CS 354
Ray Casting & Tracing

Mark Kilgard
University of Texas
April 17, 2012

Today's material

- In-class quiz
  - On *typography* lecture
- Lecture topic
  - Finish *digital typography*
  - Project 3
  - *Ray tracing & casting*

My Office Hours

- Tuesday, before class
  - Painter (PAI) 5.35
  - 8:45 a.m. to 9:15
- Thursday, after class
  - ACE 6.302
  - 11:00 a.m. to 12

- Randy's office hours
  - Monday & Wednesday
  - 11 a.m. to 12:00
  - Painter (PAI) 5.33

Last time, this time

- Last lecture, we discussed
  - *Digital typography*
- This lecture
  - *Ray casting and tracing*
- Projects
  - Project 3 due Wednesday, April 18, 2012
    - Tomorrow!
  - Project 4 on ray tracing on Piazza

Daily Quiz

On a sheet of paper
- Write your EID, name, and date
- Write #1, #2, #3, #4 followed by its answer

1. **Multiple choice:** Pica and Em are:

   a) Names of two commonly used type faces

   b) Units of typographic measurement

   c) Two dominant types of Bezier curves

   d) Graphics researchers responsible for font hinting

2. **True or False:** This text is written in a serif type face.

3. **Multiple choice:** Unicode supports how many character points?

   a) 256

   b) 65,536

   c) Over ¼ million

   d) 4,294,967,296

4. **Multiple choice:** Kerning adjusts what property of text?

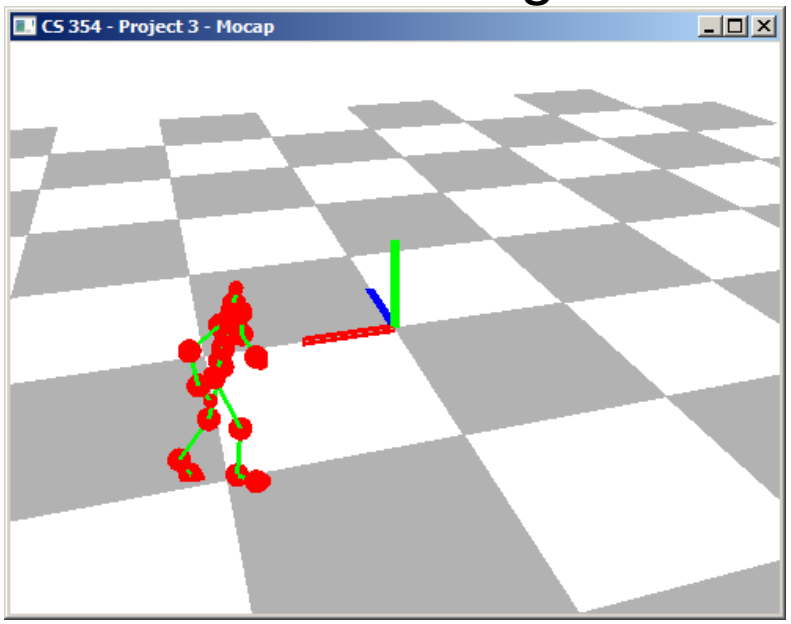
   a) Spacing between characters

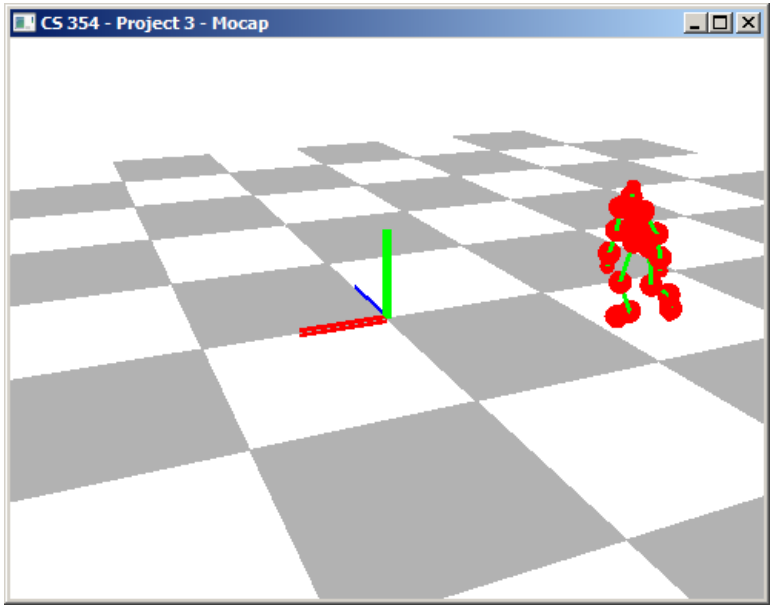
   b) Whether accent marks are included on glyphs

   c) Control of vertical layout

Project 3

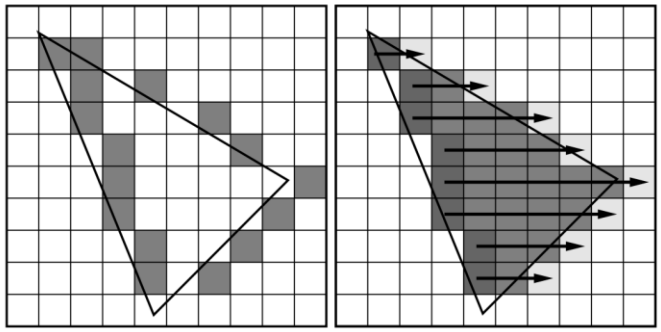
- Accept Biovision Hierarchy (BVH) files containing motion capture data
  - Hierarchy of affine transformations
- Visualize a stick figure
  - Animate the figure



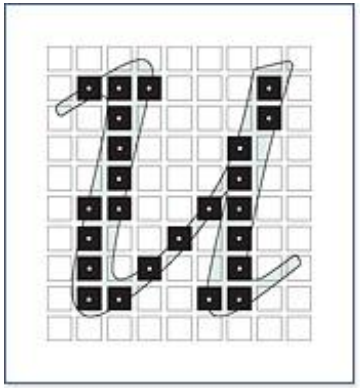


Rendering Outline Glyphs

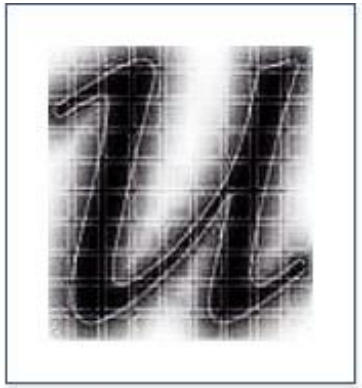
- Conventional method
  - CPU-based scan-line rasterization
  - Augmented by hinting



- Adaptively Sampled Distance Fields (ADFs)
  - MERL's Saffron type system
- Glyphs are static so pre-computation is effective
  - Bitmaps often cached

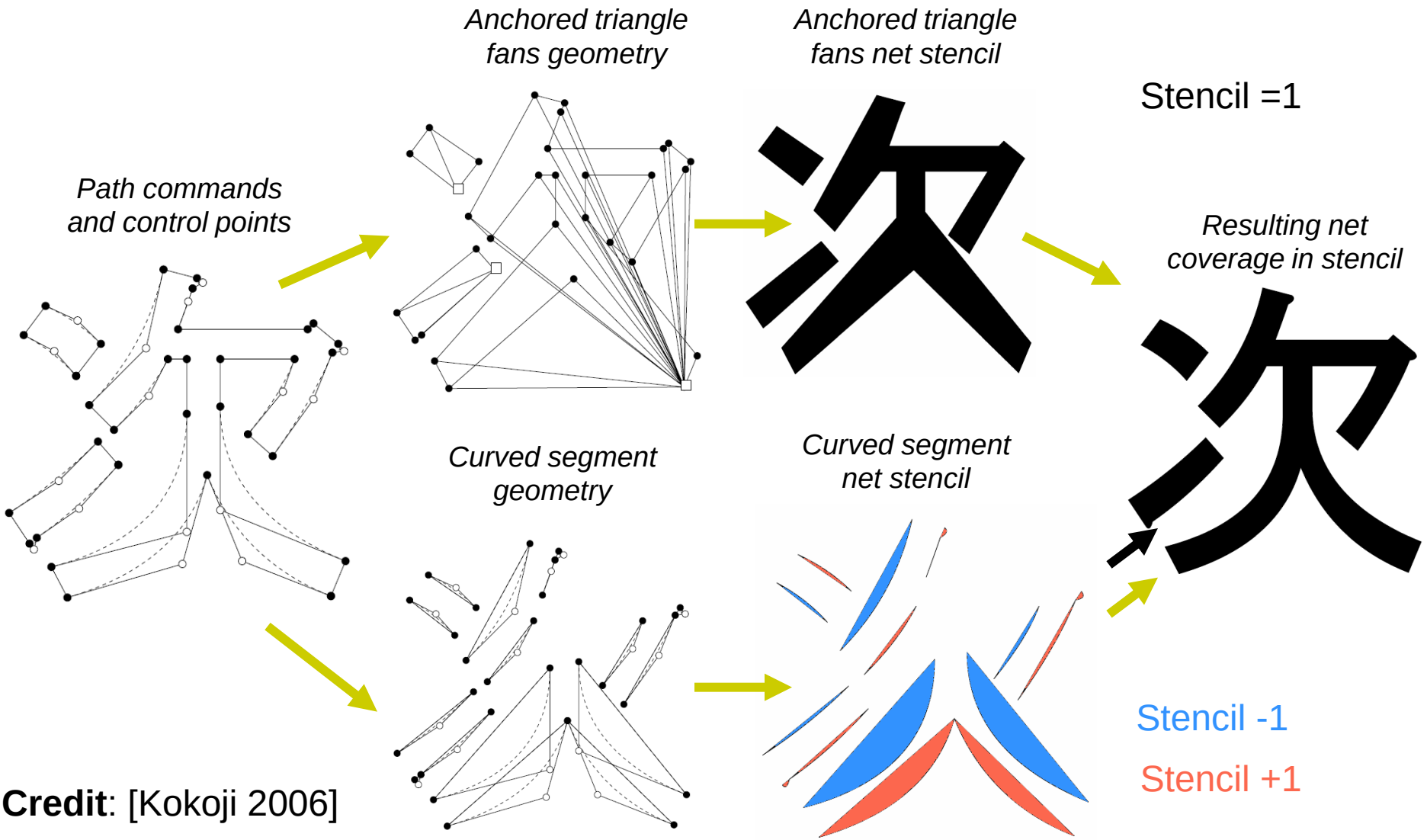


Conventional technology



Saffron

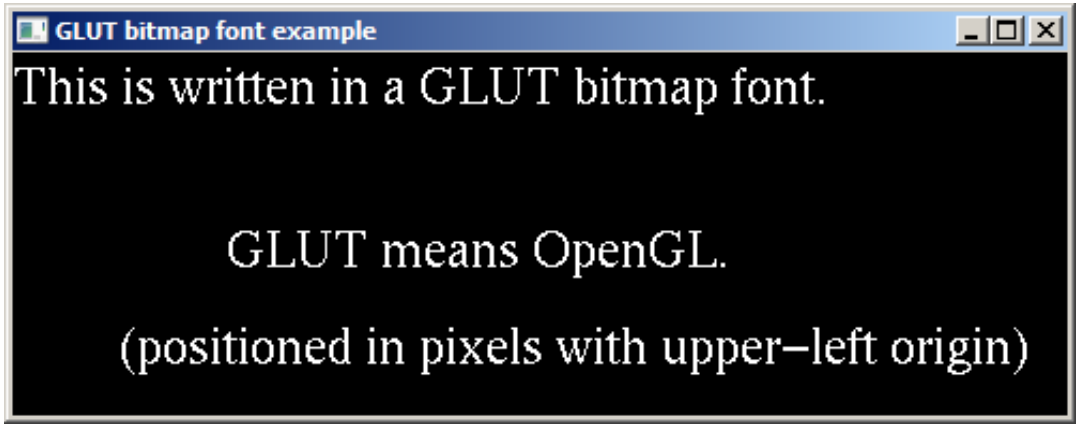
Massively Parallel GPU-accelerated Path Rendering Visualized



Credit: [Kokoji 2006]

OpenGL Bitmap Fonts

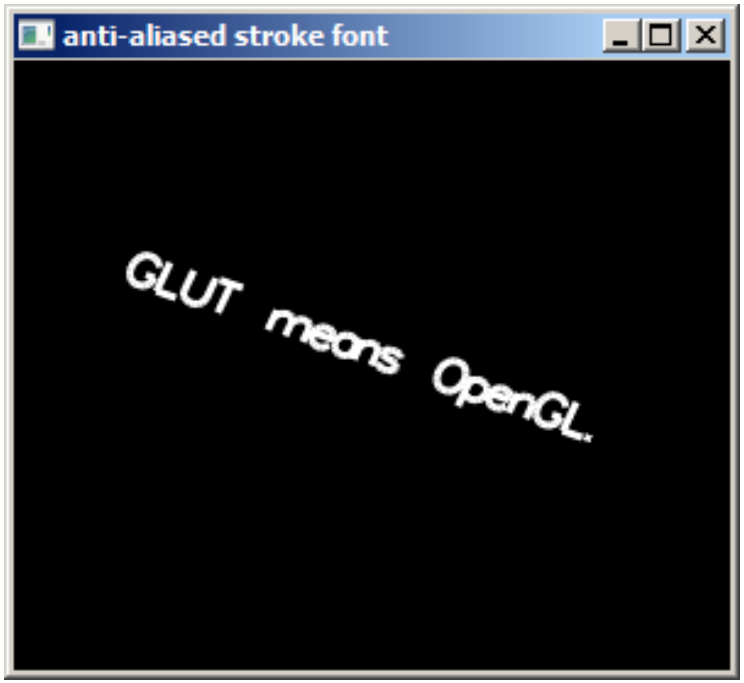
- GLUT includes implementation
  - glutBitmapCharacter
  - Calls glBitmap with pre-compiled bitmap data



- Also window system specific routines to get bitmaps from system fonts
  - wglUseFontBitmaps for Windows
  - glXUseXFont for X Window System (Linux)

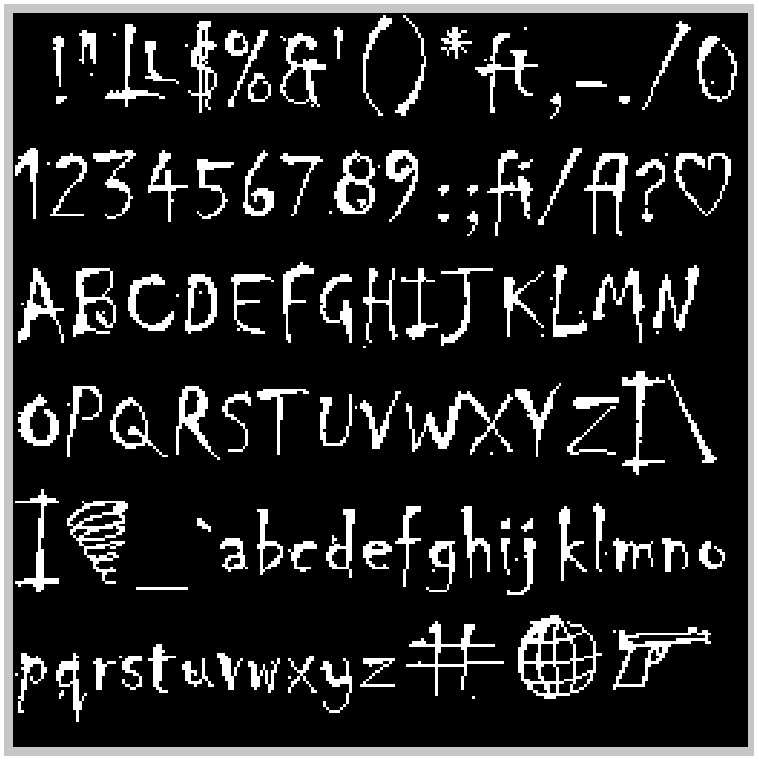
OpenGL Stroke Fonts

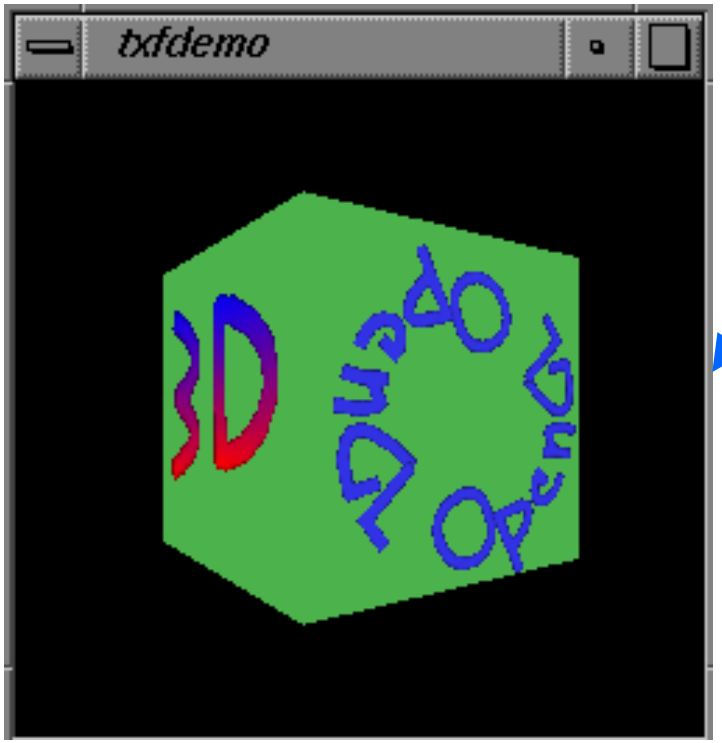
- Draw glyph's stroke as line segments
  - Can transform arbitrarily
  - Use glEnable(GL_LINE_SMOOTH) for anti-aliasing
  - glutStrokeCharacter does this



Other Text Approaches

- Pack glyphs in texture atlas
  - Then draw textured rectangles with the correct texture coordinates for each glyph





First-class Text Support in Various Rendering Systems

Rendering system	Vendor / Sponsor	First-class text?
Cairo	Open source	Yes
Direct2D / Direct3D	Microsoft	Yes
Flash	Adobe	Yes
GDI	Microsoft	Yes
Java 2D	Sun	Yes
OpenGL	**Khronos**	**No**
OpenVG	Khronos	Yes
Office Open XML	Microsoft / ECMA	Yes
PDF	Adobe	Yes
PHIGS	ISO	Yes
PostScript	Adobe	Yes
Qt	Nokia	Yes
Quart 2D	Apple	Yes
Scalable Vector Graphics (SVG)	World Wide Web Consortium	Yes
Silverlight	Microsoft	Yes
Xlib	X Consortium	Yes

How did OpenGL's lack of text support last for so long?

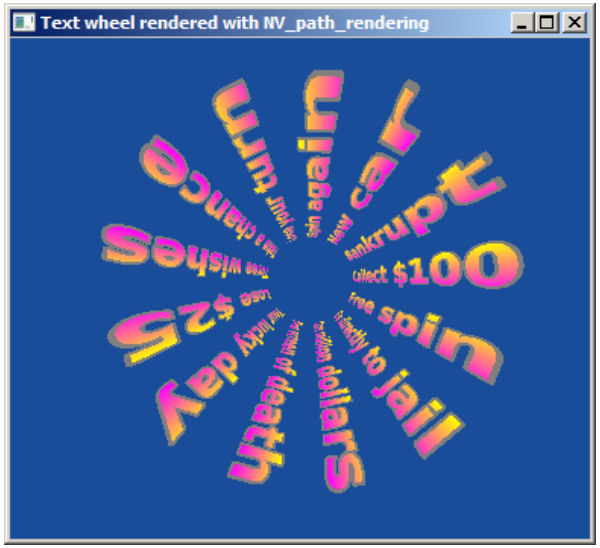
- **OpenGL was designed in 1992**
  - Pre-Unicode world
  - Fonts were bitmaps of ASCII back then
    - Now TrueType and OpenType dominate
  - The primitive initial state of OpenGL's font support has become mistaken for a philosophical dictate
- **Text-based applications uniformly ignore OpenGL**
  - Not a good thing
- **3D applications have their expectations adjusted to expect lousy text**
  - Not a good thing
  - Example: Quake 2 console text is miniscule because textured bitmap characters assumed text sized for 800x600 display

The World Has Changed Since 1992

- Unicode universally accepted now
  - Systems ship with near-complete, resolution-independent Unicode fonts now
  - Good fonts come with your operating system license
  - International character of web makes UTF-8 text common today
  - Windows adoption of UTF-16 makes that common within Windows
- Web has expanded font repertoire of systems
  - Content providers and users expect wide range of available fonts
  - Fonts have standard names embedded in web content
  - These font names span operating systems
- Screen resolution has increased
  - **1992**: 640x480 aliased
  - **2009**: 1920x1200+ multisampled
  - Needing all text to be in small point sizes to fit screen isn't a mandate anymore
- 3D support and acceleration is pervasive now
  - Text makes sense mixed with 3D content today

Direct Path Rendering of Text in OpenGL

- Recent work
  - NV_path_rendering provides GPU-acceleration of paths, including glyphs
  - Glyph support is *first-class*



Digital Typography Trends

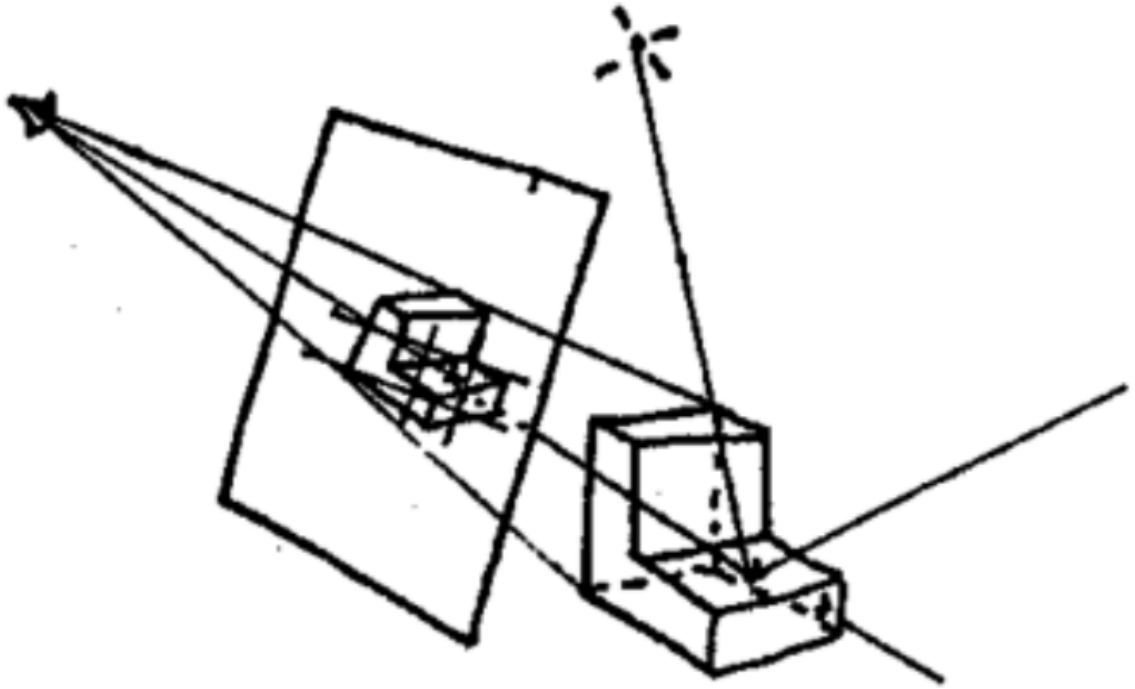
- Hinting & ClearType likely going away
  - Mainly due to denser screens
  - Expectation of antialiased font appearance
  - Text not always presented axis-aligned or black-on-white
- More font variety in web content
  - Improving standards
  - More available fonts
  - Even font variety for East Asian writing systems

OpenGL's Realism Limitations

- OpenGL is based on a pipeline model in which primitives are rendered one at time
  - No automatic shadows (except by tricks or multiple renderings)
  - No automatic multiple reflections
  - Multi-pass algorithms for shadows and reflections
- Physically plausible & global approaches
  - Rendering equation
  - Ray tracing
  - Radiosity

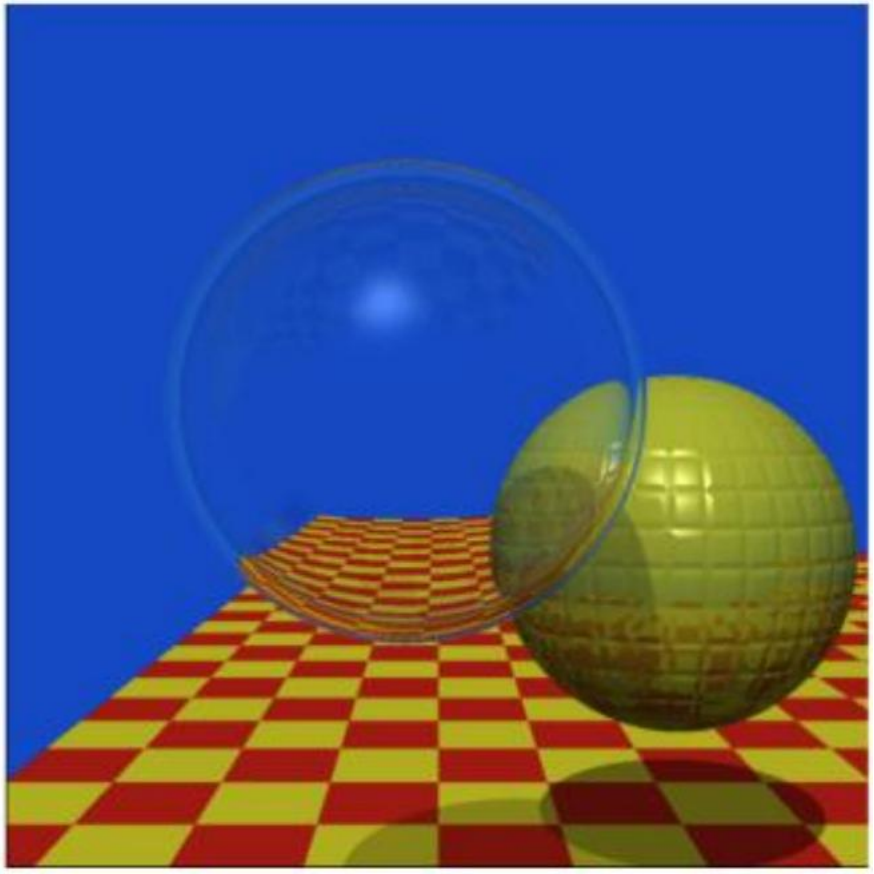
Ray Casting

- Arthur Appel (1968)
  - Cast rays from eye… what object is hit?



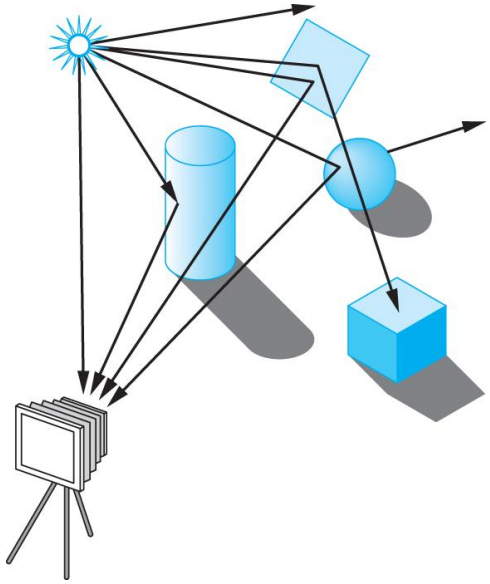
Ray Tracing

- "An Improved Illumination Model for Shaded Display"
  - Turner Whitted (1980)
  - Communications of the ACM
- Performance
  - 512x512 pixels
  - 74 Minutes on VAX (1979)
  - 6 seconds on PC (2006)
  - 60+ frames/second on GPU (2012)



Ray Tracing

- Follow rays of light from a point source
- Can account for reflection and transmission



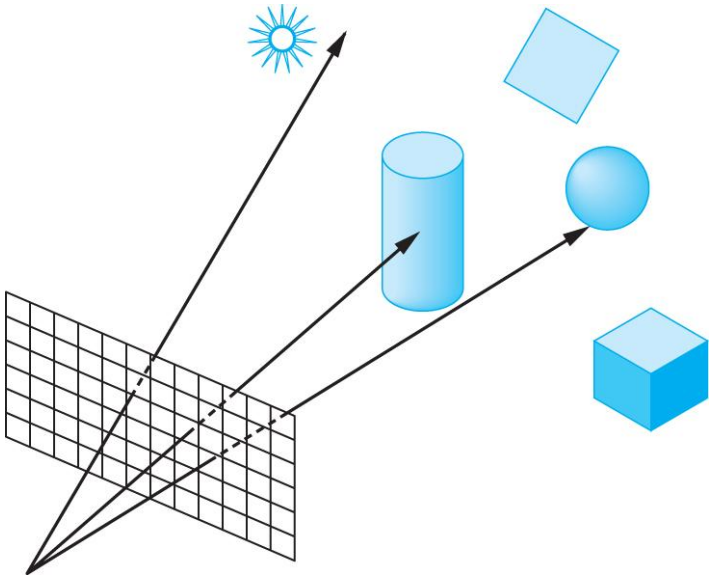
Example of Complex Ray Traced Scene

Computational Limits of Ray Tracing

- Should be able to handle all physical interactions
- Ray tracing paradigm is physically plausible
- But most rays do not affect what we see
- Scattering produces many (infinite) additional rays
- **Alternative:** ray casting

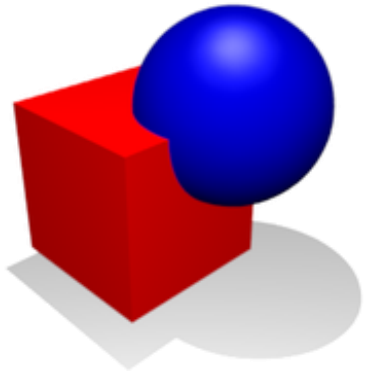
Ray Casting

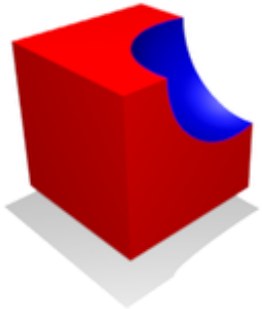
- Only rays that reach the eye matter
- Reverse direction and cast rays
- Need at least one ray per pixel



Ray Casting Quadrics

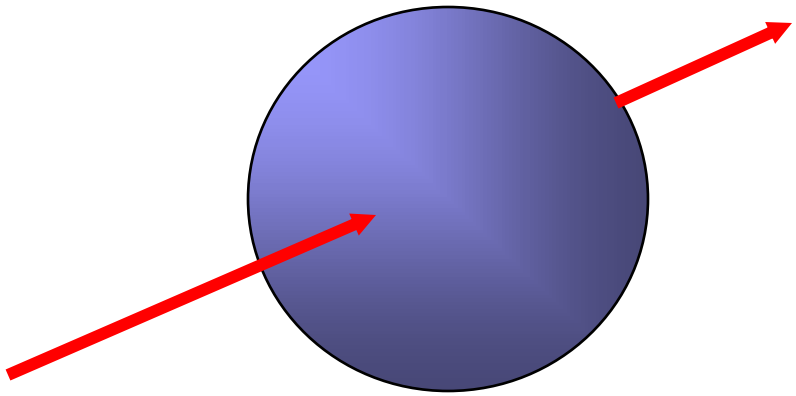
- Ray casting has become a standard way to visualize quadrics which are implicit surfaces in CSG systems
- Constructive Solid Geometry
  - Primitives are solids
  - Build objects with set operations
  - Union, intersection, set difference





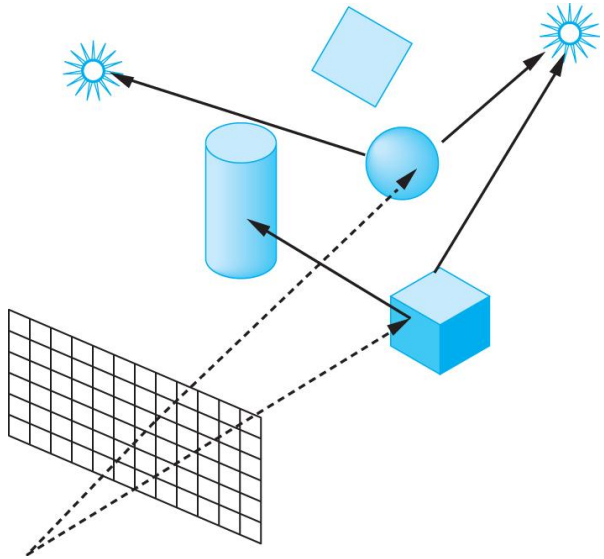
Ray Casting a Sphere

- Ray is parametric
- Sphere is quadric
- Resulting equation is a scalar quadratic equation which gives entry and exit points of ray (or no solution if ray misses)



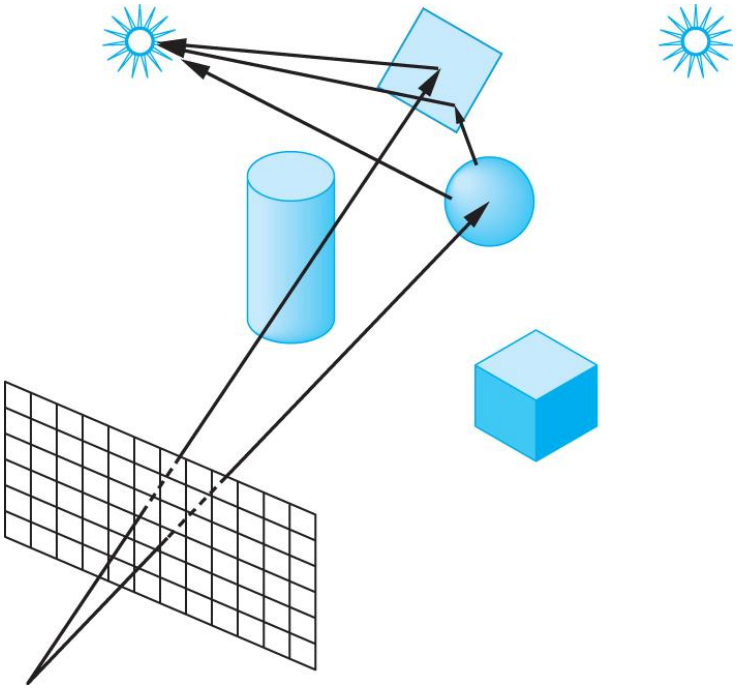
Shadow Rays

- Even if a point is visible, it will not be lit unless we can see a light source from that point
- Cast shadow or feeler rays

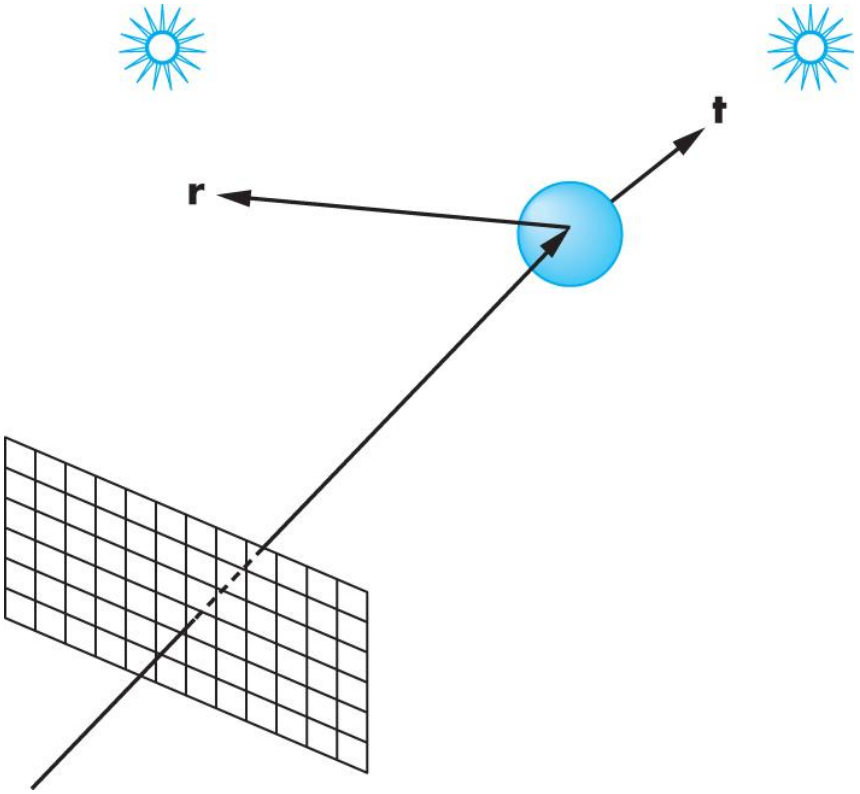


Reflection

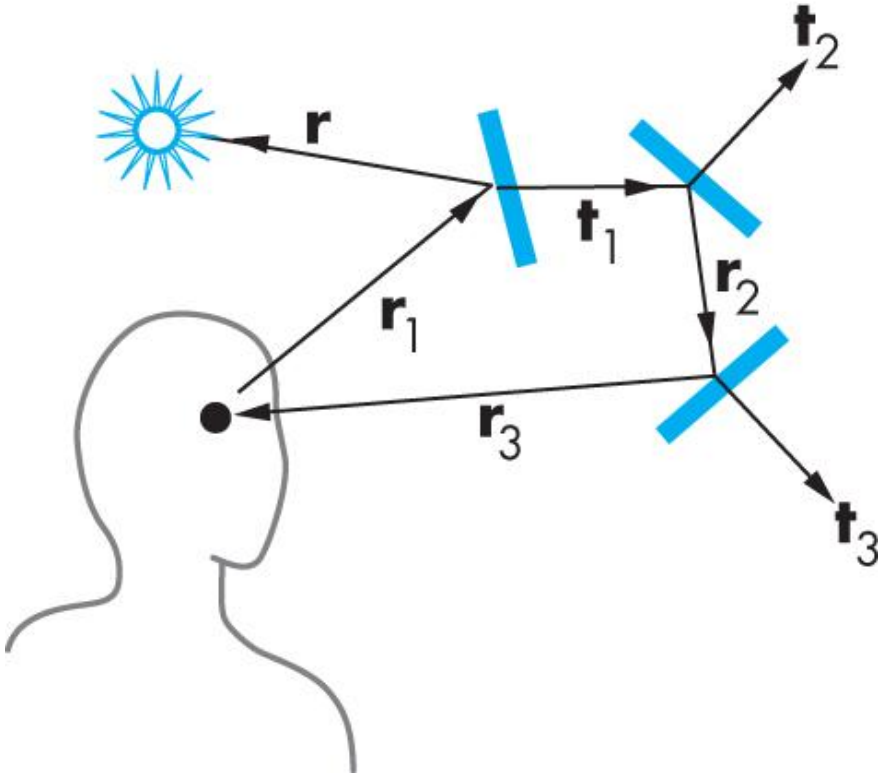
- Must follow shadow rays off reflecting or transmitting surfaces
- Process is recursive



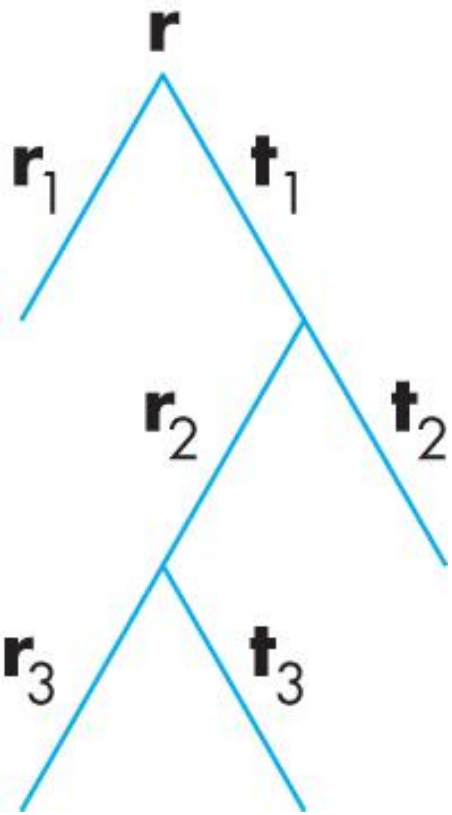
Reflection and Transmission



Ray Trees



Ray Tree



Diffuse Surfaces

- Theoretically the scattering at each point of intersection generates an infinite number of new rays that should be traced
- In practice, we only trace the transmitted and reflected rays but use the Phong model to compute shade at point of intersection
- Radiosity works best for perfectly diffuse (Lambertian) surfaces

Building a Ray Tracer

- Best expressed recursively
- Can remove recursion later
- Image based approach
  - For each ray …….
- Find intersection with closest surface
  - Need whole object database available
  - Complexity of calculation limits object types
- Compute lighting at surface
- Trace reflected and transmitted rays

When to stop

- Some light will be absorbed at each intersection
  - Track amount left
- Ignore rays that go off to infinity
  - Put large sphere around problem
- Count steps

More bounces

Fewer bounces

Recursive Ray Tracer

```
color c = trace(point p, vector d, int step)
{
  color local, reflected, transmitted;
  point q;
  normal n;
  if(step > max) return(background_color);
```

Recursive Ray Tracer

```
q = intersect(p, d, status);
if(status==light_source)
 return(light_source_color);
if(status==no_intersection)
 return(background_color);

n = normal(q);
r = reflect(q, n);
t = transmit(q,n);
```

Recursive Ray Tracer

```
local = phong(q, n, r);
reflected = trace(q, r, step+1);
transmitted = trace(q,t, step+1);

return(local+reflected+transmitted);
```

Computing Intersections

- Implicit Objects
  - Quadrics
- Planes
- Polyhedra
- Parametric Surfaces

Implicit Surfaces

Ray from $\mathbf{p}_0$ in direction $\mathbf{d}$

$$\mathbf{p}(t) = \mathbf{p}_0 + t\,\mathbf{d}$$

General implicit surface

$$f(\mathbf{p}) = 0$$

Solve scalar equation

$$f(\mathbf{p}(t)) = 0$$

General case requires numerical methods

Quadrics

General quadric can be written as

$$\mathbf{p}^T\mathbf{A}\mathbf{p} + \mathrm{b}^T\mathbf{p} + \mathrm{c} = 0$$

Substitute equation of ray

$$\mathbf{p}(\mathrm{t}) = \mathbf{p}_0 + \mathrm{t}\,\mathbf{d}$$

to get quadratic equation

Sphere

$$(\mathbf{p} - \mathbf{p}_c) \bullet (\mathbf{p} - \mathbf{p}_c) - r^2 = 0$$

$$\mathbf{p}(t) = \mathbf{p}_0 + t\,\mathbf{d}$$

$$\mathbf{p}_0 \bullet \mathbf{p}_0\, t^2 + 2\,\mathbf{p}_0 \bullet (\mathbf{d} - \mathbf{p}_0)\, t + (\mathbf{d} - \mathbf{p}_0) \bullet (\mathbf{d} - \mathbf{p}_0) - r^2 = 0$$

Planes

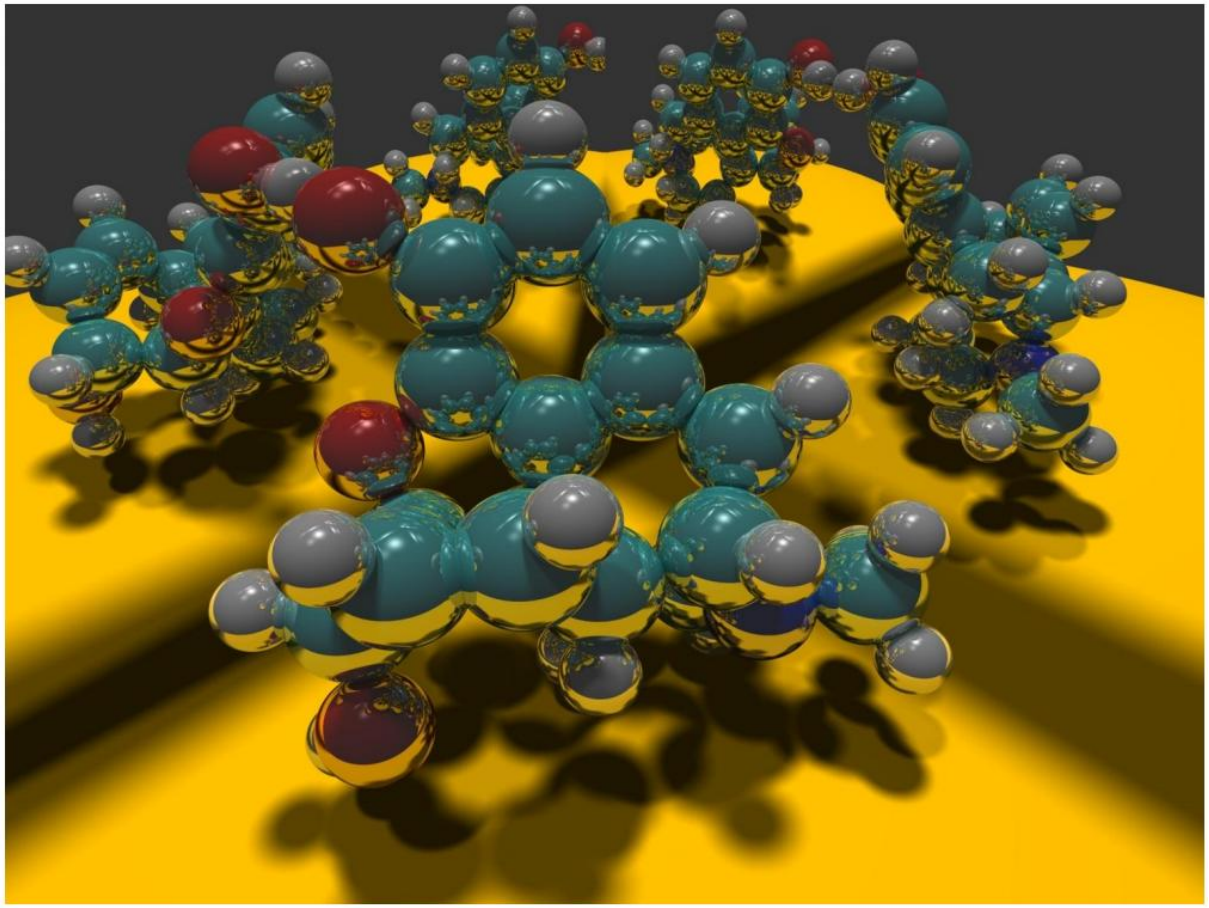
$$\mathbf{p} \cdot \mathbf{n} + c = 0$$

$$\mathbf{p}(t) = \mathbf{p}_0 + t\,\mathbf{d}$$

$$t = -(\mathbf{p}_0 \cdot \mathbf{n} + c)/\,\mathbf{d} \cdot \mathbf{n}$$

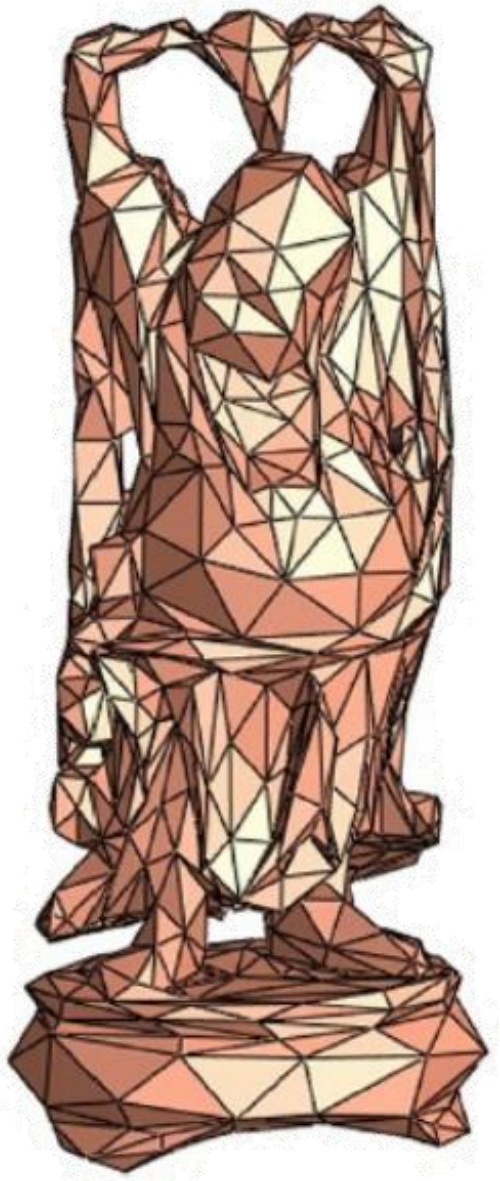
Implicit Surface Ray Tracing Example

- Paul Heckbert 1983



Polyhedra

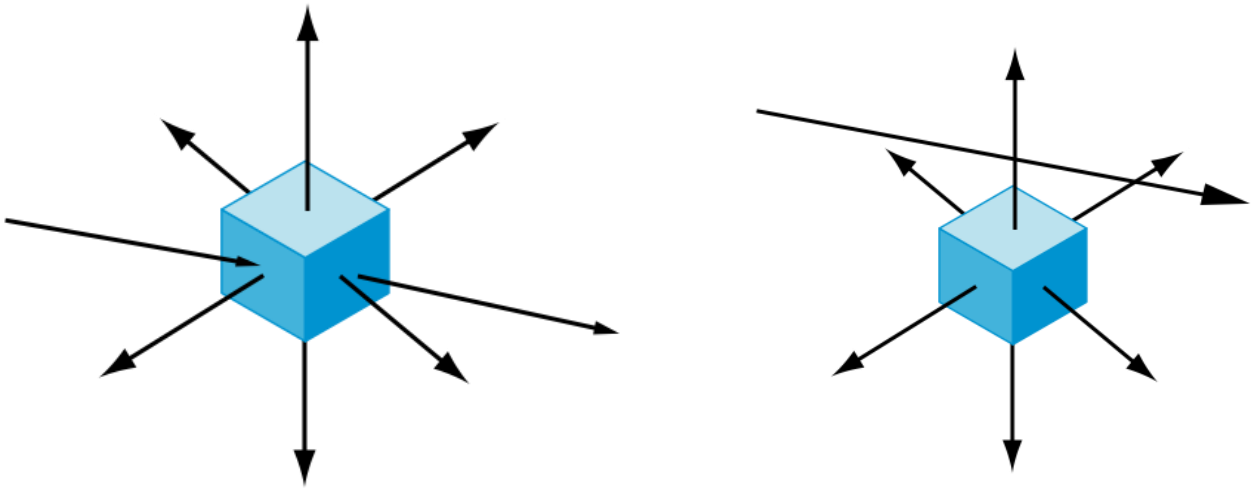
- Generally we want to intersect with closed objects such as polygons and polyhedra rather than planes
- Hence we have to worry about inside/outside testing
- For convex objects such as polyhedra there are some fast tests



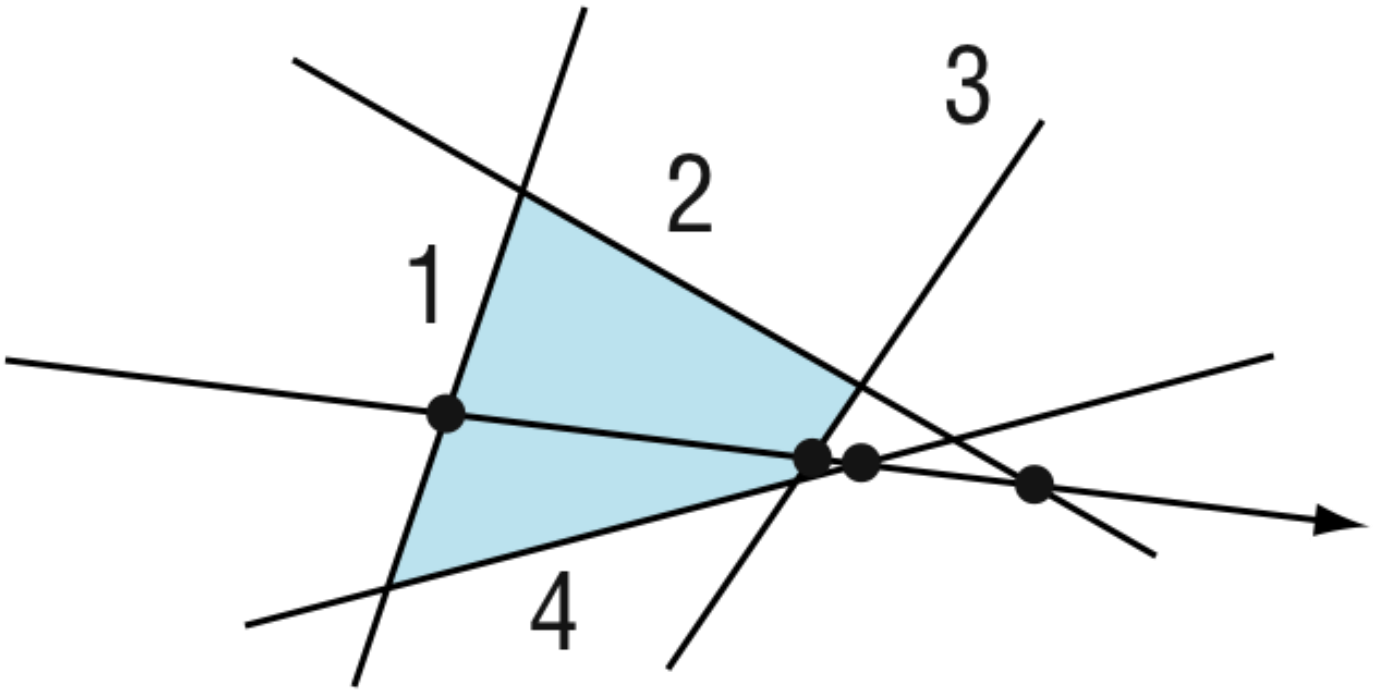
Ray Tracing Polyhedra

- If ray enters an object, it must enter a front facing polygon and leave a back facing polygon
- Polyhedron is formed by intersection of planes
- Ray enters at furthest intersection with front facing planes
- Ray leaves at closest intersection with back facing planes
- If entry is further away than exit, ray must miss the polyhedron

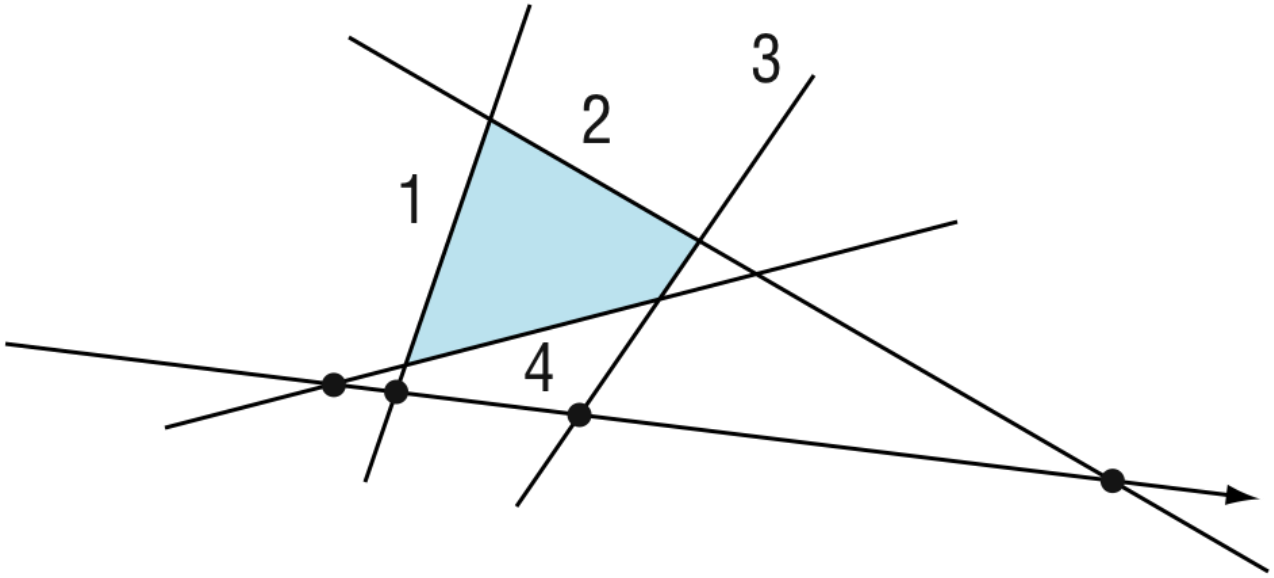
Ray Tracing Polyhedra



Ray Tracing a Polygon



Ray Tracing a Polygon



Algorithmic Complexity of Ray Tracing

- Time complexity
  - Rasterization: $O(b \cdot m)$
    - coherent!
  - Naïve ray tracing: $O(b \cdot p)$
  - Accelerated ray tracing: $O(\log b \cdot p)$
    - incoherent
- Key
  - b = # of objects in the scene
  - p = # of pixels in the scene
  - m = average # of pixels per object

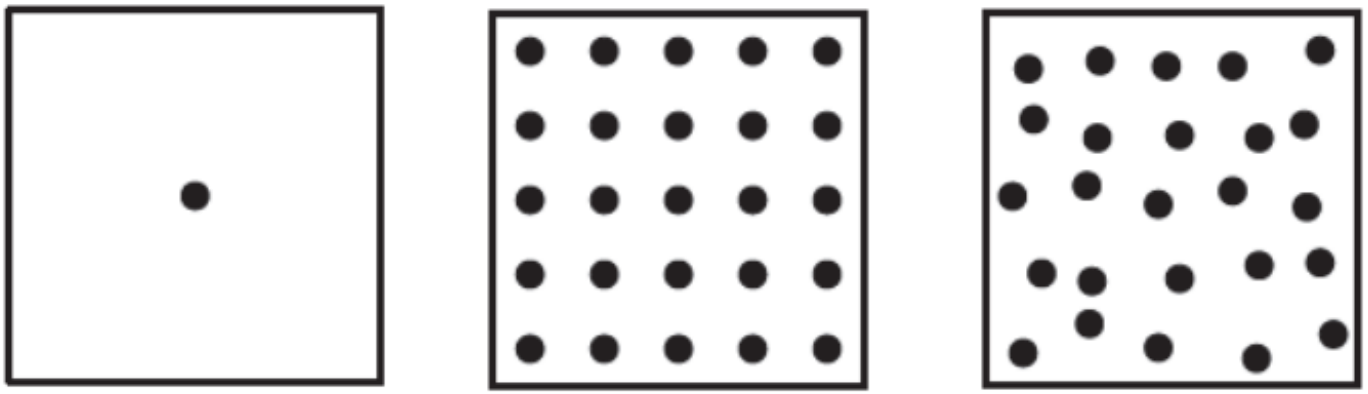
Real-time Ray Tracing

- GPUs doing ray tracing
  - General purpose compute on GPUs

[Optix, NVIDIA, SIGGRAPH 2009]

Ray Tracing Quality

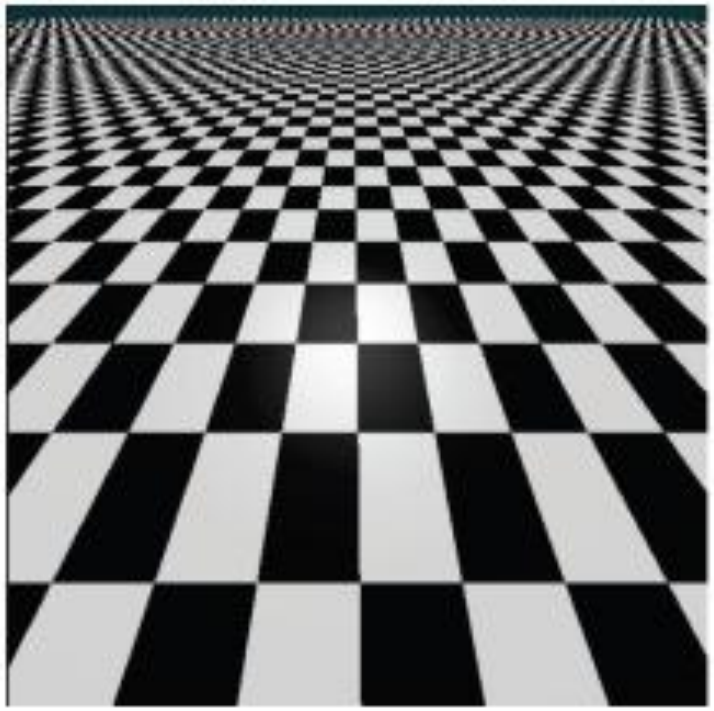
- Aliasing is a problem in ray tracing
  - Spatial and temporal
- Spatial anti-aliasing
  - Average over several samples per pixel



1 sample | 5x5 grid | 5x5 jittered grid

Distribution Ray Tracing

- Spatial anti-aliasing

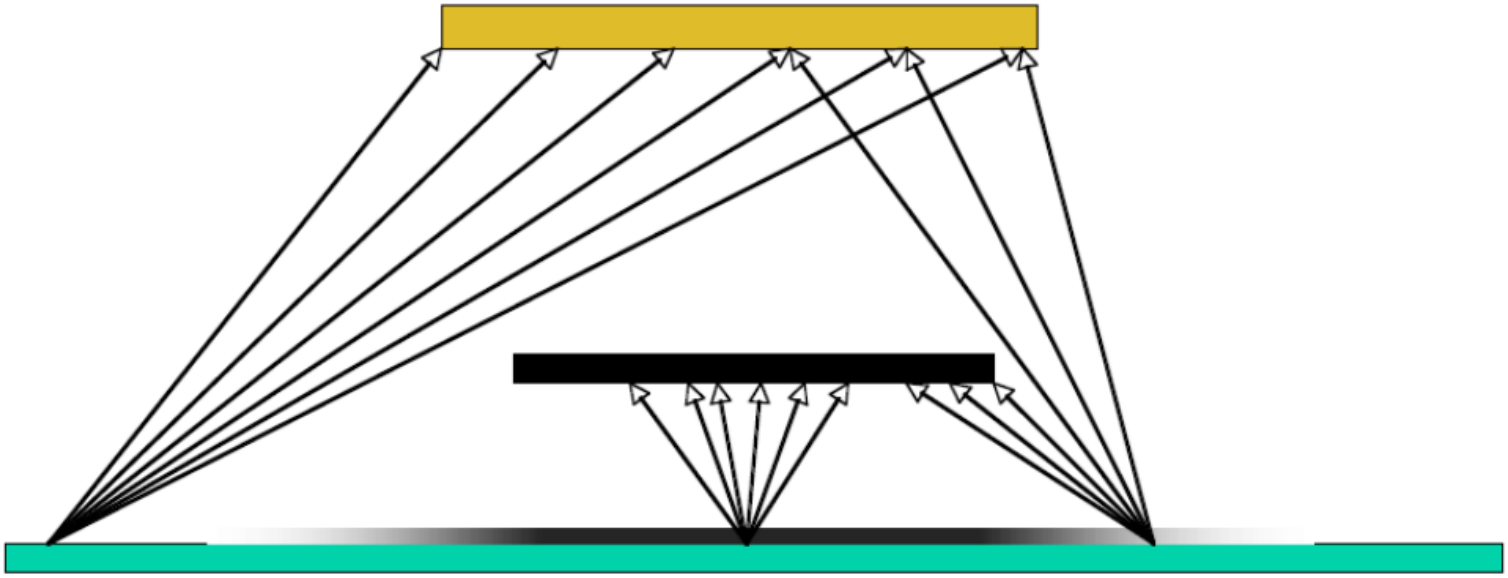


1 sample per pixel

multiple samples per pixel

Distribution Ray Tracing

- Soft shadows
  - Distribute shadow rays over light source region



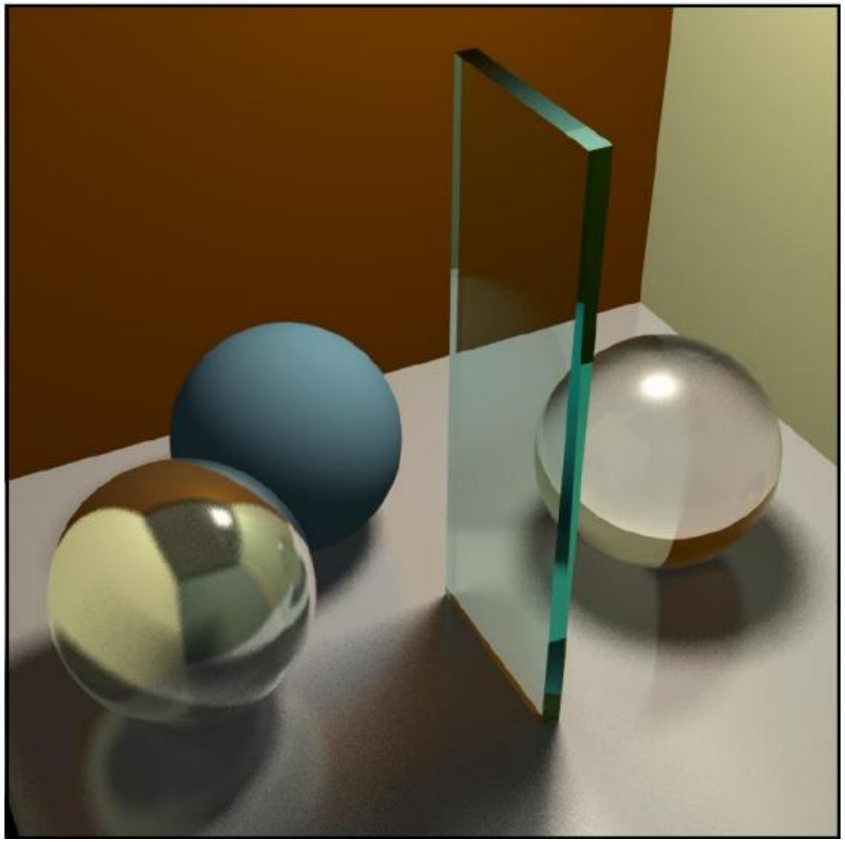
All shadow rays hit light source, fully illuminated

No shadow rays hit light source, fully shadowed

Some shadow rays hit light source, partially shadowed

Soft Shadows Example

- Multiple ray tracing effects
  - Including soft shadows



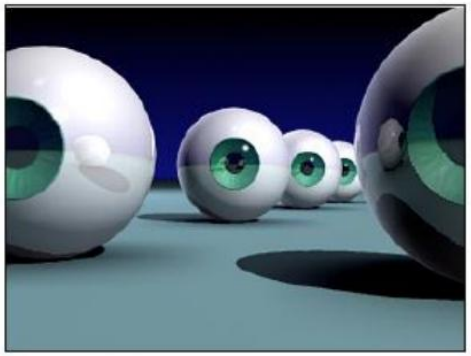
Distribution Ray Tracing

- Motion blur
  - Distribute rays over time

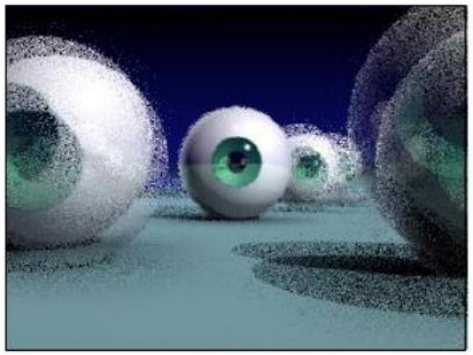
Pool Balls
Tom Porter
RenderMan

Distribution Ray Tracing

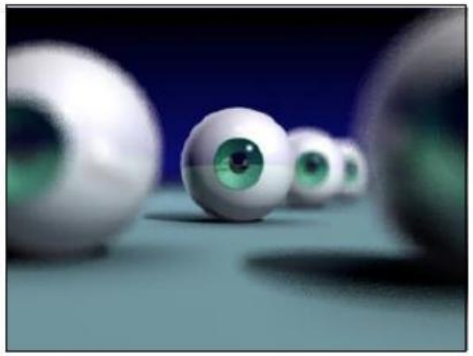
- Depth of field
  - Distribute rays across a discrete camera aperture



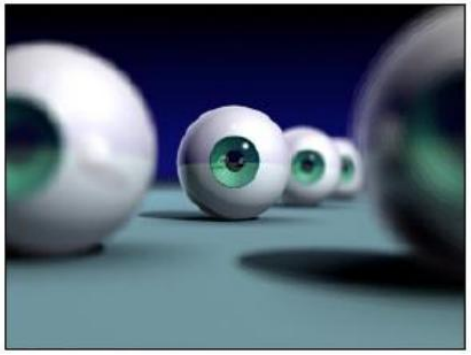
No depth-of-field



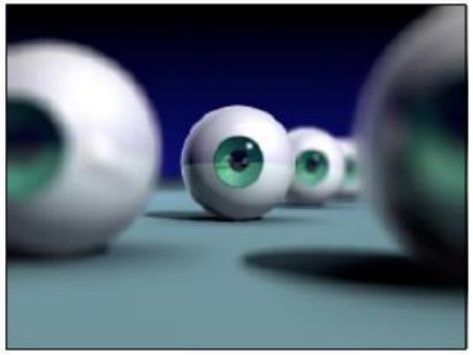
Jittered depth-of-field



More rays



More images for depth-of-field



Even more rays

Next Class

- Next lecture
  - Global illumination
  - *How is the complex interaction of light within a scene realistically modeled?*
- **Reading:** Chapter 11, 559-568
- **Project 3**
  - Due tomorrow—Wednesday, April 18, 2012
- *Next up: Project 4 is a simple ray tracer*
  - *Due May 2, 2012*

Credits

- Ed Angel and Dave Shreiner: *Interactive Computer Graphics* 6E © Addison-Wesley (2012)
- Pat Hanrahan
- Paul Heckbert