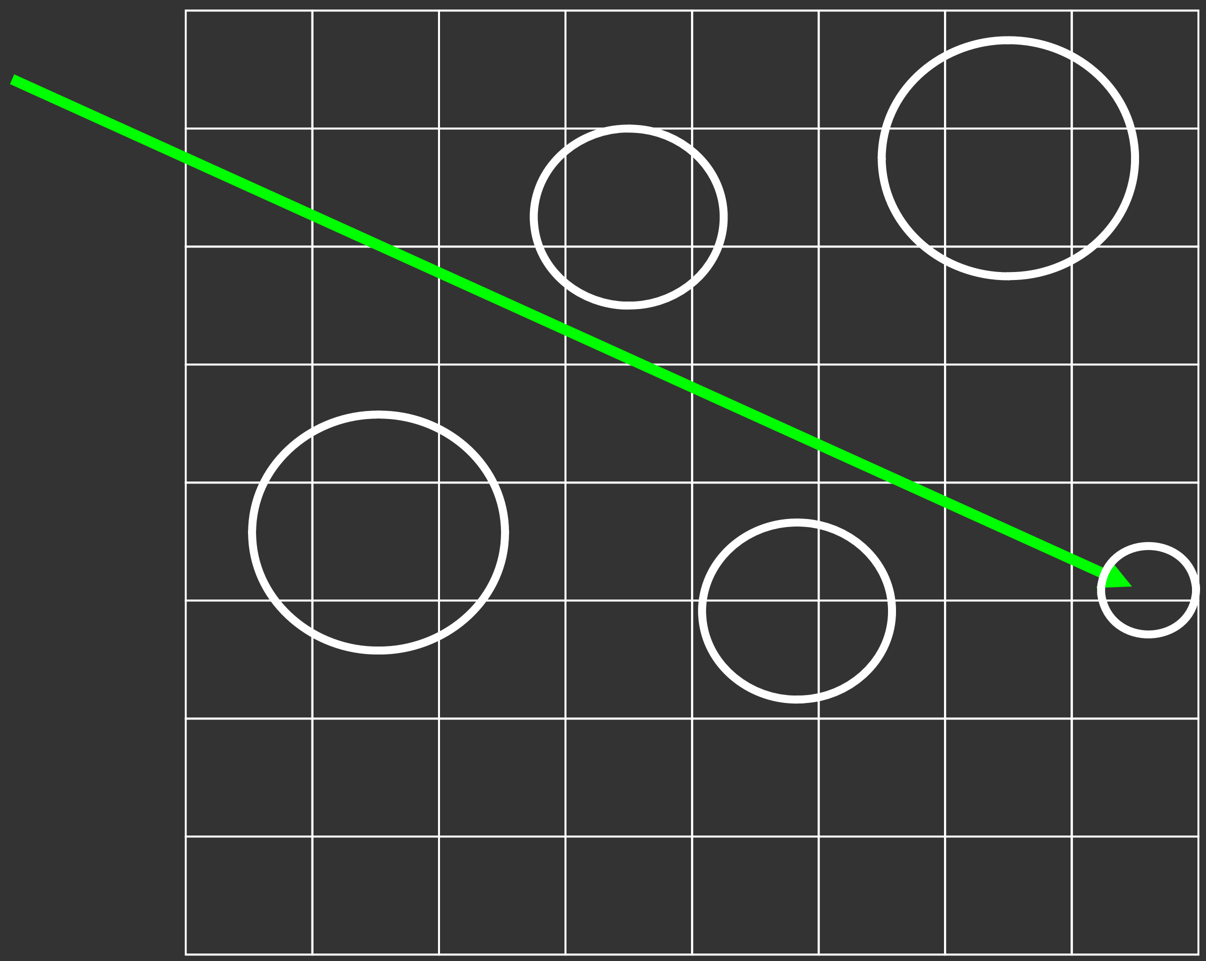
Octree


Adaptive quadtree where no square contains more than 1 particle

K-d tree

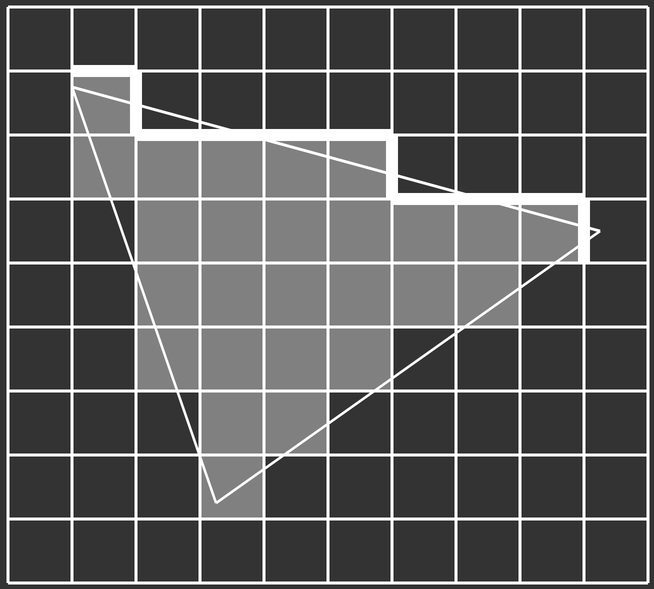

P4
P3
P2
P1
Domain
Decomposition
Tree
P1
P3
P2
P4
Local Tree 1
Local Tree 2
Local Tree 3
Local Tree 4

Acceleration Structures: Grids



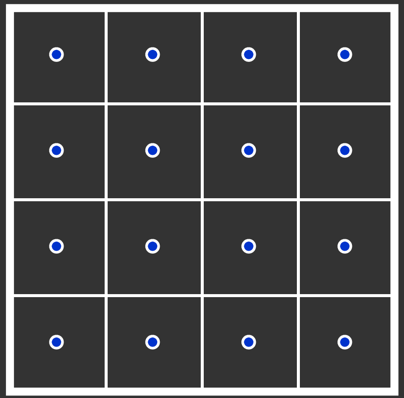
Anti-aliasing

- Aliasing when drawing a diagonal on a square grid:
  - *stairstepping*
  - AKA *jaggies*
- Especially noticeable:
  - high-contrast edges
  - near horizontal or near vertical
  - As line rotates (in 2D)
    - steps change length
    - corners of steps slide along the edge
    - known as *crawlies*



Supersampling

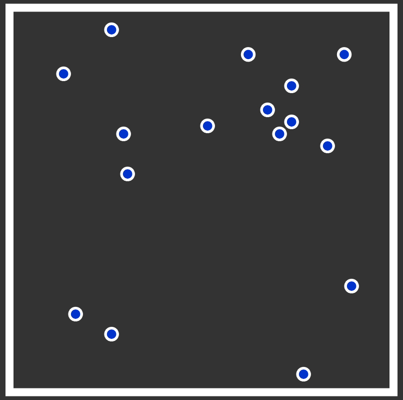
- A more popular method (although less elegant) is *supersampling:*
  - Point sample the pixel at several locations
  - Combine the results into the final pixel color
- By sampling more times per pixel:
  - Raises the sampling rate
  - Raises the frequencies we can capture
- Commonly use 16 or more samples per pixel
  - Requires potentially 16 times as much work to generate image
  - 16 times Memory?
- A brute-force approach
  - But straightforward to implement
  - Very powerful



Moiré Artifact

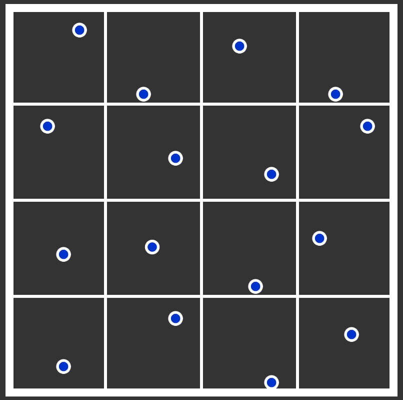
Random Sampling

- Supersample at several randomly located points
- Breaks up repeating signals
  - Eliminates Moiré patterns
  - Instead of aliasing, frequencies greater than 1 pixel appear as *noise* in the image
- Noise tends to be less objectionable to the viewer than jaggies or Moiré patterns
  - The human eye is pretty good at filtering out noise
- But suffers from potential clustering and gaps
  - Result is not necessarily accurate
  - Too much noise.

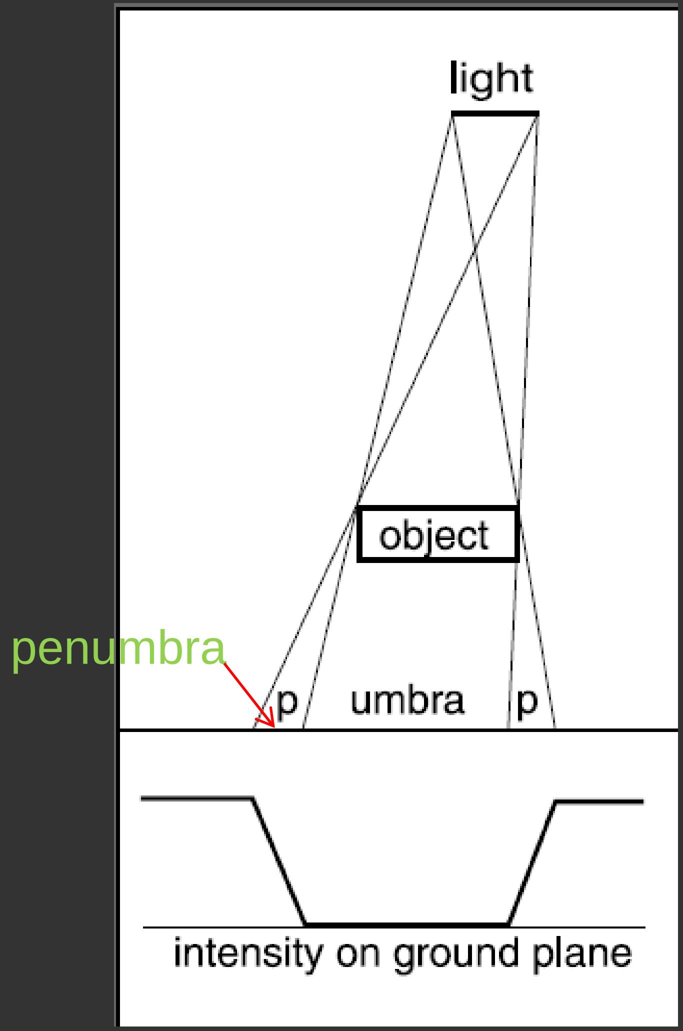


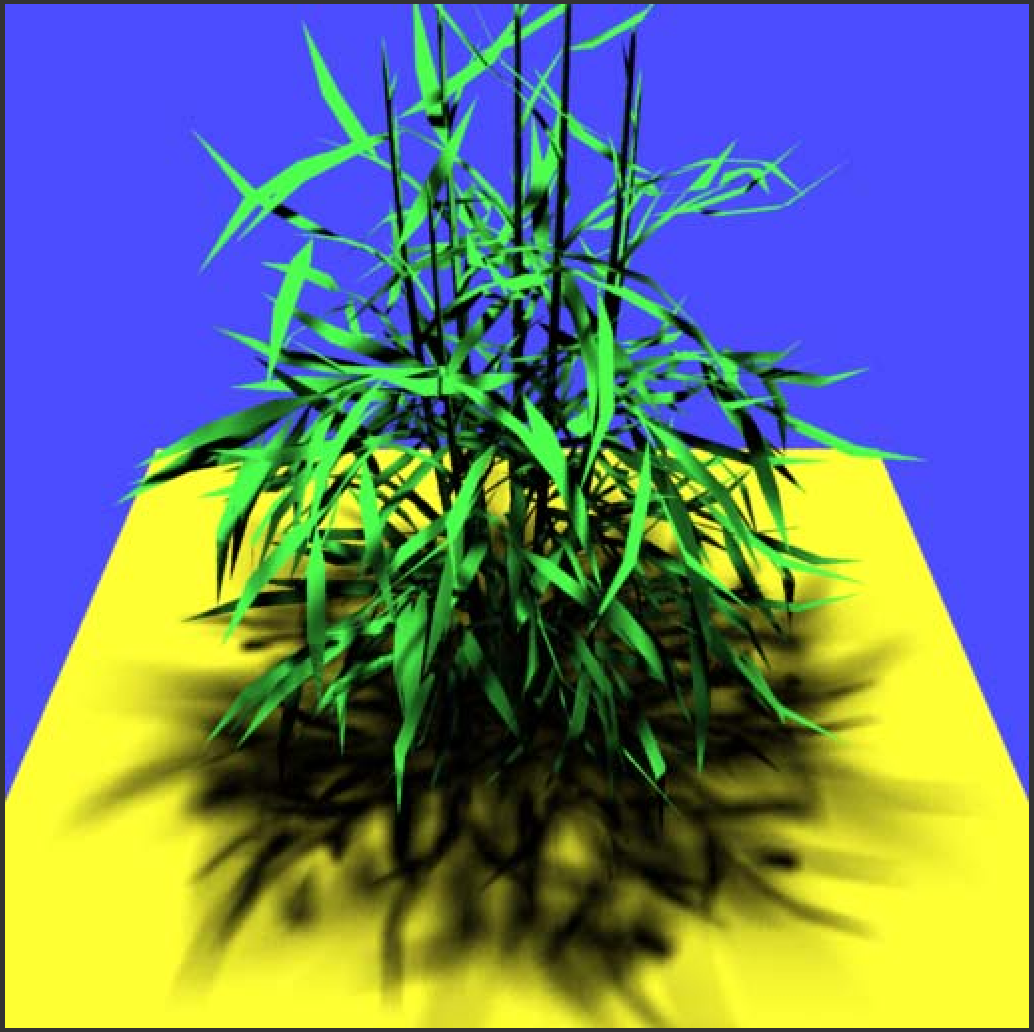
Jittered Sampling

- AKA *stratified sampling*,
- Divide pixel into a grid of *subpixels*
  - Sample each subpixel at a random location
- Combines the advantages of both uniform and random sampling
  - filters high frequencies
  - frequencies greater than subpixel sampling rate turned into noise
- Commonly used

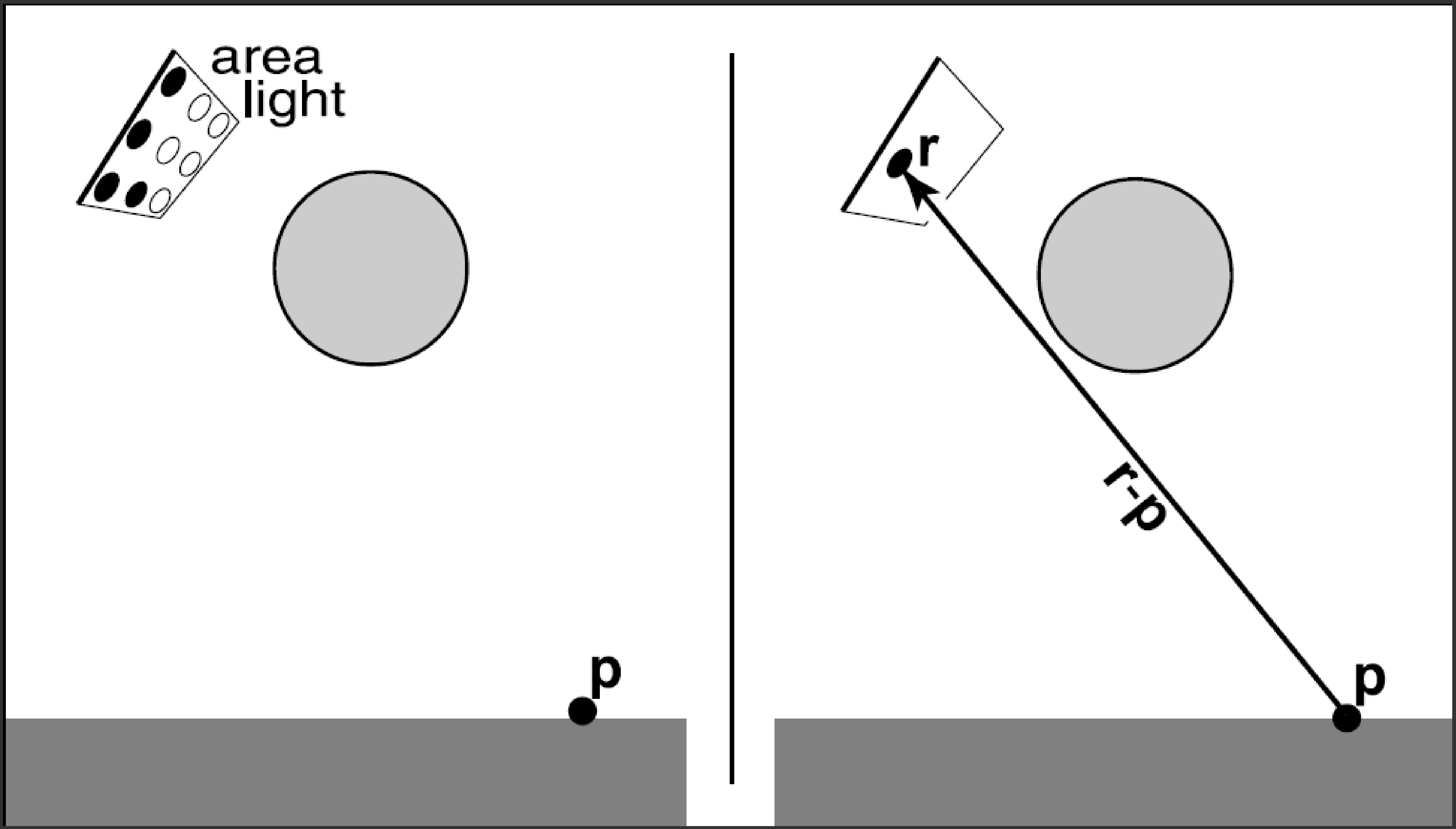


Soft shadow

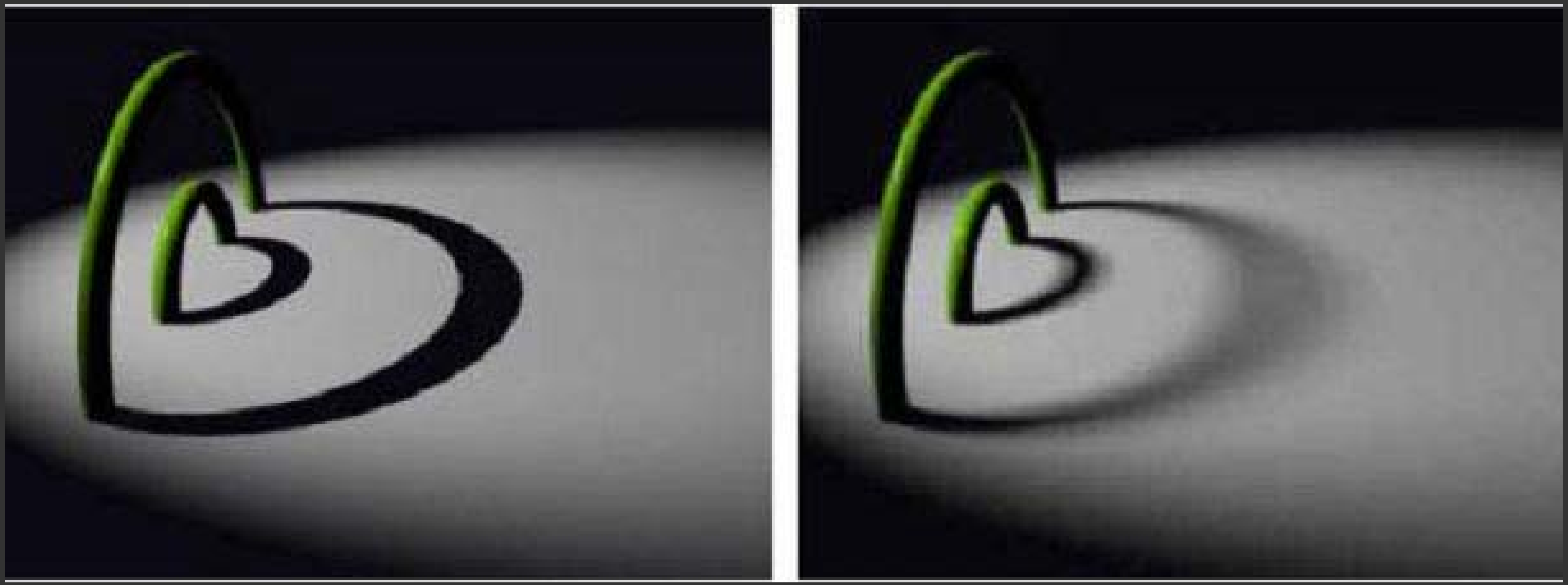




Soft Shadow



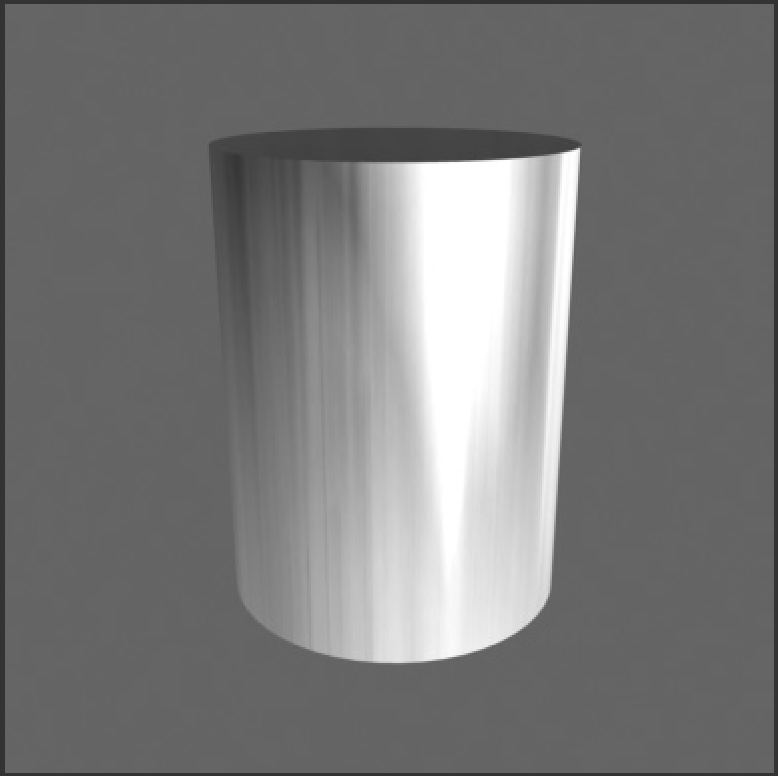
Comparison

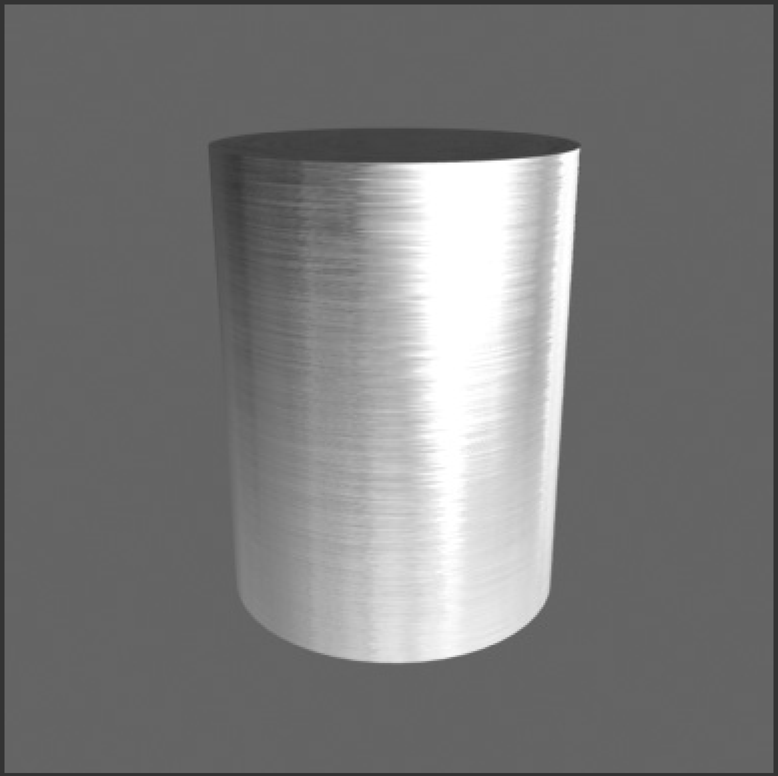


Glossy Surface

http://www.neilblevins.com/cg_education/brushed_metal/brushed_metal.htm

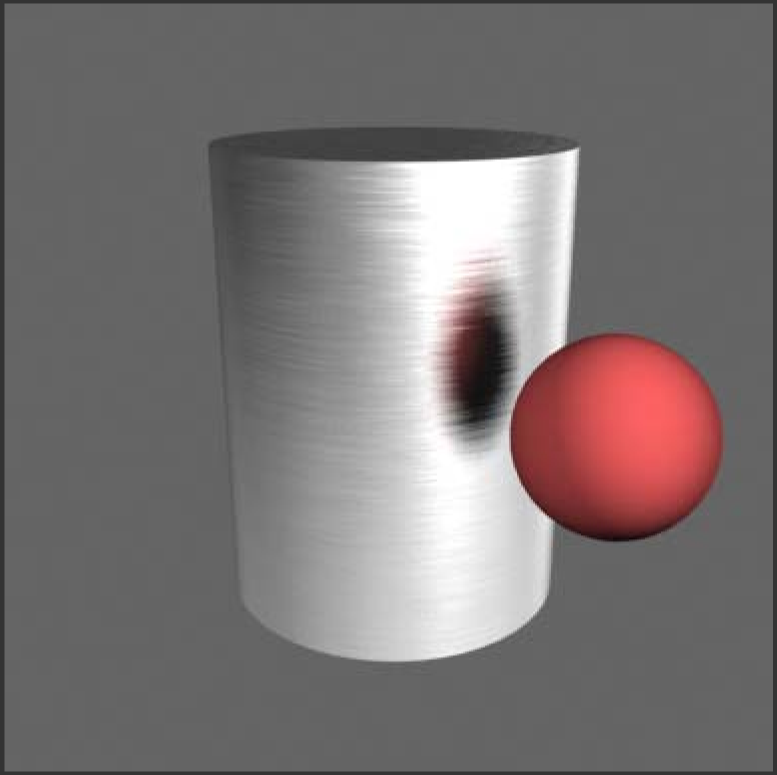
Vertical vs Horizonal roughness





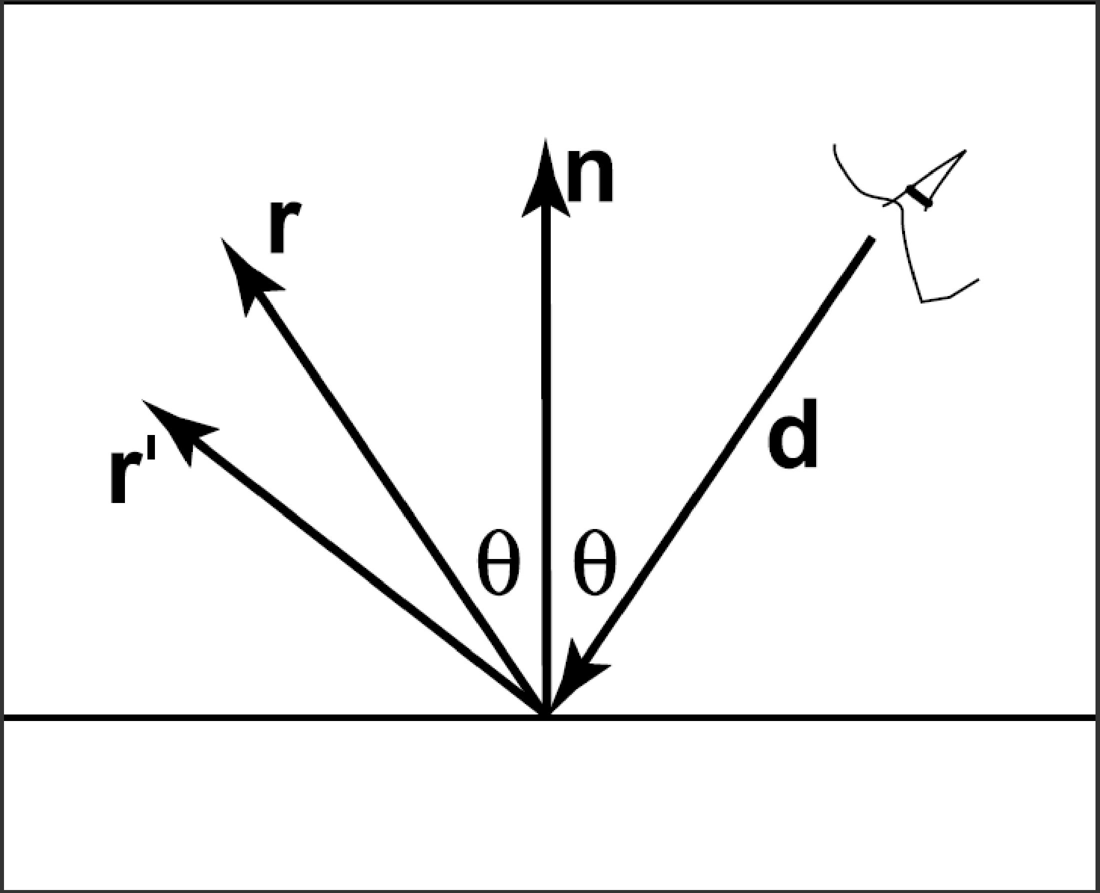
http://www.neilblevins.com/cg_education/brushed_metal/brushed_metal.htm

Ray tracing a glossy surface



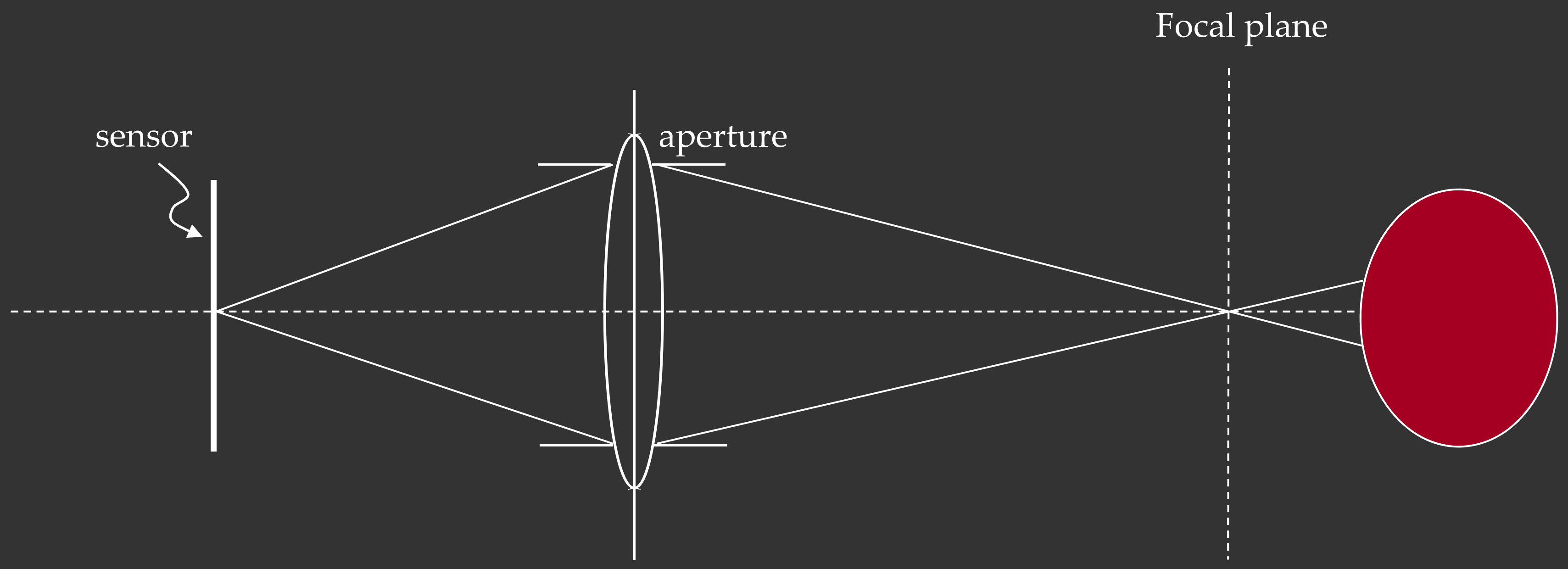
http://www.neilblevins.com/cg_education/brushed_metal/brushed_metal.htm

Ray tracing a glossy surface

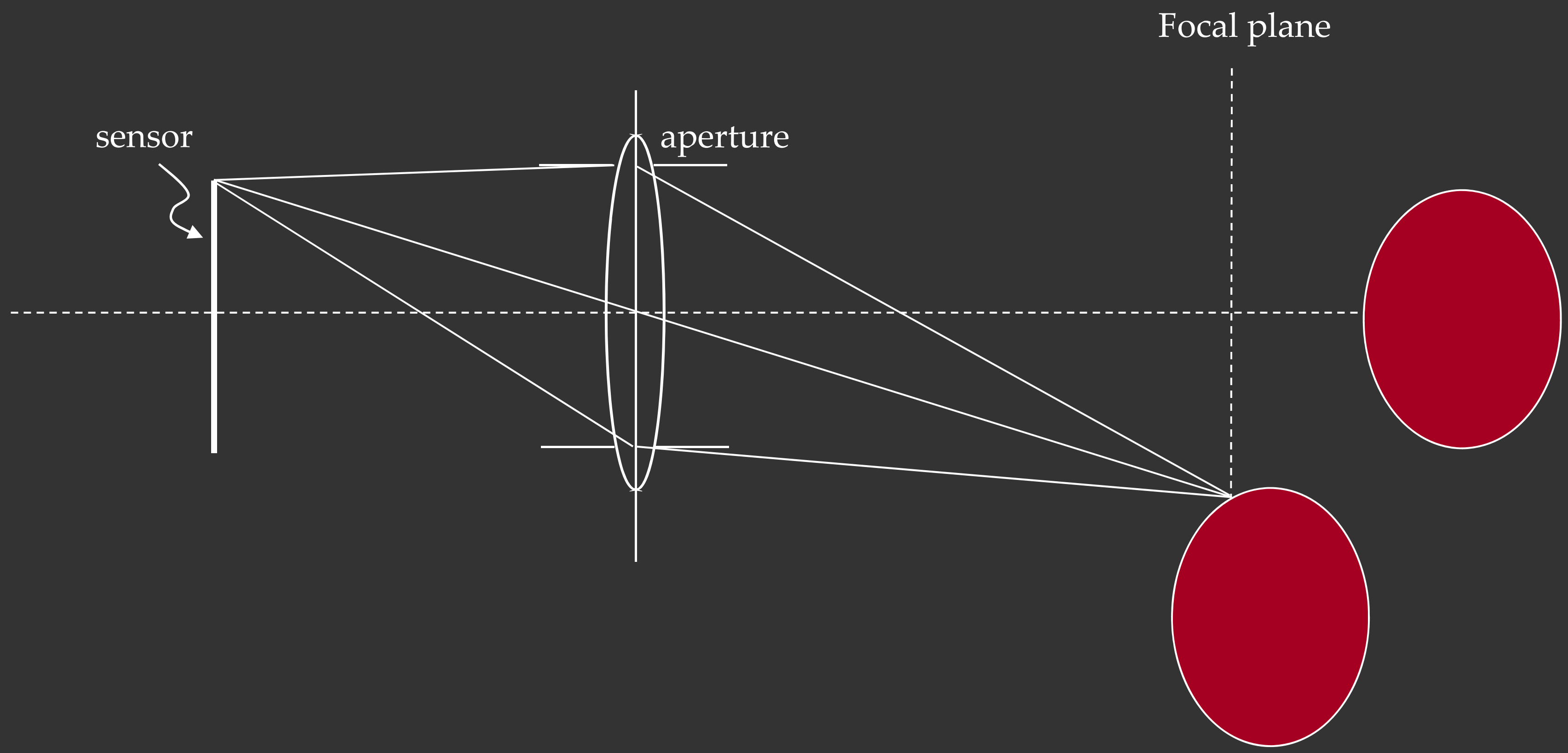


Depth of Field

Depth of Field



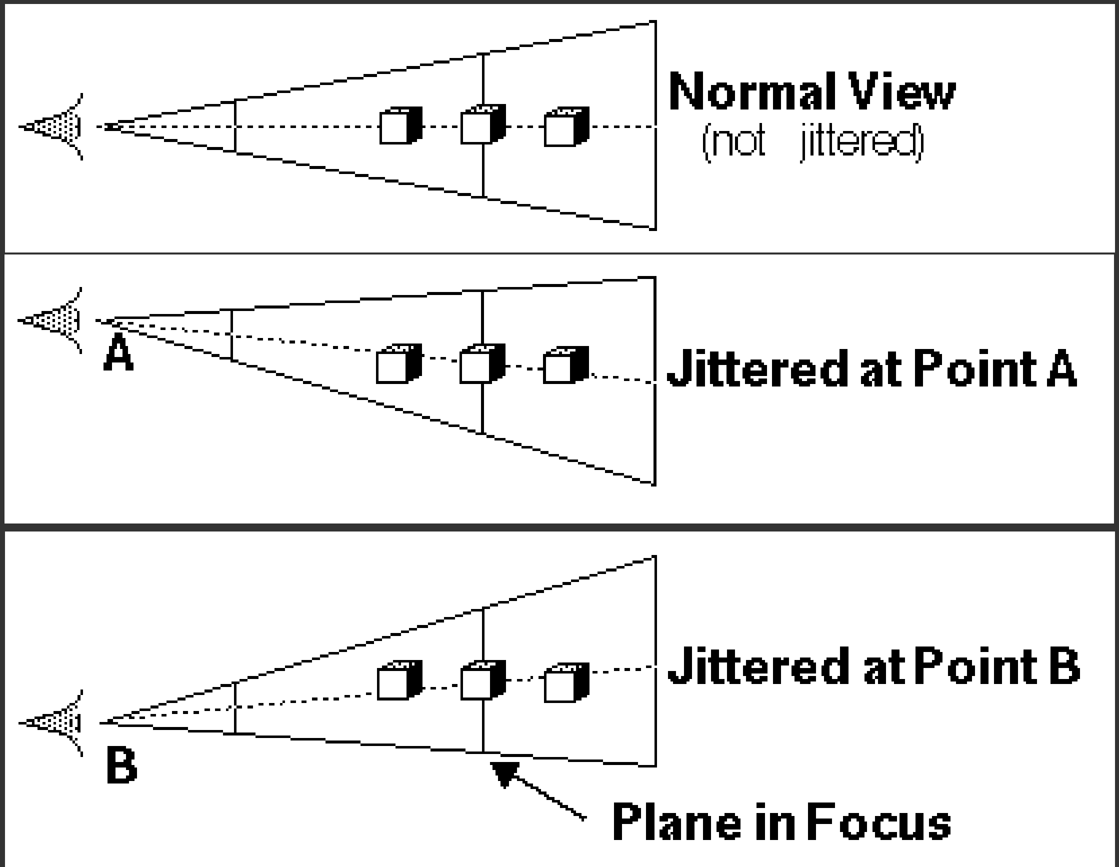
Depth of Field



Depth of Field in OpenGL

Depth of Field in OpenGL

- Render an image at each jittered location
- Then average the images



Motion Blur

- Ray trace a moving scene at different time instance and average the images

Motion Blur in OpenGL

- Render a moving scene at different time instance
- Average the images (using Accumulation buffer)

Ray tracing examples

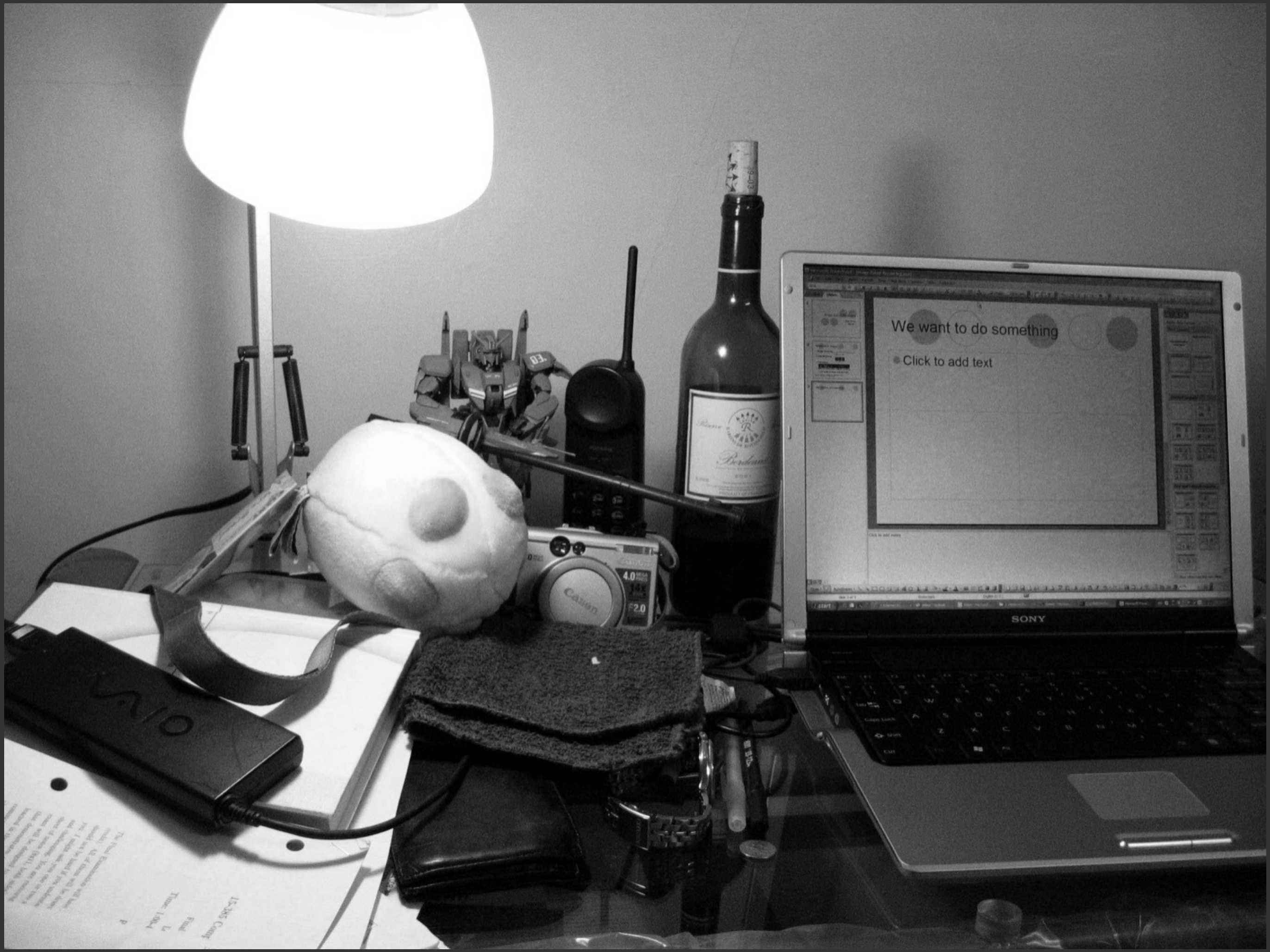
Ray tracing examples

Ray tracing examples

Image Based Rendering

- Motivation
  - Realistic Rendering requires
    - realistic 3D models
    - realistic material models
    - takes time

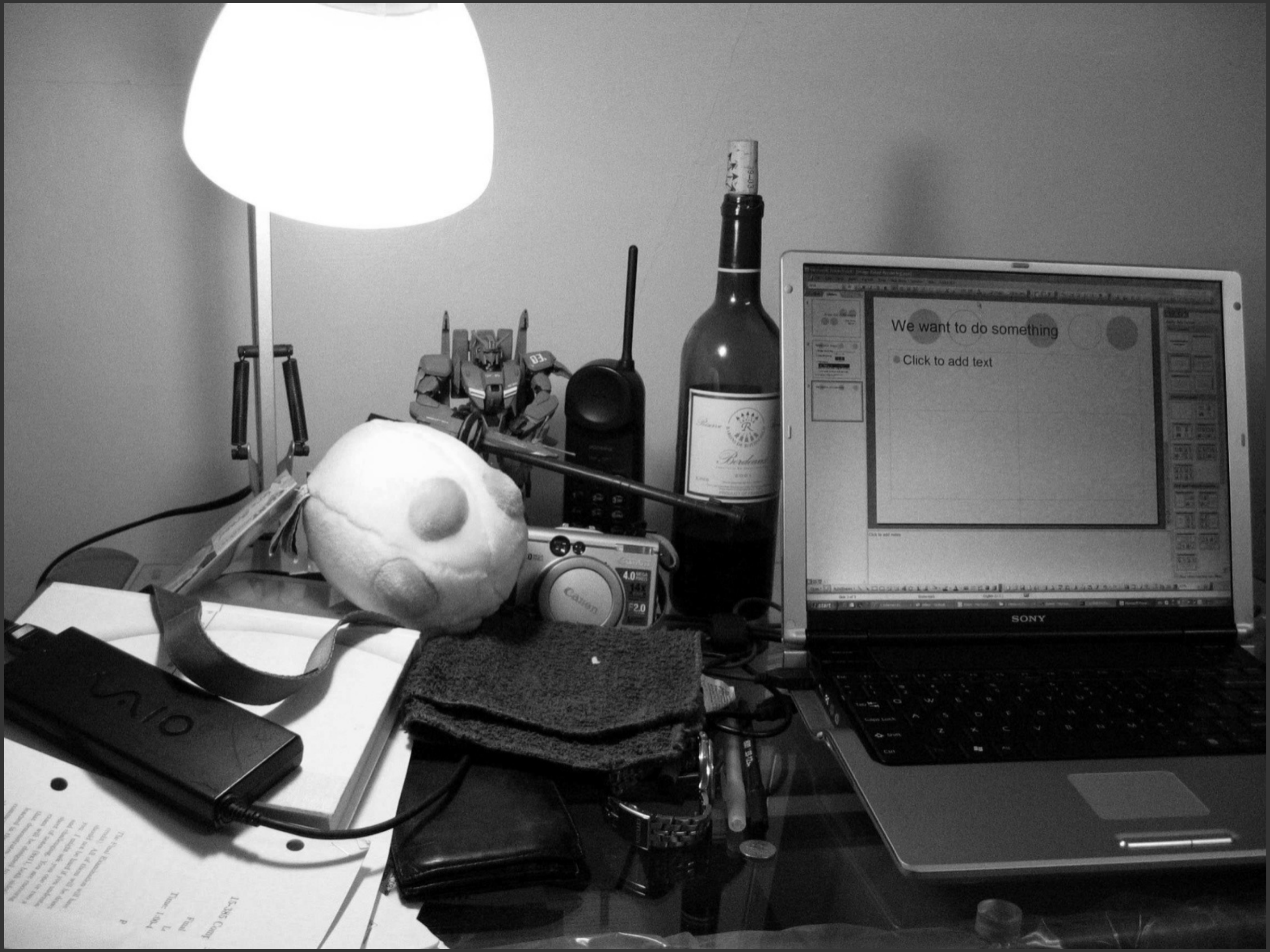
Rendering a desktop




Bordeaux
2001


Backspace
Enter

Rendering a desktop



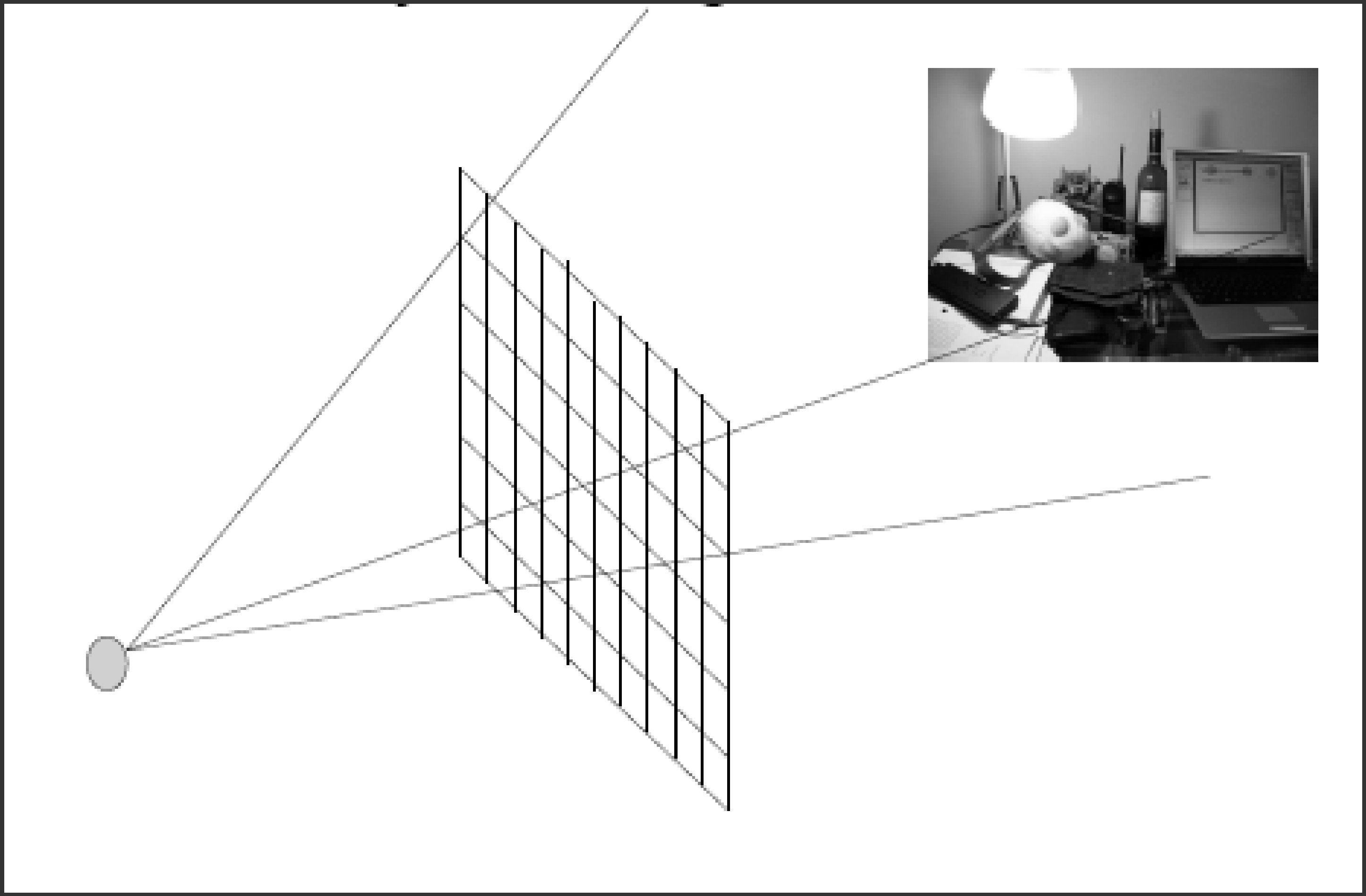
Rendering in real-time, with global illumination effect (e.g. inter-reflection)

Image Based Rendering

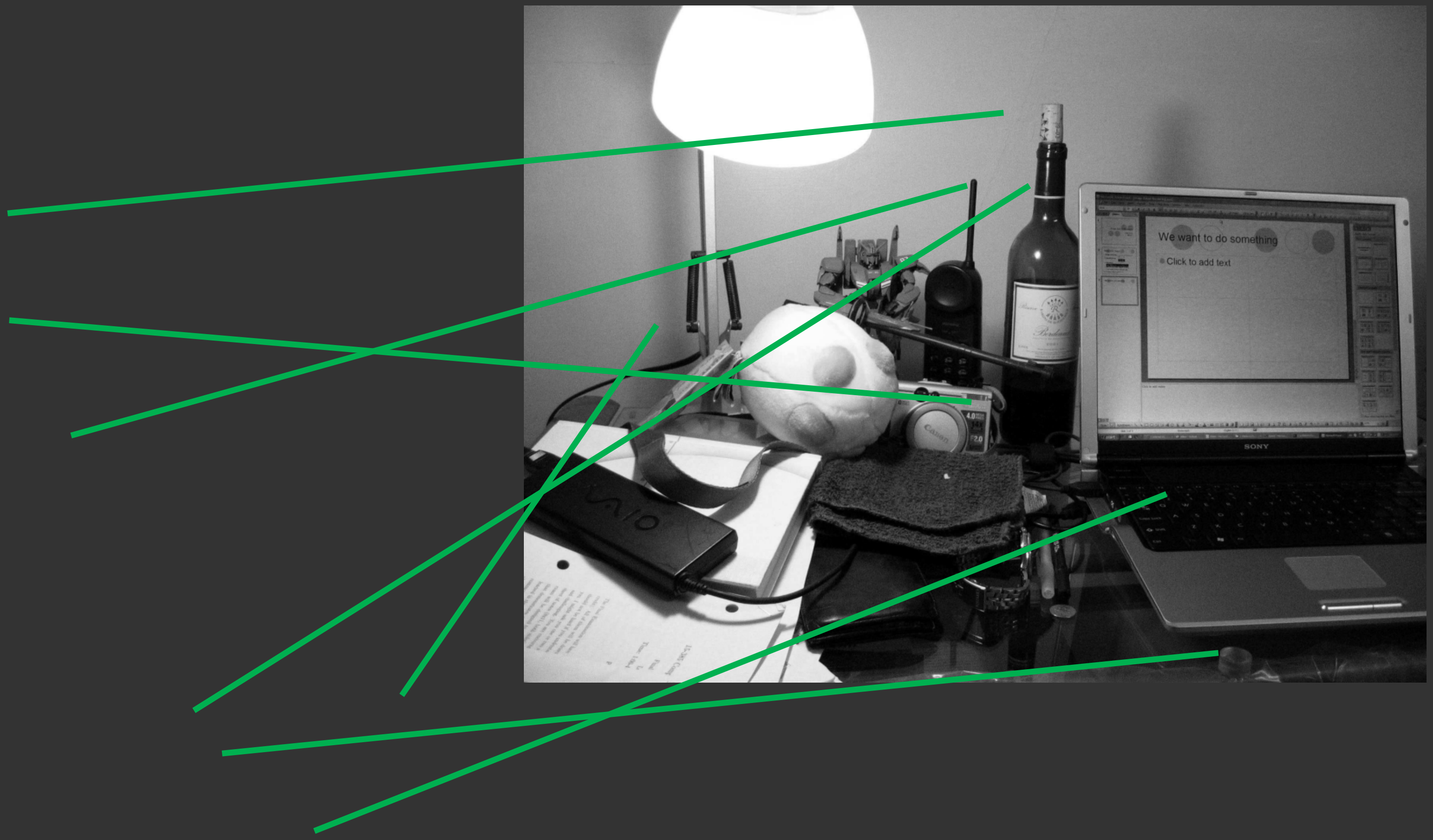
- Fast Realistic Rendering without 3D models

Start from Ray Tracing

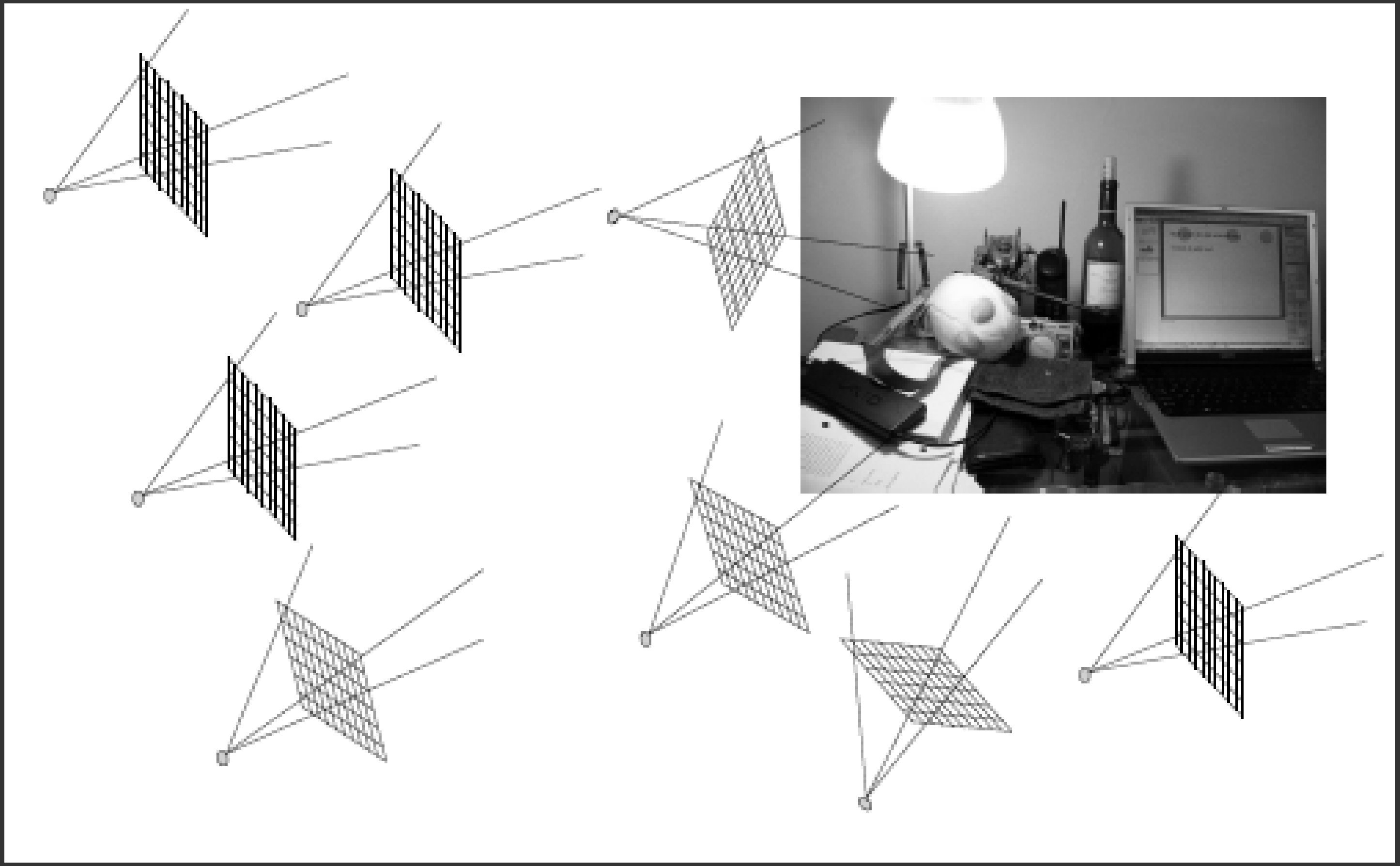
- Rendering is about computing color along each ray



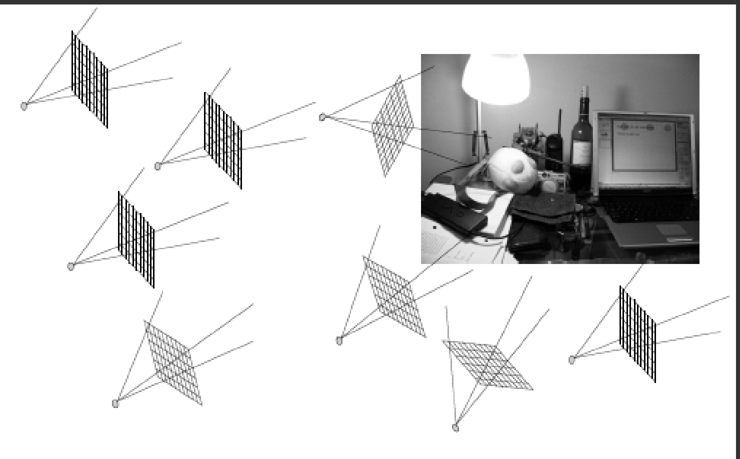
Sampling Rays

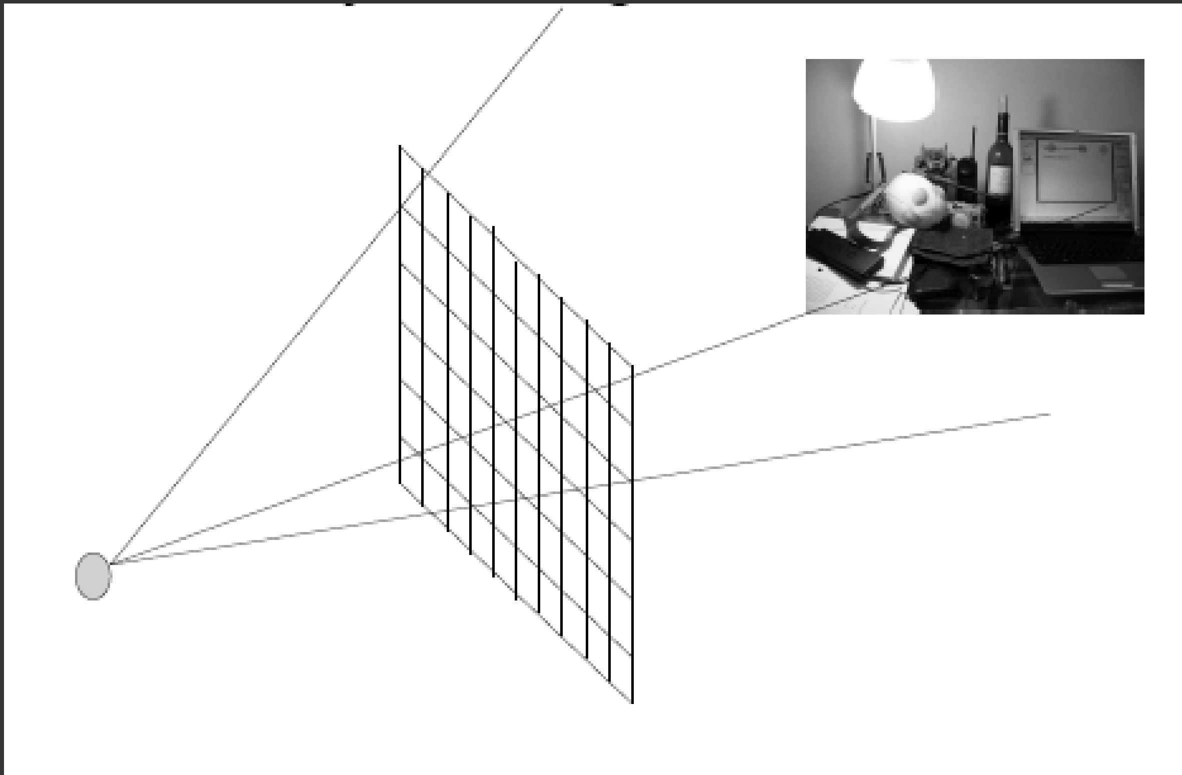


Sampling Rays by Taking Pictures



Rendering as Ray Resampling

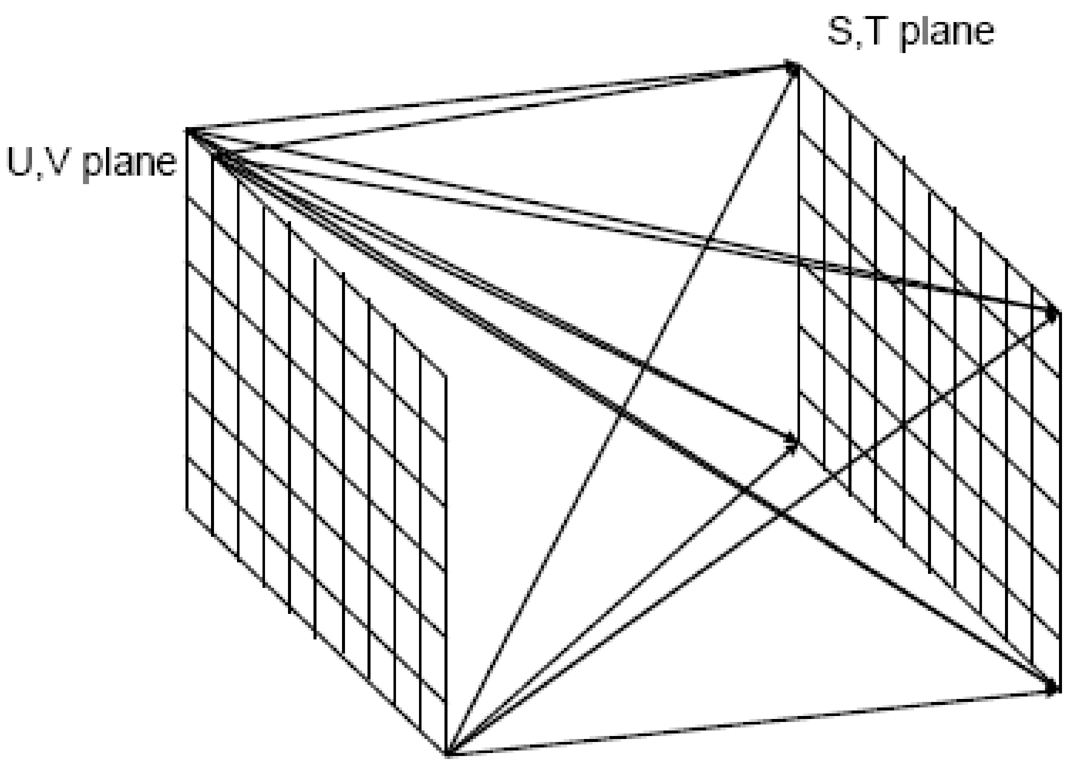


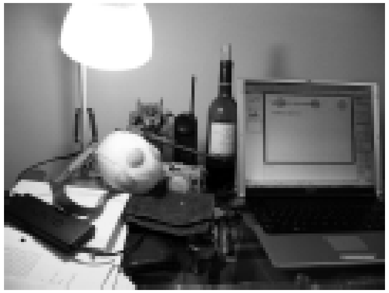


Ray space

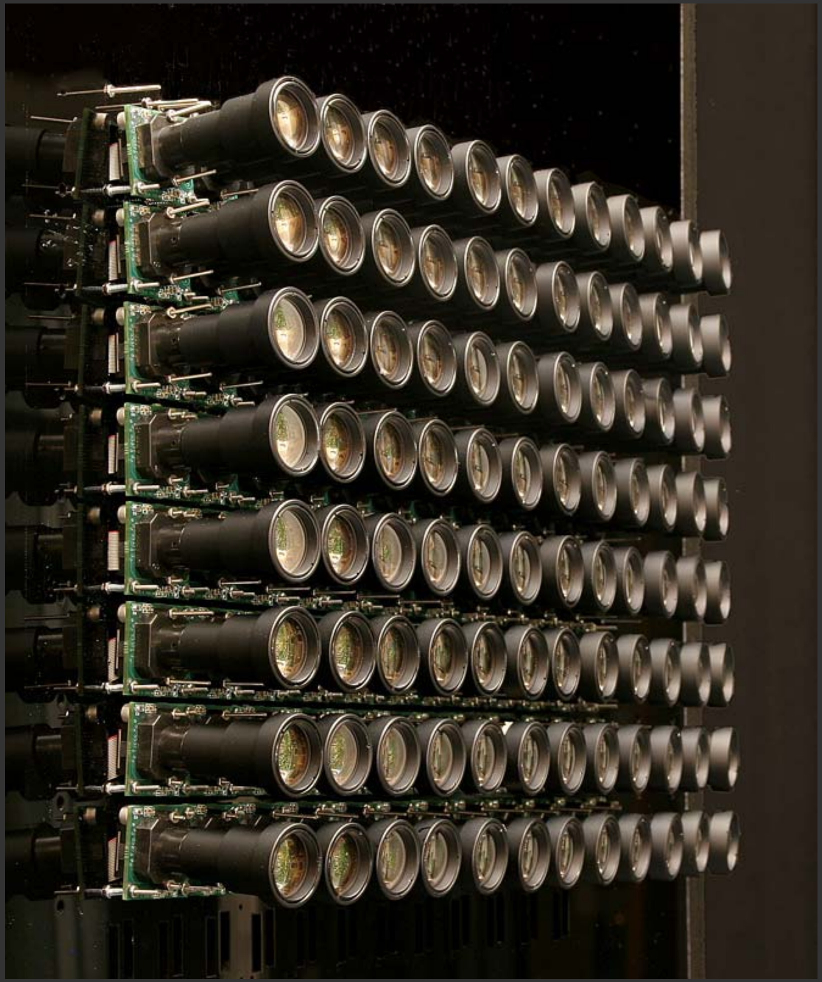
- How to parameterize the ray space
- How to sample and resample rays

Two Plane Parameterization



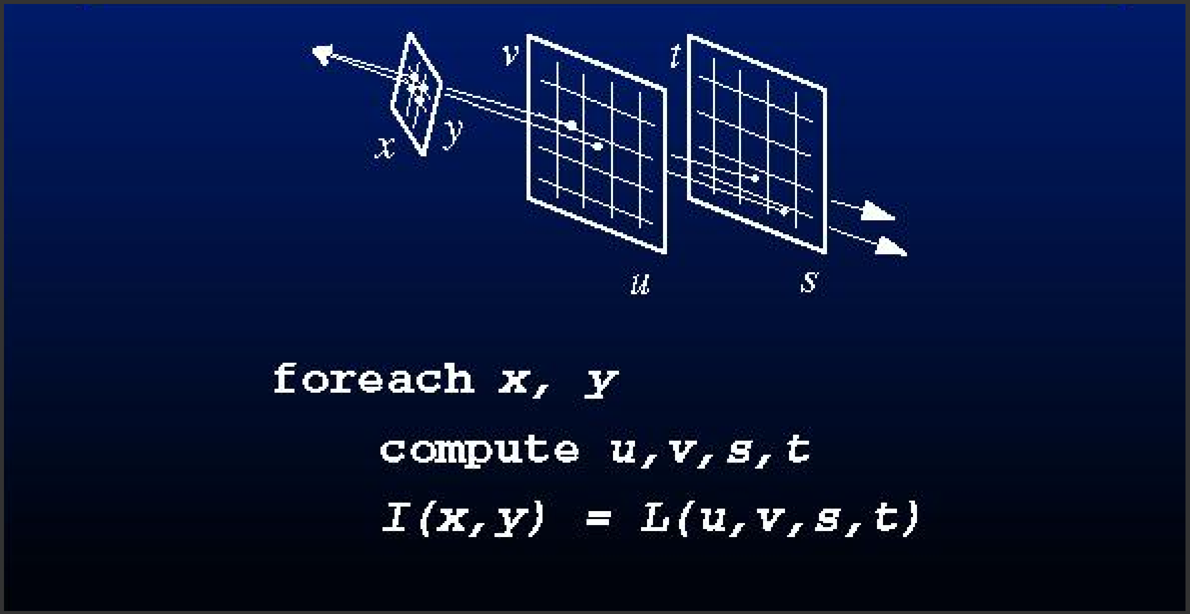


Stanford Camera Array



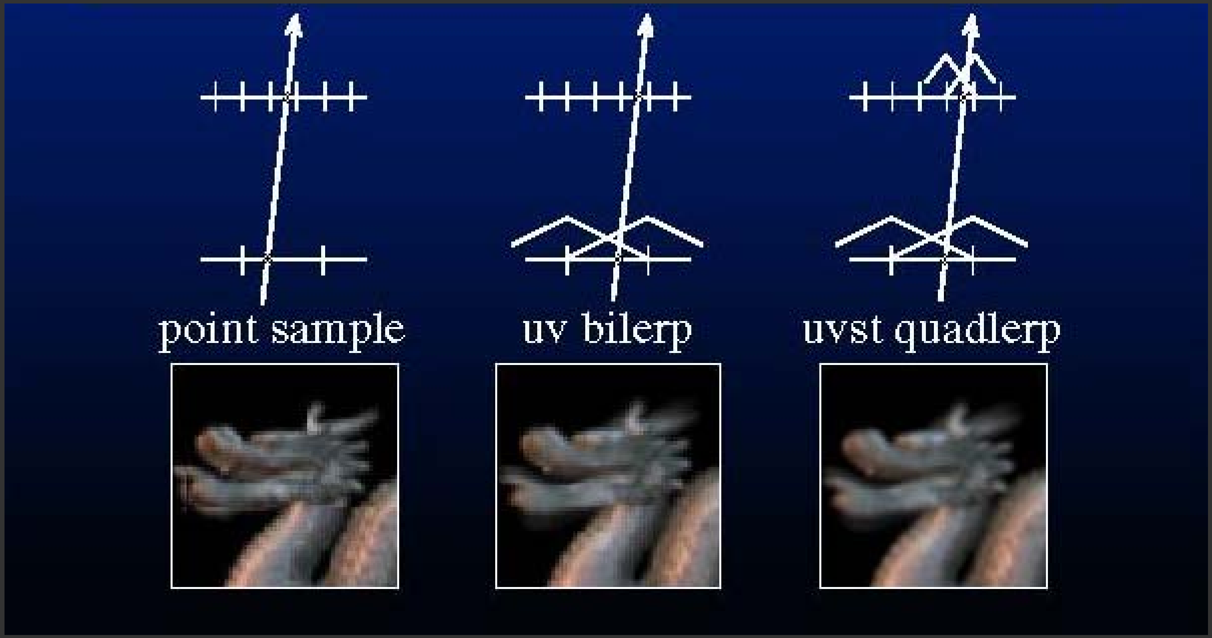
Light Field Rendering

- Very Fast



Light Field Rendering

- 4D interpolation



Light Field Rendering

- Don’t need to model anything:
  - surface model,
  - volumetric model,
  - lighting model,
  - surface property model…
- NOTHING but sampling and resampling rays.

Application in Movies

Capture scene with a camera array

Bullet time in Games

[Max Payne](#) (2001)

Discussion

- Limitation
  - Sampling density must be high
  - Fixed Illumination, static scene

Methods using Fewer Cameras

- High-quality video view interpolation using a layered representation. C. L. Zitnick, S.B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, *SIGGRAPH* 2004

  http://research.microsoft.com/~larryz/videoviewinterpolation.htm

CS559: Computer Graphics

Final Review

Li Zhang

Spring 2010

What's not in the final

- OpenGL and FLTK syntax
- Image based Rendering

Eyes and Cameras

- Camera obscura
  - Pinhole, lens
  - Different ways of capturing color
- Optical effect
  - Motion blur
  - Depth of Field

Images

- Minimum Sample requirement
  - Sampling theorem
- Re-sampling
  - Up-sampling, down-sampling
  - Anti-aliasing
- Compositing
  - Alpha channel

Image Filtering

- Convolution
  - Continuous and discrete
- Linear filter
  - Blur, shift, sharpen, edge detection…
- Painter algorithm, Project 1
  - Iteratively apply strokes

Image warping

- 2D transformation
  - Scale, Rotate, affine, translate, …
  - Inverse transformation
- Properties of 2D transformations
  - Line to line, parallel to parallel, …
- Homogeneous transformation
- Forward warping
  - Splatting
- Inverse warping
  - Reconstruction

Image morphing

- What do we need?
  - Avoid ghosting
- How to do it?
  - Warping + blending

3D transform

- Homogenous Coordinate
  - Point vs direction
  - Transforming normals
- 3D rotation
  - property
  - Different representation
  - Geometric interpretation
- Concatenation of transforms
  - Hierarchical modeling

Projection

- Graphics pipeline
- Orthographic vs perspective projection
  - Matrix representation
  - Vanishing point
- View frustum
  - Clipping plane, Field of view
  - Convert to projection matrix
- Canonical view volume
  - From perspective view volume

Scan conversion and visibility

- Draw lines and triangles
  - Tricks to make it fast
  - Anti-aliasing
- BSP
  - How to construct and how to use
- Z buffer vs A buffer
  - Pros and cons

Shading

- Phong shading model
  - Emission, diffuse, specular
- Types of light sources
  - Point, spot, directional
- Shading interpolation
  - Flat, Gouraud, and Phong

Curves

- Implicit vs Parametric Curves
- Polynomial Curves
  - How to evaluate polynomial
  - How to compute the curve
  - Problem with high order polynomials
- Piecewise cubic polynomial
  - Continuity: C0,C1,C2
  - Local control
  - interpolation

Curves

- Natural, Hermite, Catmull-Rom, Cardinal, Bezier,
  - Commonality and differences
- Bezier curves
  - Subdivision
  - De Casteljau
  - Generalization
  - …

Texture

- Calculate texture coord
  - Perspective correct interpolation
- Texture resampling
  - Antialiasing: Mipmap, Ripmap, SAT
    - How do they work,
    - What can they do, limitation
- Other usages:
  - Bump Map, Displacement Map, 3D Texture, Environment Map, Shadow map
  - Projector texture (no requirement)

Shape

- Boundary vs Solid modeling
- Parametric, Implicit, Procedural
  - Pros and cons
- Polygon meshes
  - Why popular
  - Pros and cons
  - Data structure

Shape

- Sweep objects
- Spatial enumeration
  - Oct tree
- Bezier Patch
  - Bilinear, biquadric, bicubic
  - De Casteljau

Subdivision Curves and Surfaces

- Approximating vs Interpolating
- Regular vs Irregular vertices
- Continuity
- Loop, sqrt(3), Catmull-Clark
  - Commanality and difference
  - Piecewise smoothness (no requirement)
- Fractal Modeling
  - Terrains, trees, …

Animation

- Particle Systems
  - Euler method
  - Collision Detection and Response
- Principles of Cartoon

Raytracing

- Recursive procedure
  - Shadow, Transparency, Reflection, Refraction
  - Why inter-reflection is hard?
  - Anti-aliasing: jittered sampling, why
  - Soft shadow, glossy surface,
  - Depth of field, Motion blur
- Ray object intersection
  - Simple objects: triangle, polygons, …
- Spatial data structure for Acceleration
  - BSP, octtree, grid