SCHOOL of ELECTRICAL ENGINEERING & COMPUTING
FACULTY of ENGINEERING & BUILT ENVIRONMENT
The UNIVERSITY of NEWCASTLE

Comp3320/6370 Computer Graphics

Semester 2, 2018

Ray Tracing

COMP3320/6370 Computer Graphics:
Ray Tracing

Mostly based on the book by Peter Shirley

(lecture notes partially prepared by Steven Nicklin)

Setting the Scene

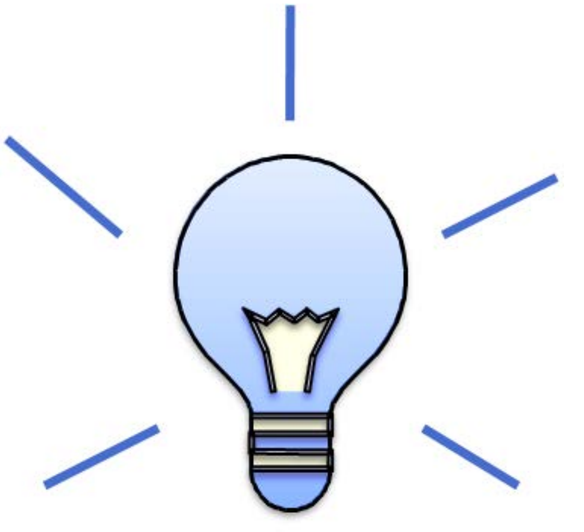
- The scene is a collection of objects, and light sources that are to be viewed via a camera.
- The scene is arranged in a world, or world space.
- The world has a width, height and depth in which the objects and light sources can be arranged.

Objects

- A thing that will be displayed in the scene.
- Objects can consist of shapes which can be described mathematically.
- Can be gas, liquid or solid.
- All objects have some kind of texture, describing its colour, shininess and other properties.

Light Sources

- A light source has three important properties
  - Position
  - Colour
  - Brightness
- Without lighting nothing in the scene is visible.

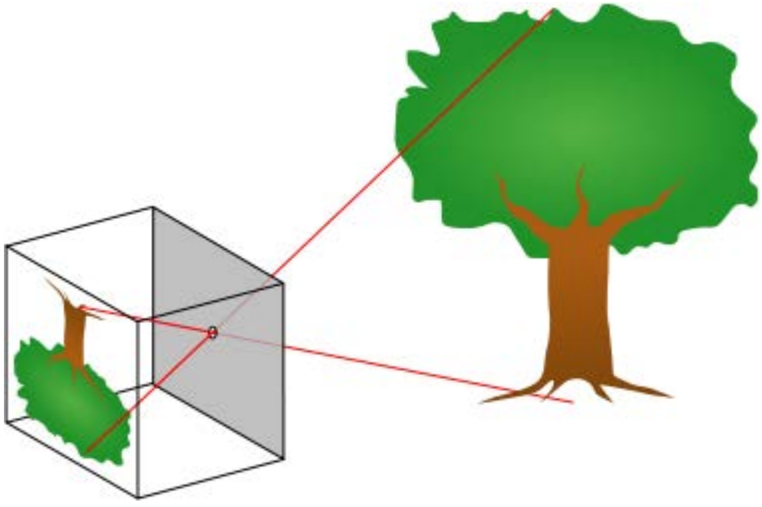


Camera

- The camera represents the point at which we are viewing the scene.
- The image produced is the scene as viewed through the camera.

Pinhole Camera

- The small hole restricts the amount of light that enters the camera.
- Each part of the film (in our case each pixel) is only exposed to a small part of the scene.
- Only one possible path for the light to take to reach each pixel.



What is Ray Tracing?

- Rendering method produced by tracing the path of light from the viewing point to the light source/s (i.e. backwards).
- Current method first developed in mid 1980’s
- Simple, yet powerful rendering method
- Many extensions have been developed to create photo realistic images

Why is Ray Tracing Important?

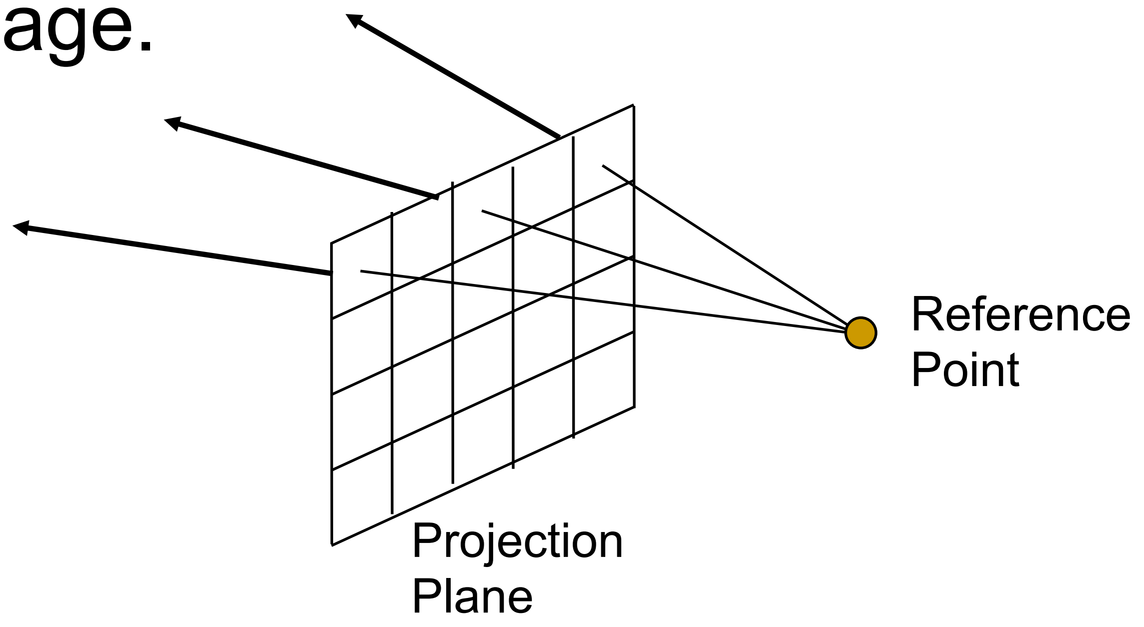
- Aims at producing photorealistic images in 3D Computer Graphics.
- The animated films *Ice Age*, *Ice Age 2*, *Robots*, *Bunny*, and *The Cathedral* were fully ray traced.
- Ray tracing was also used in *Happy Feet* and *Cars*.

Basic Premises of Ray Tracing

- Light rays travel in straight lines
- Light rays do not interfere with each other when they cross
- Light travels from the light source to the eye, but the physics are invariant under path reversal (!)

General Method

- Primary rays are produced by projecting from a reference point through a projection plane that has been divided into pixels representing the image.



Ray Casting

- Uses the same principles as raytracing, however does not generate new rays following an intersection.

General Method

- Rays are tested for intersection with objects in scene, the closest intersection is used
- Effect of intersection on ray is determined
- New rays are created if necessary
- Process is then repeated for new rays

Intersection Methods

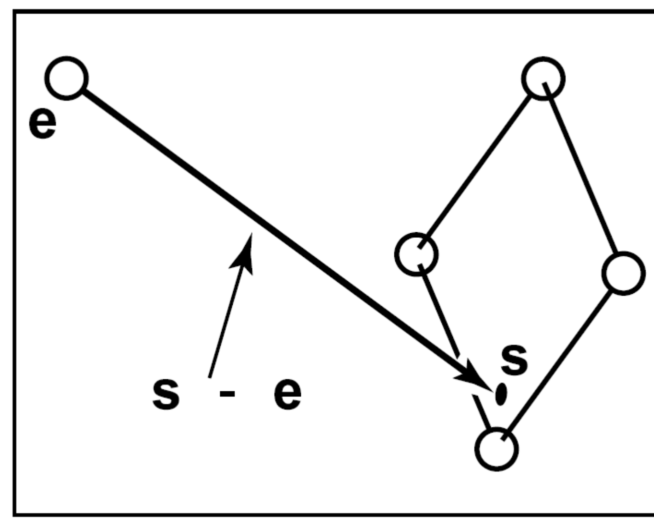
- Many possible intersections and collisions are possible: between objects, light rays and various objects, clipping planes and objects, etc..
- Each situation has a method for fast calculation.
- In our exercises and course we restrict ourselves to a simple case and understanding of the underlying concept.
- But there is much more and if you have a look at the following page you can get some idea and pointers to the different situations http://www.realtimerendering.com/intersections.html

Finding an intersection

- Rays can be described by a 3D parametric line

$$\mathbf{p}(t) = \mathbf{e} + t(\mathbf{s} - \mathbf{e})$$

- **e** = position of eye
- **s** = point ray passes through screen
- *t* = distance
- When an intersection is found, the value of *t* tells us distance from **e**



Finding an intersection: Sphere

- Given a ray $\mathbf{p}(t) = \mathbf{e} + t\mathbf{d}$ and a surface given by equation $f(\mathbf{p}) = 0$ we want to find the intersection
- Intersection occurs when the point on the ray satisfies the surface equation $f(\mathbf{p}(t)) = f(\mathbf{e} + t\mathbf{d}) = 0$
- A sphere with centre $\mathbf{c} = (c_x, c_y, c_z)$ and radius R can be represented as such an equation

$$(x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2 - R^2 = 0$$

- i.e. in vector form $(\mathbf{p} - \mathbf{c}) \cdot (\mathbf{p} - \mathbf{c}) - R^2 = 0$

Finding an intersection: Sphere cont.

- Substituting $\mathbf{p} = \mathbf{e} + t\mathbf{d}$ gives

$$(\mathbf{e}+t\mathbf{d}-\mathbf{c})\cdot(\mathbf{e}+t\mathbf{d}-\mathbf{c})-R^2=0$$

- Rearranging gives quadratic equation

$$(\mathbf{d}\cdot\mathbf{d})t^2+2\mathbf{d}\cdot(\mathbf{e}-\mathbf{c})t+(\mathbf{e}-\mathbf{c})\cdot(\mathbf{e}-\mathbf{c})-R^2=0$$

- Solving the quadratic equation for t results in

$$t=\frac{-\mathbf{d}\cdot(\mathbf{e}-\mathbf{c})\pm\sqrt{(\mathbf{d}\cdot(\mathbf{e}-\mathbf{c}))^2-(\mathbf{d}\cdot\mathbf{d})((\mathbf{e}-\mathbf{c})\cdot(\mathbf{e}-\mathbf{c})-R^2)}}{(\mathbf{d}\cdot\mathbf{d})}$$

Finding an intersection: Sphere cont.

- If the discriminante is negative, roots are imaginary and there is no intersection
- If the discriminante is zero there is one unique solution and the ray touches the edge of the sphere at one point
- If the discriminante is positive, there are two solutions and ray passes through sphere

Interactions Between Rays and Objects

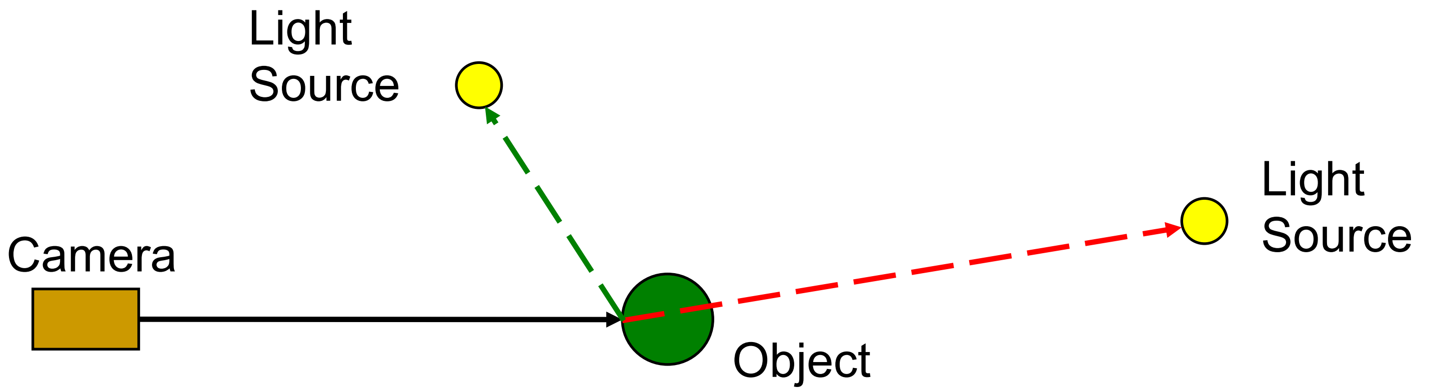
- After an intersection between a ray and an object, the ray can be affected by:
  - Shadows
  - Reflection
  - Refraction

Shadows

- Shadows are caused when a ray hits an object which has no line of sight to a light source
- The amount of shadow is determined by the light sources not in line of sight

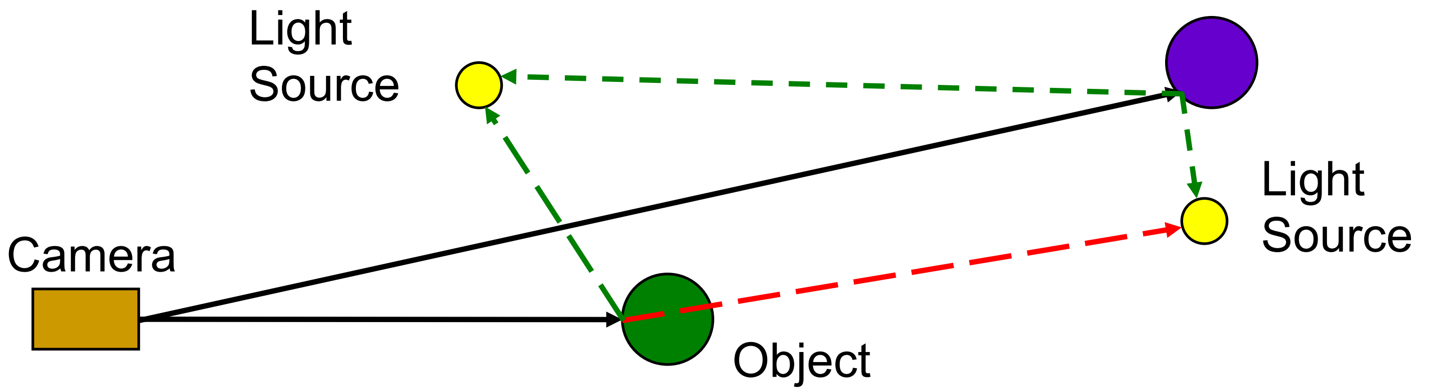
Shadows

- Shadows are found by sending rays towards the light source/s to determine line of sight
- If there is no line of sight to the light source, that point is in shadow with respect to that source.



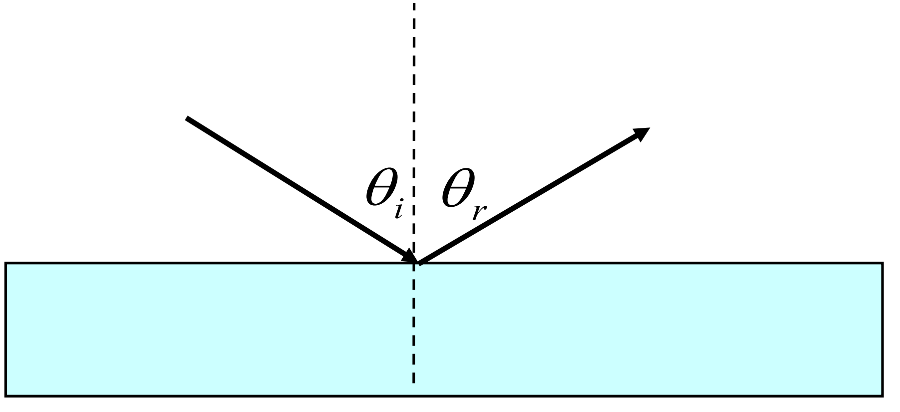
Shadows

- The lower the number of light sources acting on the point, the darker the shadow



Reflection

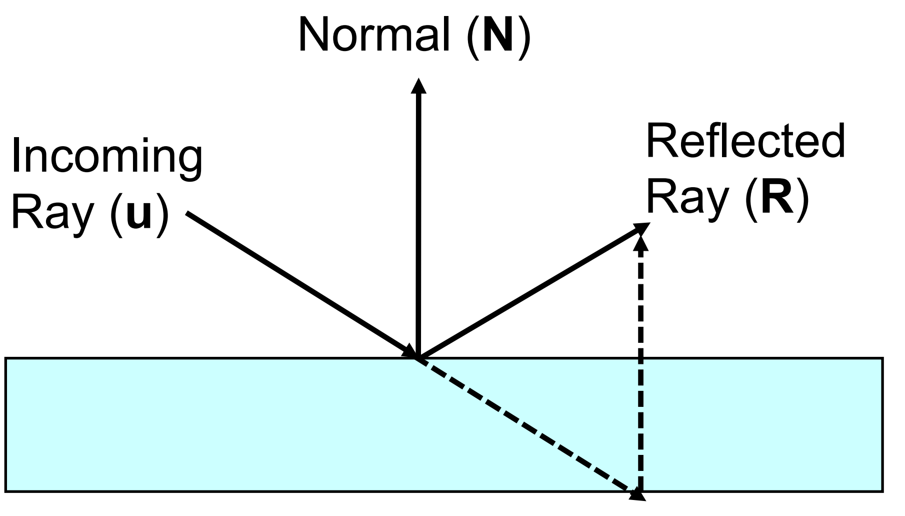
- Reflections occur when the ray hits a reflective object
- The angle of reflection equals the angle of incidence
- $\theta_r = \theta_i$



Reflection

- To find the reflected ray **R** from the incoming ray **u** and the unit normal vector **N**

$$\mathbf{R} = \mathbf{u} - (2\mathbf{u} \bullet \mathbf{N})\mathbf{N}$$

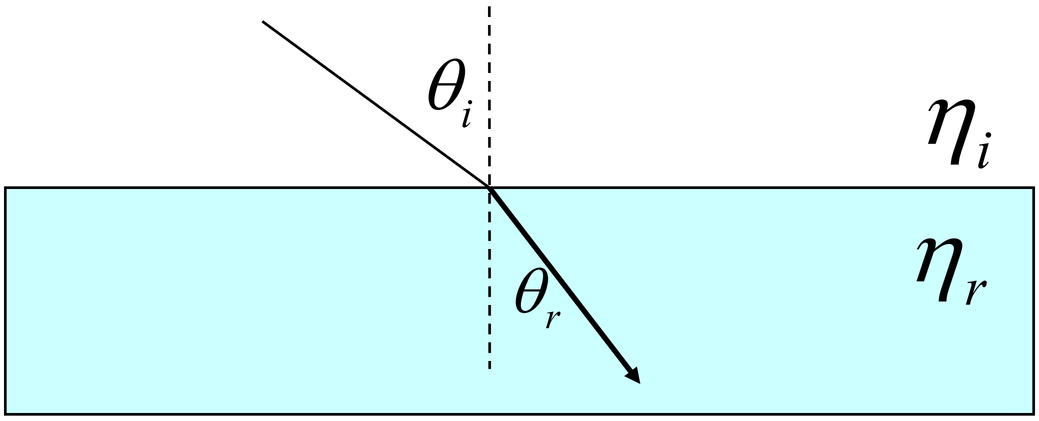


Refraction

- Light rays are *refracted* when they change mediums.
- Described by Snell's Law

$$\eta_i \sin\theta_i = \eta_r \sin\theta_r$$

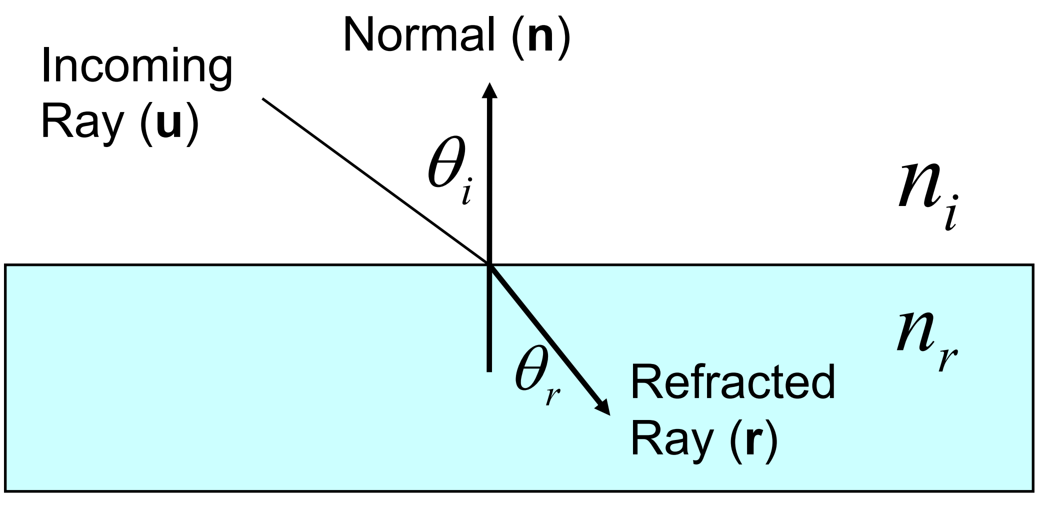
- η is given by a materials refractive index



Refraction

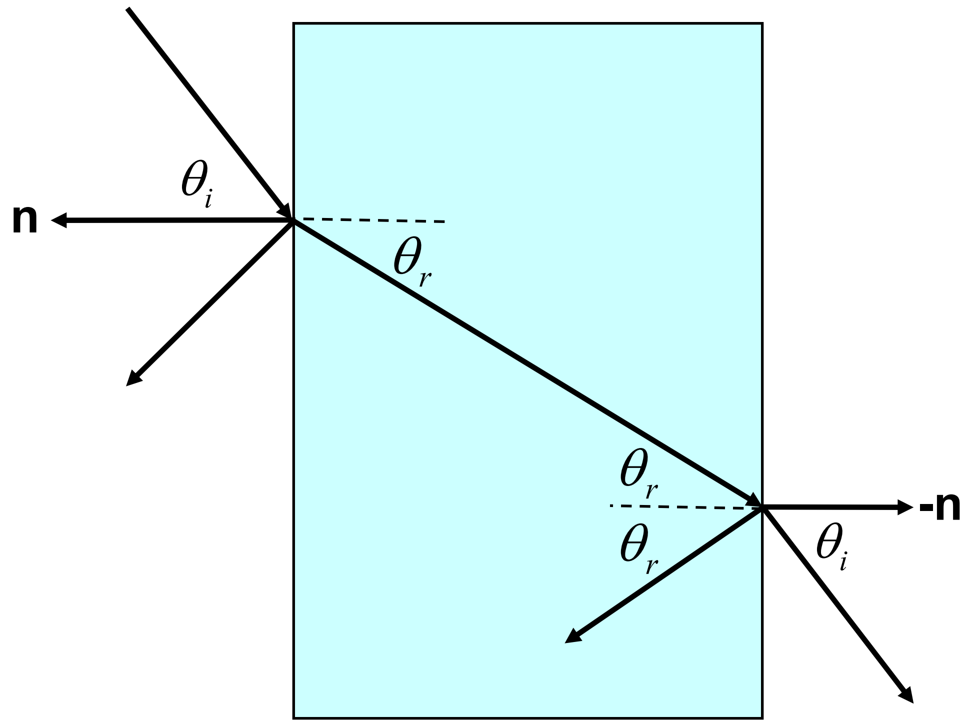
- We note (e.g. by looking at a physics book [not asked in exam]) that the refracted ray **r** (vector) is given by

$$\mathbf{r} = \frac{n_i\left(\mathbf{u} - \mathbf{n}(\mathbf{u}\cdot\mathbf{n})\right)}{n_r} - \mathbf{n}\sqrt{1 - \frac{n_i^2\left(1 - (\mathbf{u}\cdot\mathbf{n})^2\right)}{n_r^2}}$$



Refraction

- Refracted rays can pass through the object causing further refraction and internal reflections as they leave the medium



Refraction

- Since light is both refracted and reflected we need to determine how the total light is split
- The reflectivity varies depending on incident angle according to the *Fresnel Equations*
- This can be approximated using the *Schlick approximation* [formula not asked in exam].

$$R(\theta) = R_0 + (1 - R_0)(1 - \cos\theta)^5 \qquad R_0 = \left(\frac{n_r - 1}{n_r + 1}\right)^2$$

Tracing a Ray – Basic Algorithm

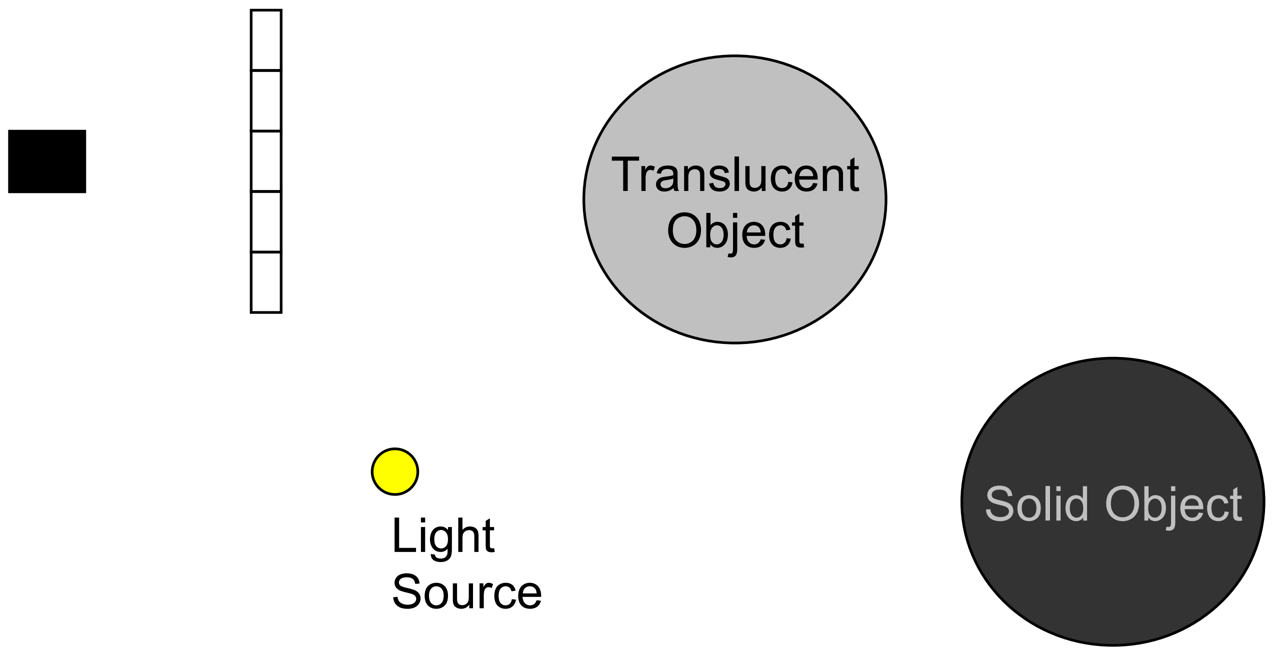
- For each pixel
  - Form primary ray
  - Find closest intersection
  - If intersection found
    - Shade(depth+1,final)   // compute pixel colour based on the results of ray intersections
  - Put final shade in pixel

- Shade(depth,**rtn**shade)   // **rtn** = reflection, transparent, normal
  - Form shadow ray
  - Find intersection
  - If reflective
    - Form reflected ray
    - Find closest intersect
    - Shade(depth+1,reflshade)
  - If transparent
    - Form refract ray
    - Find closest intersect
    - Shade(depth+1,refrshade)
  - Compute **rtn**shade

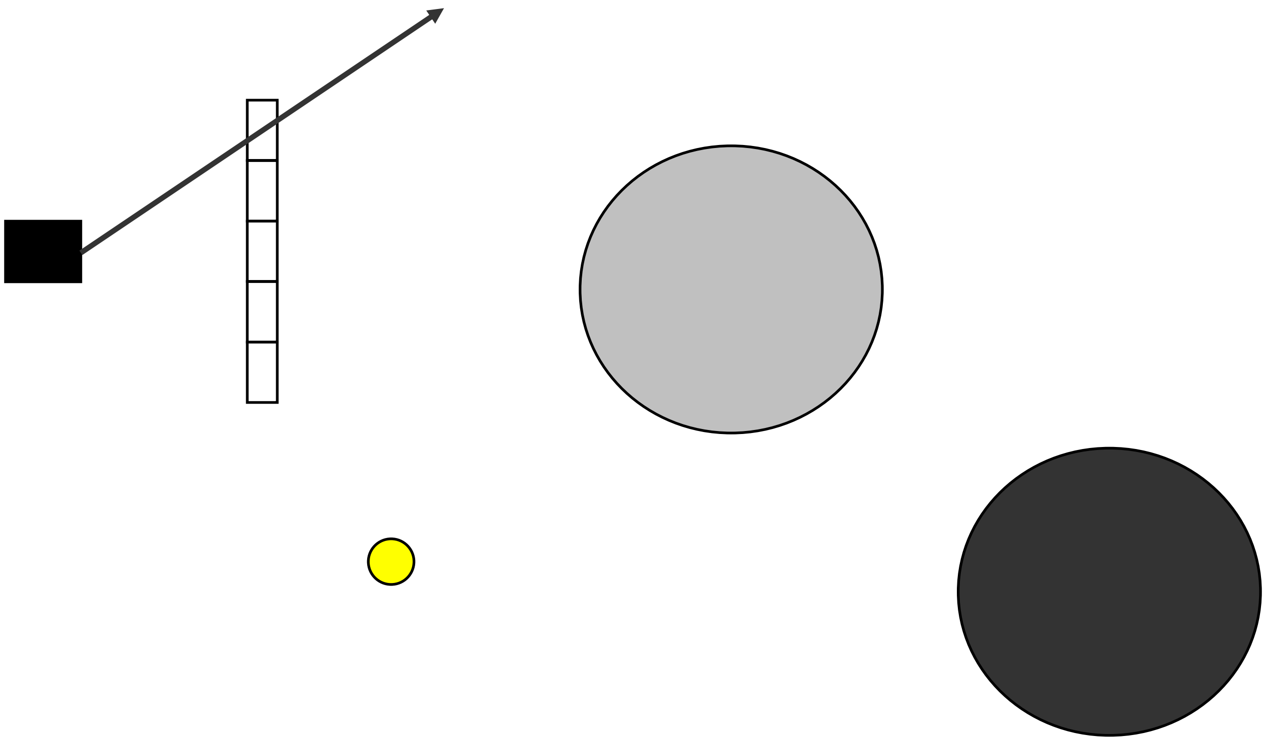
Tracing a Ray – Basic Process

- Recursive process
- Need a limit on number of rays formed or it could become infinite!

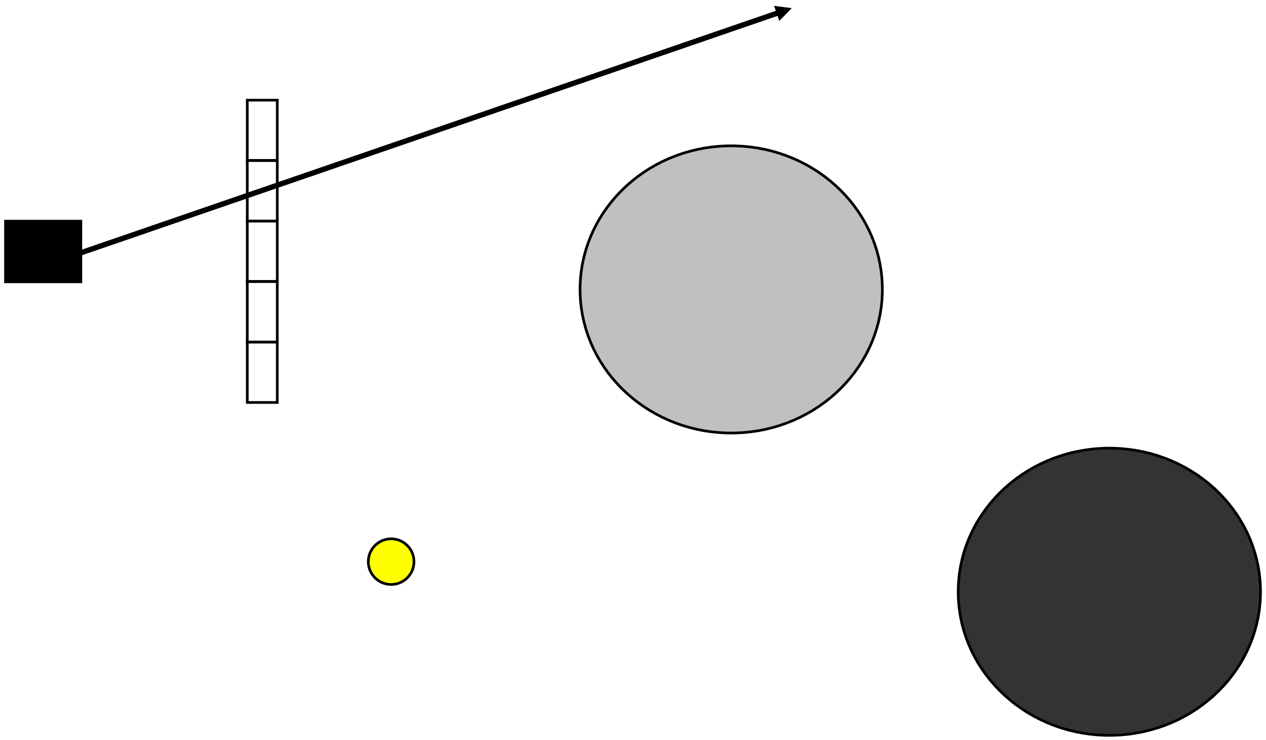
Tracing a Ray – 2D Example



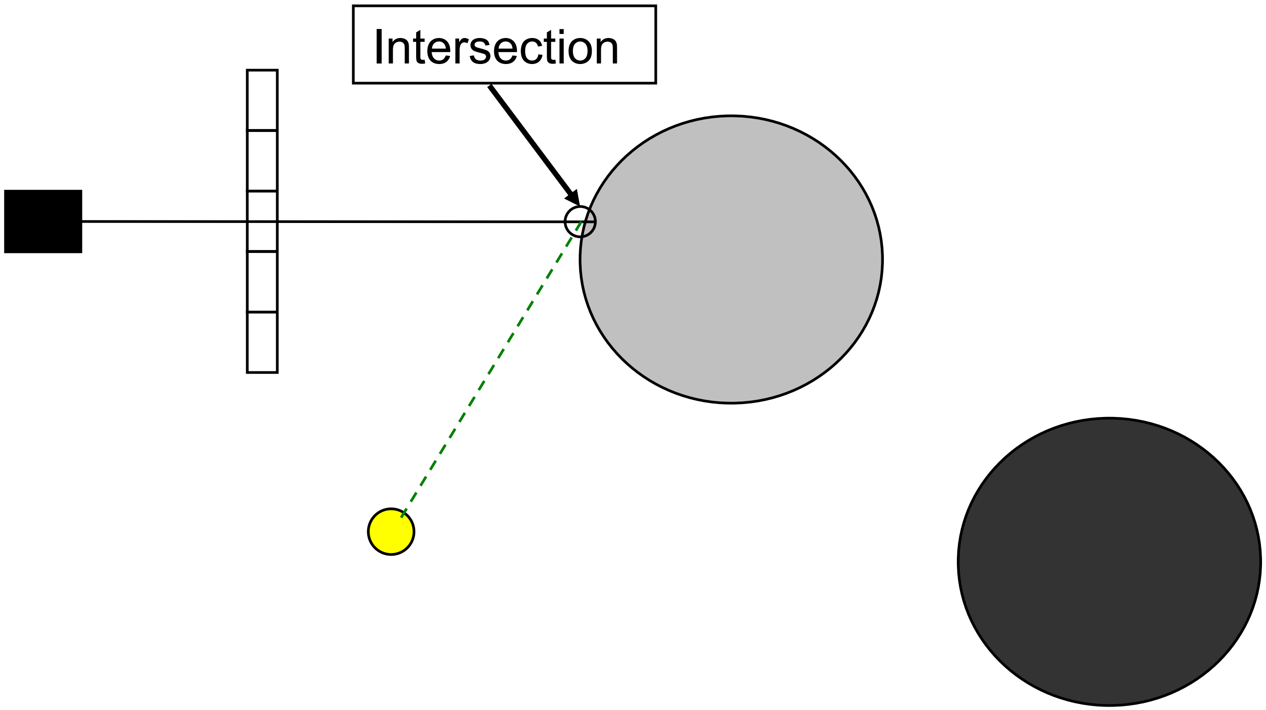
Tracing a Ray – 2D Example



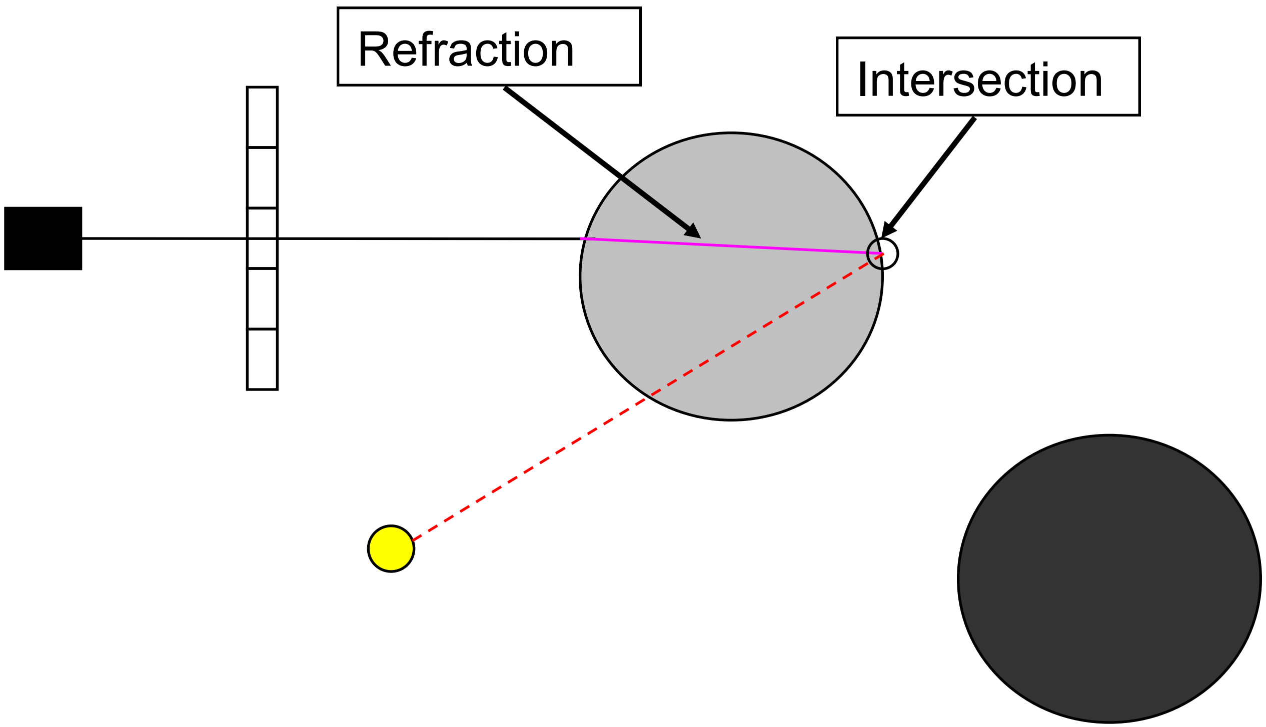
Tracing a Ray – 2D Example



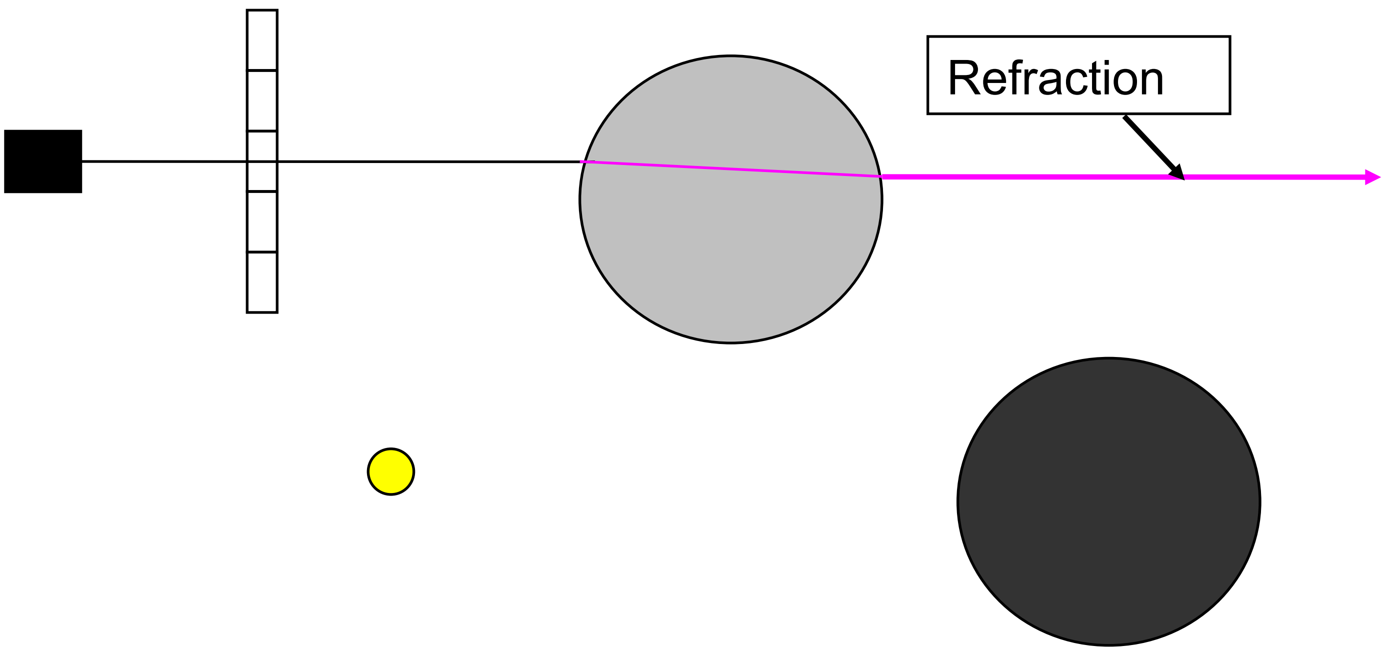
Tracing a Ray – 2D Example



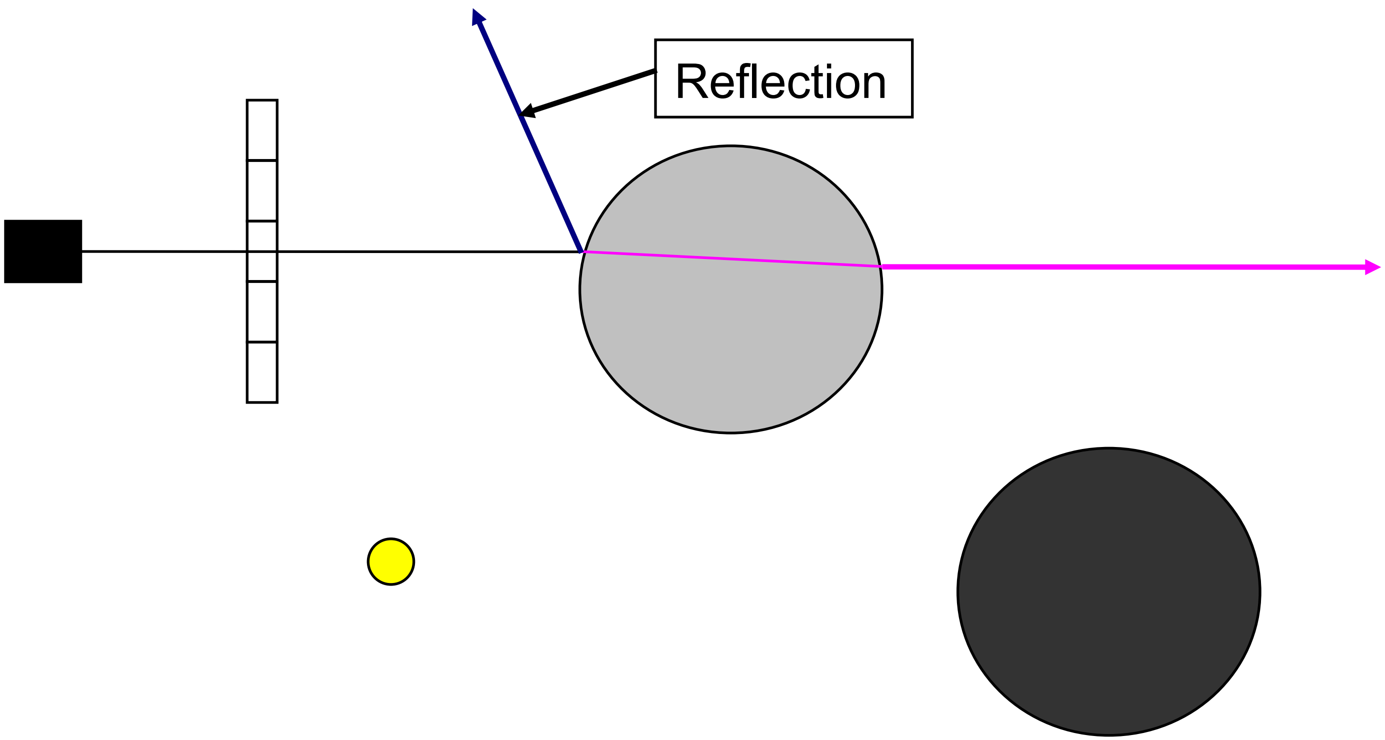
Tracing a Ray – 2D Example



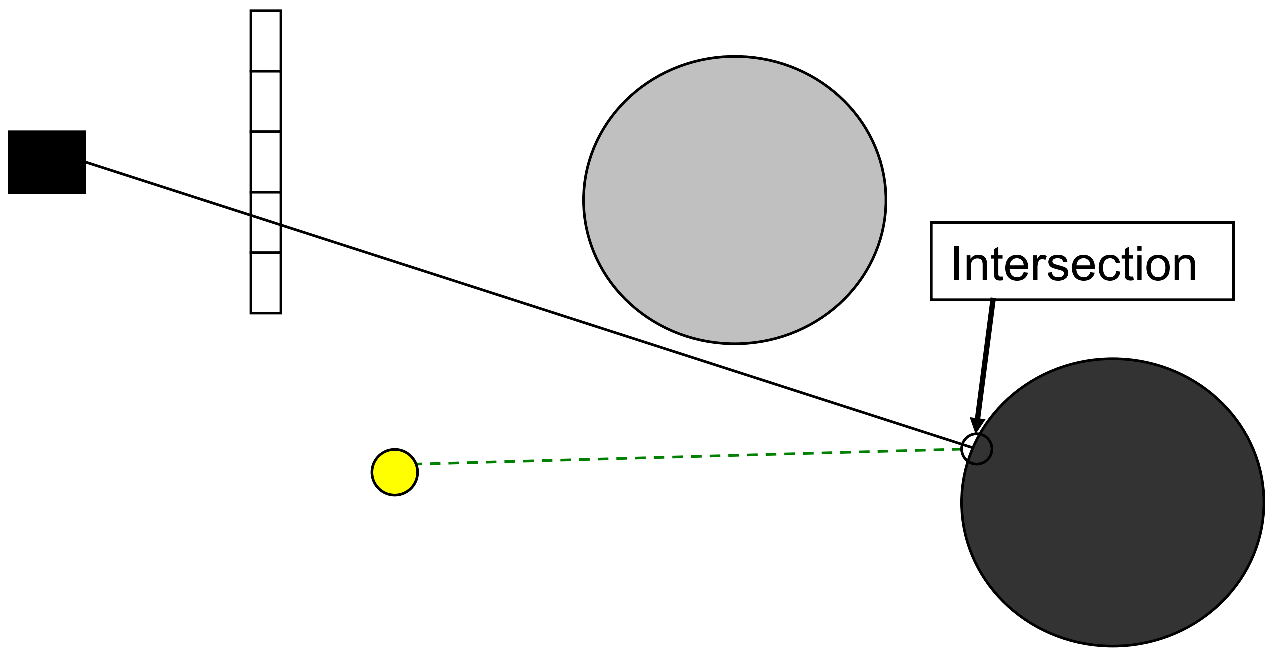
Tracing a Ray – 2D Example



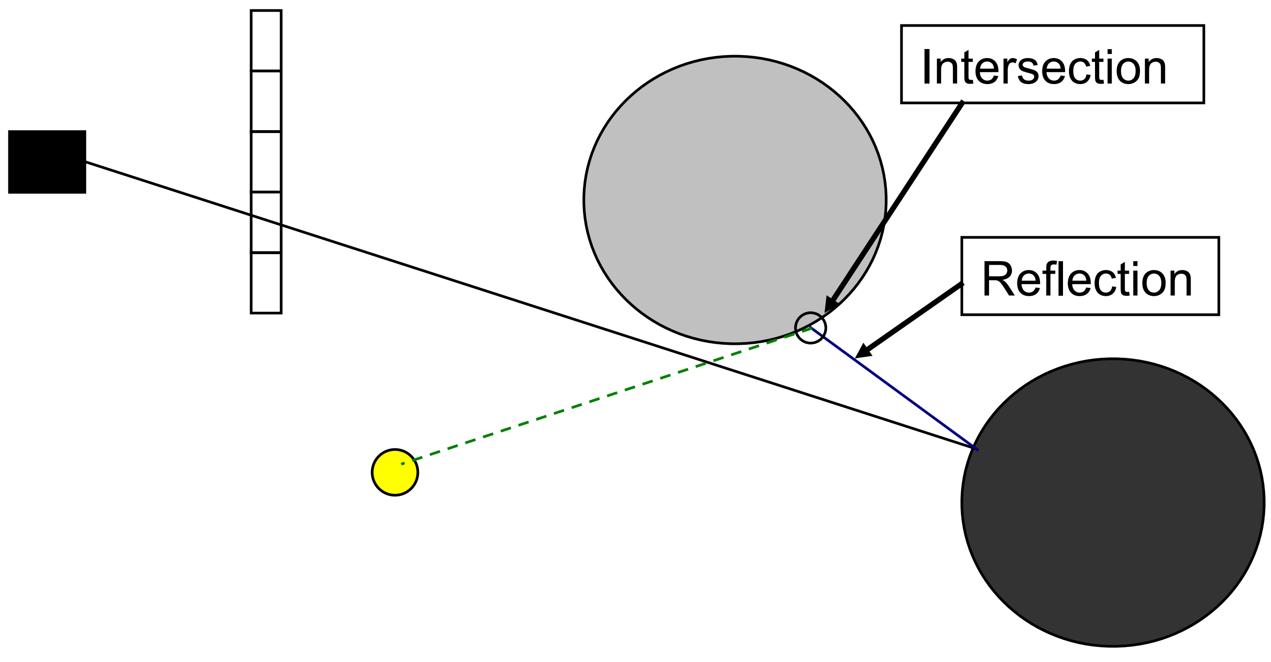
Tracing a Ray – 2D Example



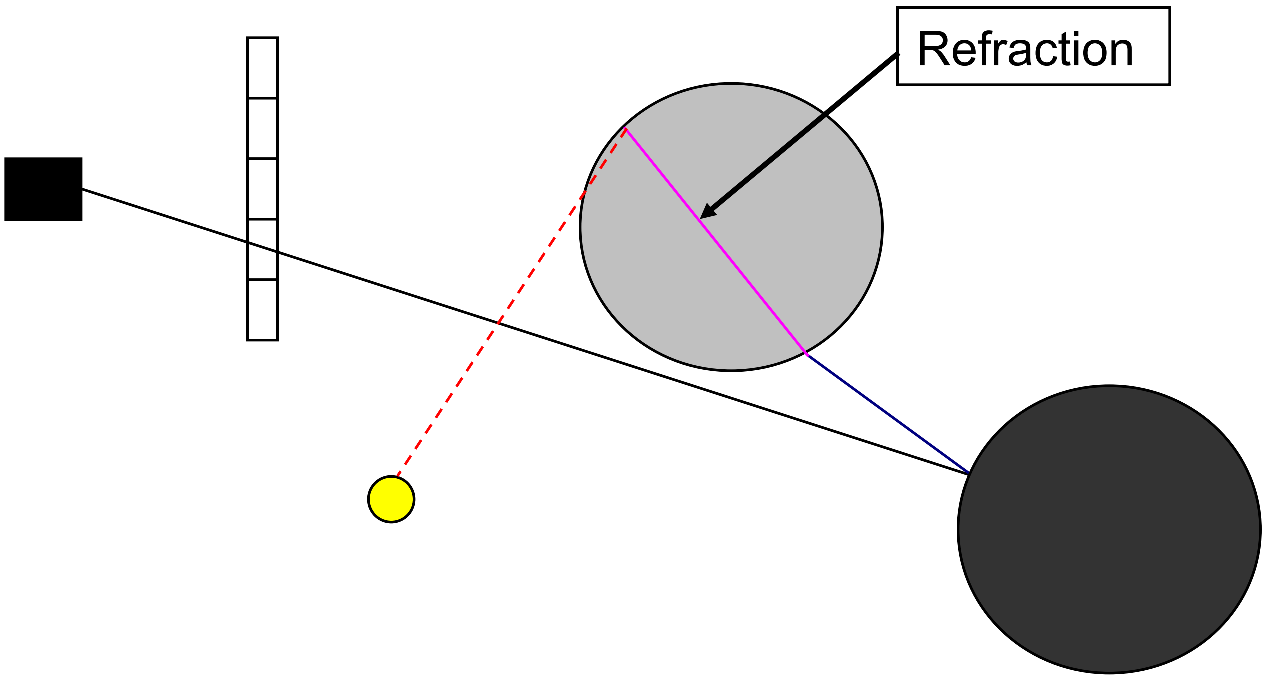
Tracing a Ray – 2D Example



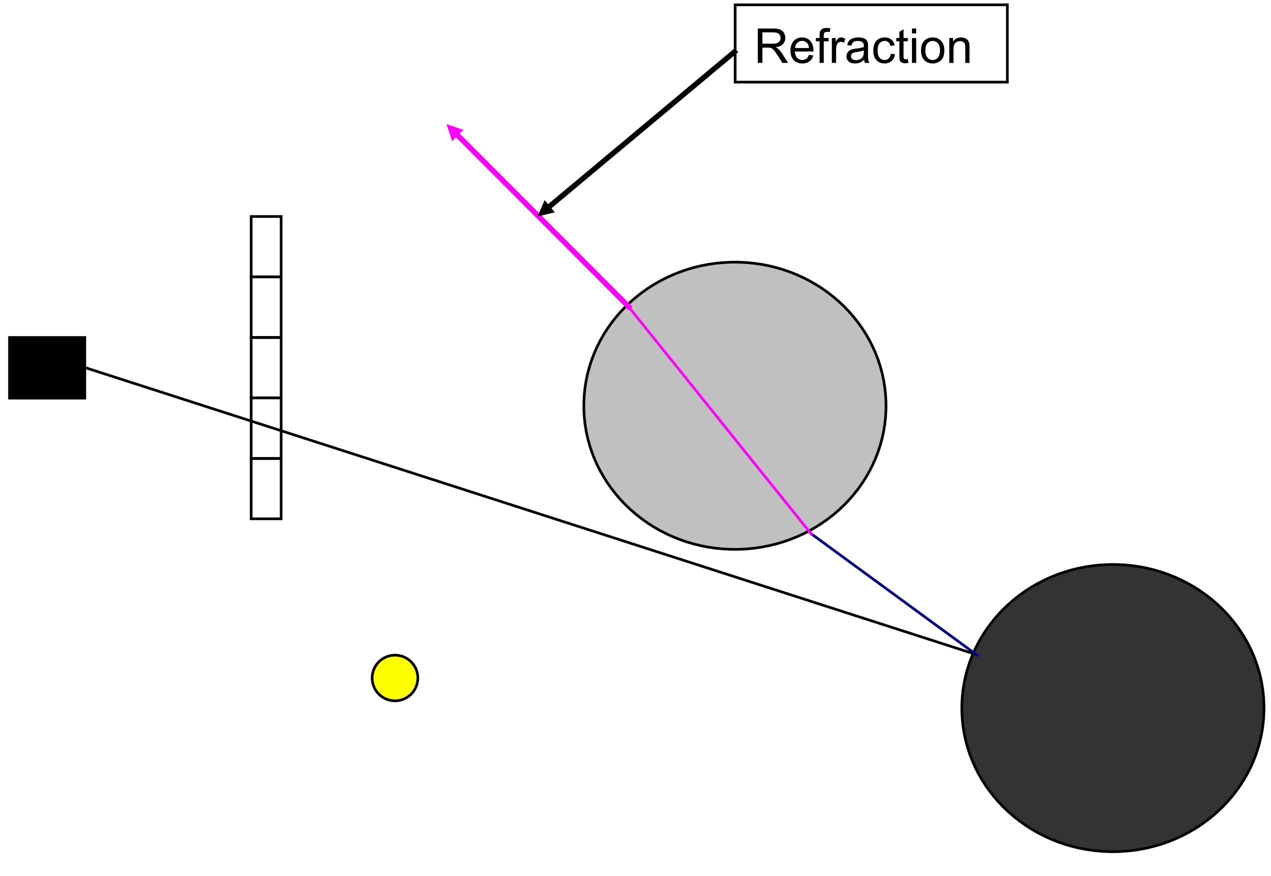
Tracing a Ray – 2D Example



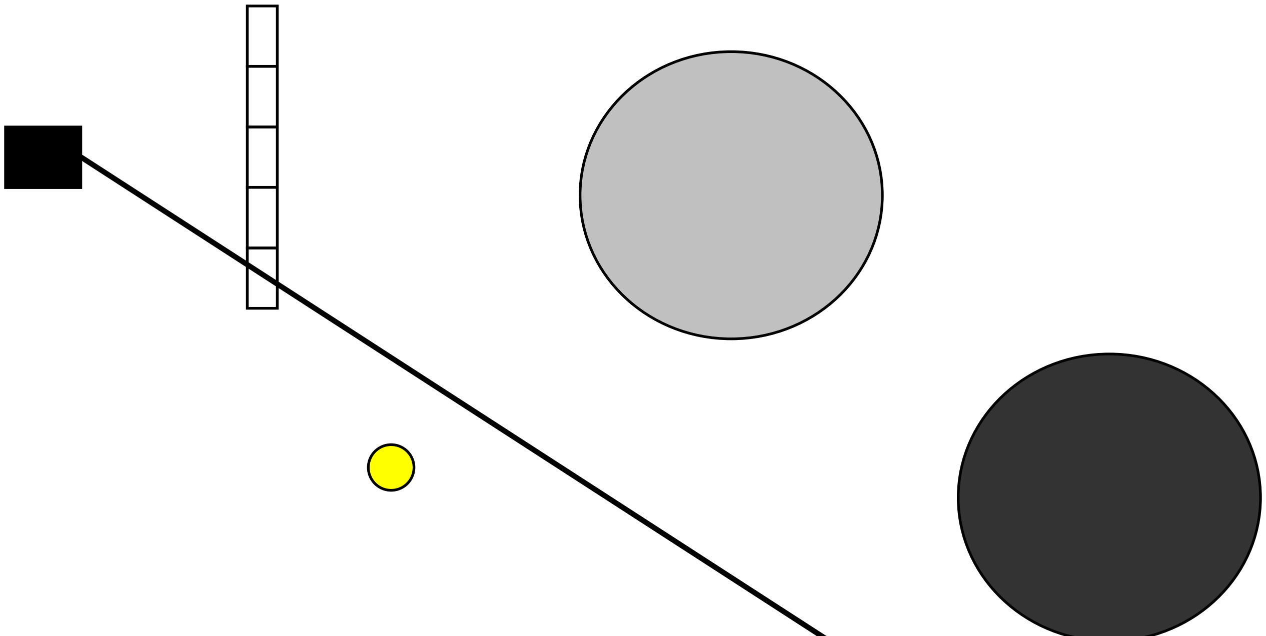
Tracing a Ray – 2D Example



Tracing a Ray – 2D Example



Tracing a Ray – 2D Example



Ray-Tracing in Practice

- Often used for realistic computer generated images
- Advanced version used e.g. in the Pixar film "*Cars*", due to the high level of reflections in the film
- Its use is limited by high computation requirements. This is becoming less of an issue with modern processors and methods.

Advantages of Ray Tracing

- Technique inherently produces:
  - Global reflections
  - Detects visible surfaces
  - Shadows
  - Transparency effects
  - Perspective-projection views

Limitations

- Images are often too precise, with sharp shadows, sharp edges, all objects in focus, and mirror-like reflections
- Many other effects are missing
- Combining with more advanced methods leads to best results
- Takes a large amount of computing resources

Ray-Tracing in Practice

Why is Traditional Ray-Tracing so Processor Intensive?

- Image at 1024x768 = 786,432 Pixels
- One primary ray must be sent for each of the pixels
- Each pixel may have multiple rays occurring from intersections
- Every ray must be compared with all objects in the scene for intersections
- **Intersection calculations** can be expensive and use up to 95% of processing time !!

Distributed Ray Tracing

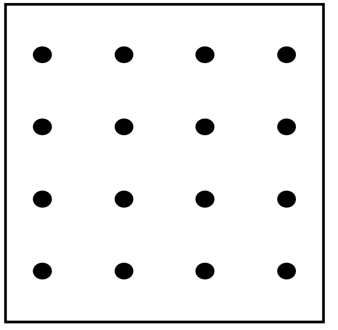
- Images produced by simple Ray-Tracing are often too “clean”
- Distributed ray tracing can be used to produce the following effects:
  - Anti-Aliasing
  - Soft Shadows
  - Depth of Field
  - Glossy Reflections
  - Motion Blur

Anti-Aliasing

- Simple way to perform anti-aliasing is to calculate the average colour value over the area of the pixel
- To do anti-aliasing with ray tracing we do this by creating multiple rays for each pixel and computing the average from these
- Where should rays be formed within the pixel?

Anti-Aliasing

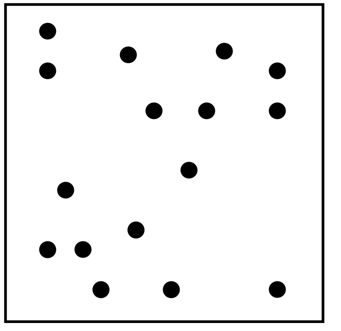
- Regular Grid
  - Produces same result as scaling down a higher resolution ray-trace
  - Regular pattern can create regular artefacts such as Moire patterns



Regular Samples

Anti-Aliasing

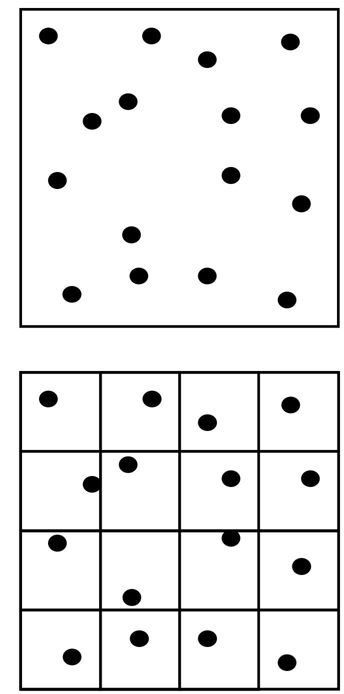
- Random
  - Samples randomly placed within pixel
  - Noise from randomness of sample selection can be a problem unless a lot of samples are used



Random Samples

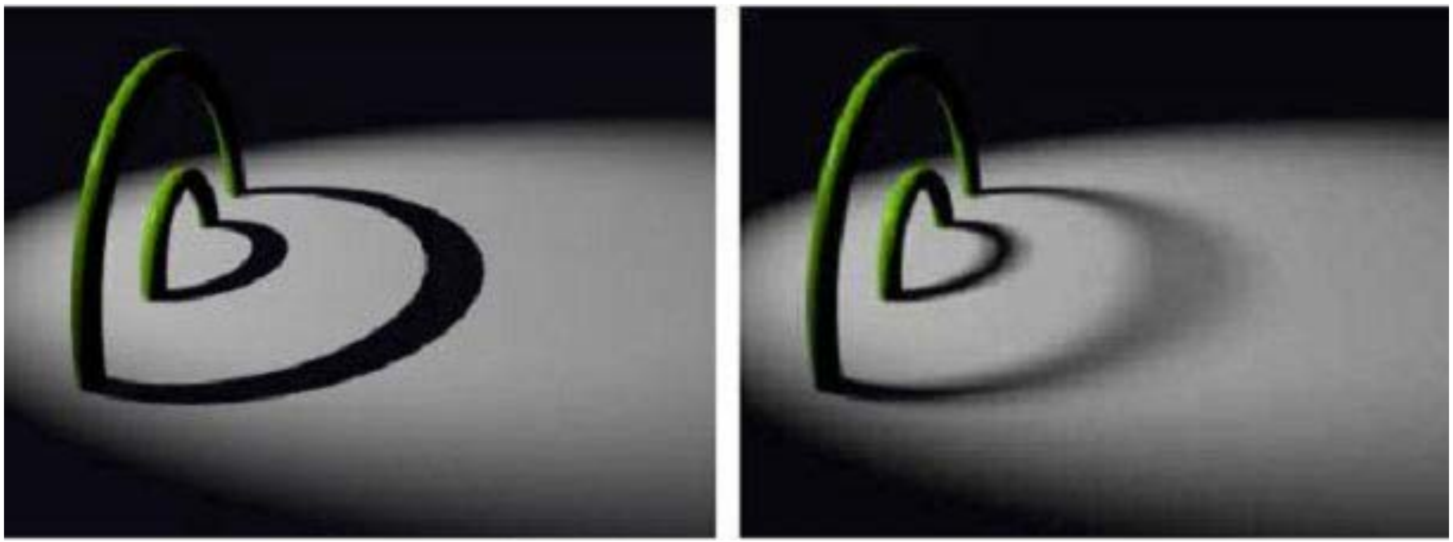
Anti-Aliasing

- Stratified or jittered Samples
  - Combines the best parts of both techniques
  - Divides the pixel in to a grid and then places each sample randomly inside each section
  - Allows semi-random positioning while still covering the majority of the pixel



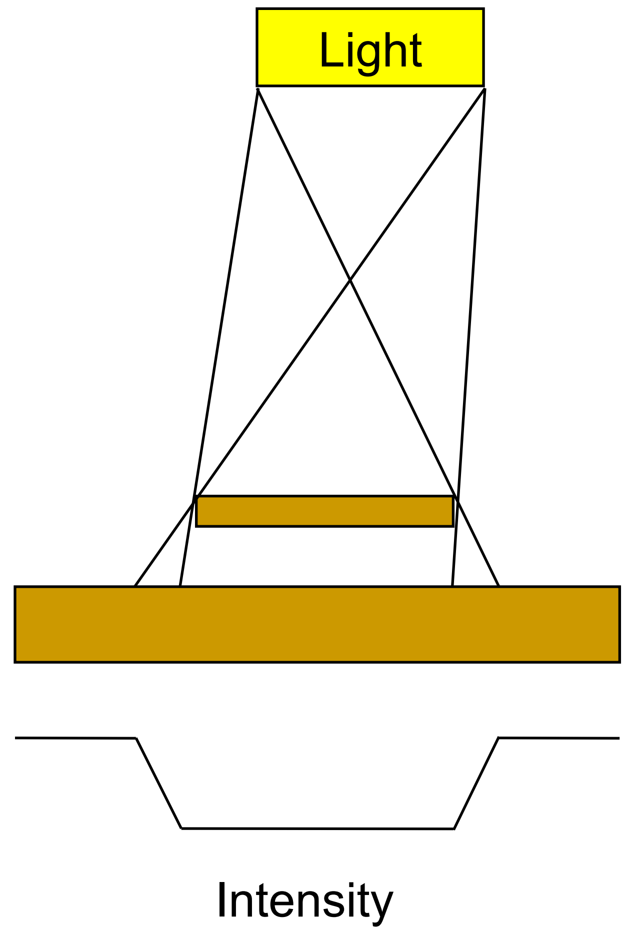
Stratified (jittered) samples

Soft Shadows



Soft Shadows

- Soft shadows occur because real light sources are not singular points (as assumed previously).
- Actual light sources have an area and therefore can give varying levels of shadow



Soft Shadows

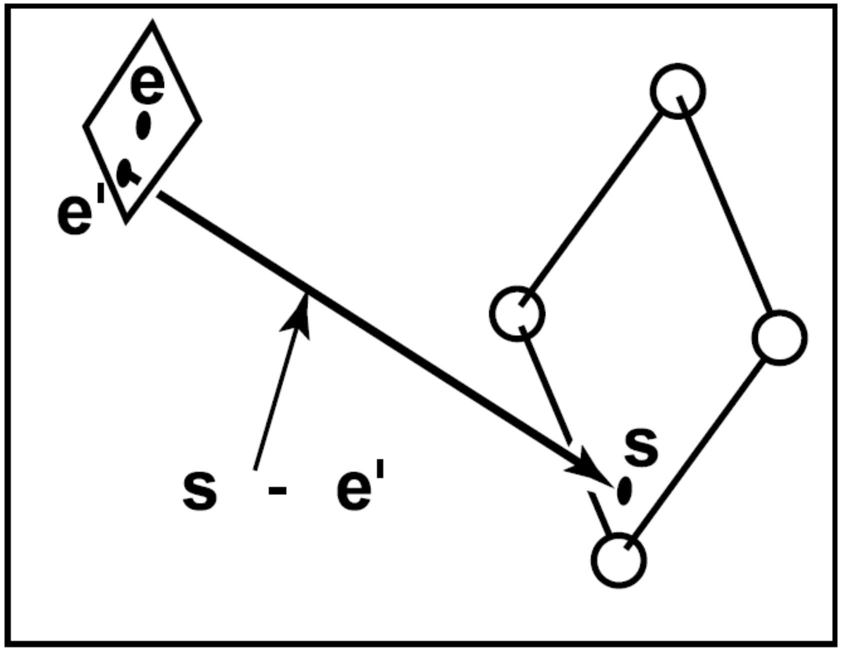
- Need to account for light source being an area rather than a point
- Could represent the light source as a series of points, however this would require many sources to get desired results
- Choose a random spot on the light source for each ray’s shadow determination
- Because of multiple samples for each pixel an average value results

Depth of Field

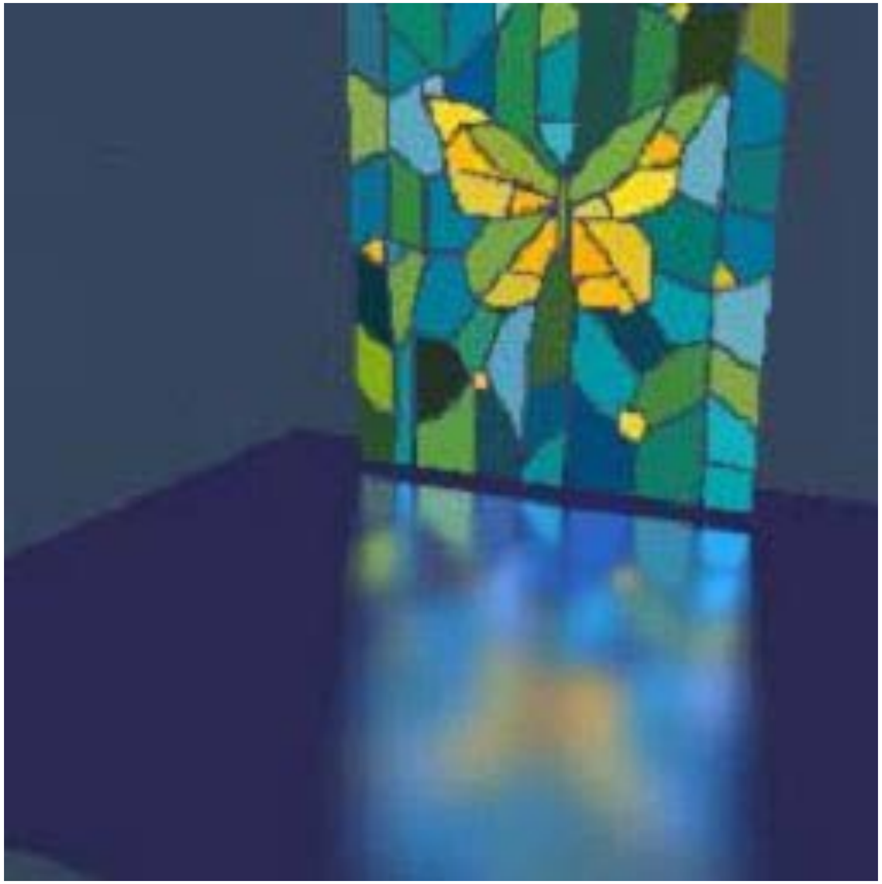
Depth of field

- The blurring effect when objects are out of focus, often seen in photographs
- Does not occur in simple ray tracing because the eye is represented by a single point, simulating an ideal pinhole camera.
- Solution: Represent eye by a non zero sized “lens” selecting a point on the surface at random for each ray

Depth of field



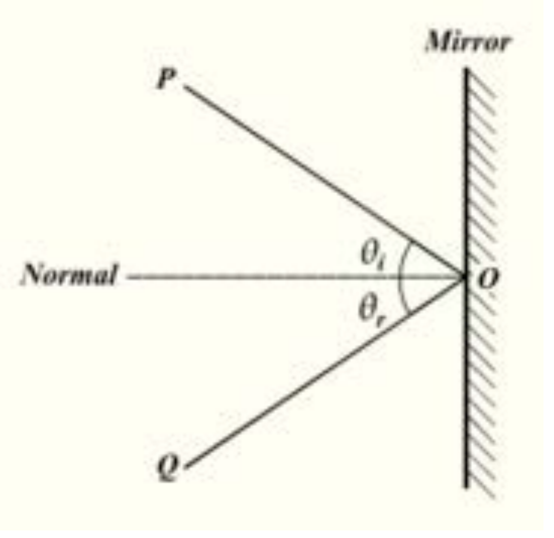
Glossy Reflections



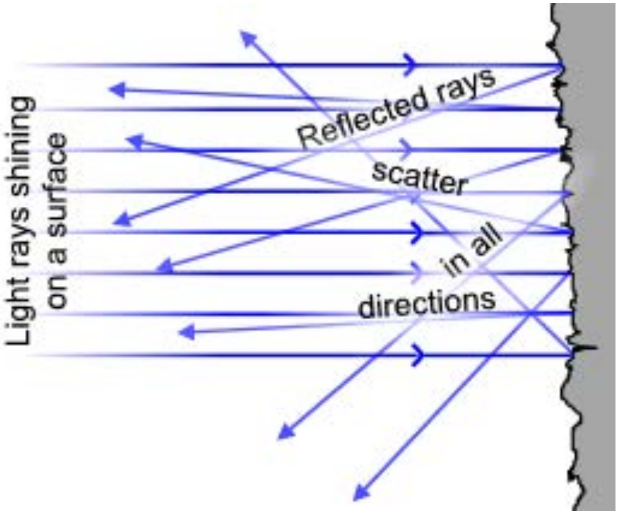
Glossy Reflections

- Previous rules for reflections only apply to an ideal mirror like surface
- Many surfaces such as brushed metal are somewhere between an ideal mirror and a diffuse surface producing a blurred reflection
- Reflection angle is somewhere between the ideal calculated response of a mirror surface and the generally random value of the diffuse surface

Glossy Reflections



Mirror Surface

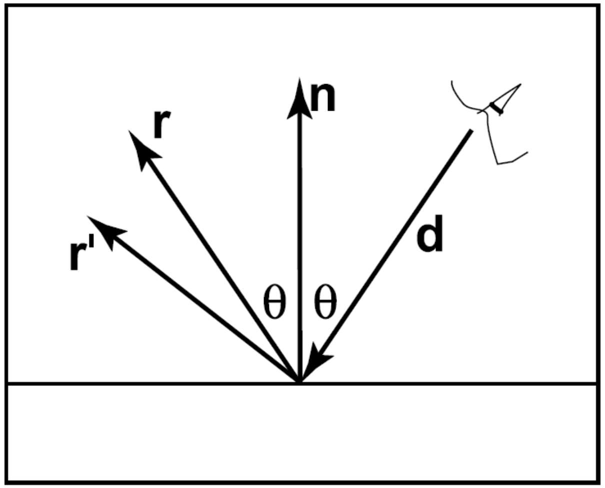


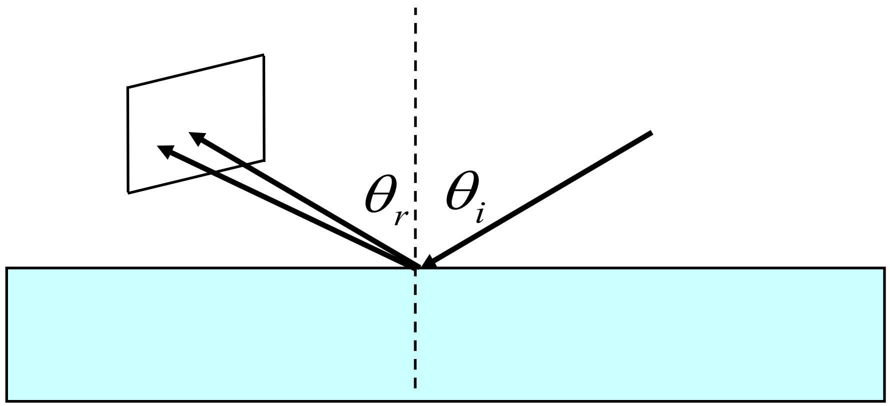
Diffuse Surface

Glossy Reflections

- Need to add a random element to the ideal reflection.
- Create a square centred on the ideal reflected ray and select a random spot within the square
- Size of square determines the blurriness of reflected image

Glossy Reflections





Motion Blur

Motion Blur

- When a camera takes an image, the image is formed over a non-zero length of time
- This creates blur as objects move over time
- Can simulate this in ray-tracing by setting a time variable ranging from T_0 to T_1
- Each viewing ray is sent at a random time within this range
- When combined, the rays sent at differing times and hence encountering objects at different positions to create blur

Distributed Ray Tracing - Limitations

- Distributed Ray Tracing creates many more rays than the basic method
- More processing required
- Need methods to decrease processing for rays
- Methods to reduce intersection calculations
  - Bounding Boxes
  - Hierarchical Bounding Boxes
  - Uniform Spatial Subdivision

Ray-Tracing in Practice

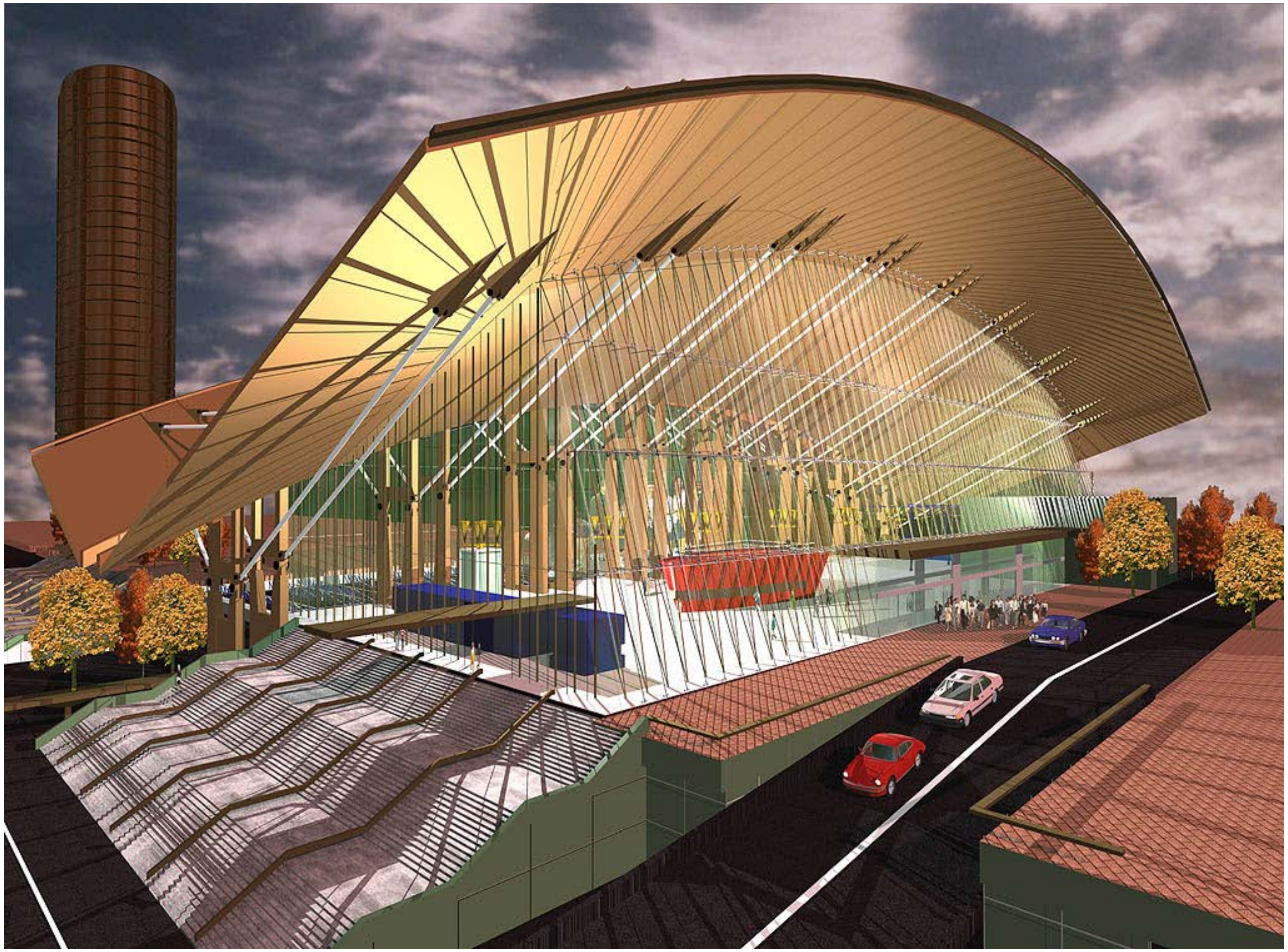
Ray-Tracing in Practice

Environment map

Ray-traced reflections

Ray-Tracing in Practice

Ray-Tracing in Practice



Ray-Tracing in Practice

Ray-Tracing in Practice

NVIDIA's GPU Ray Tracing Demo 2008
http://hothardware.com/News/NVIDIA-Shows-Interactive-Ray-Tracing-on-GPUs/

Some Links

Future Game Engines: Real Time Ray Tracing
https://www.youtube.com/watch?v=MidOCkSsQ-I

OpenCL Real-Time Raytracing
http://www.youtube.com/watch?v=TX56aqXoDW0&feature=related

Julia 4D ray tracing using CUDA
http://www.youtube.com/watch?v=QT4LLbyH3qY

Nvidia's demos
http://www.youtube.com/watch?v=w9SH8xIgzoI
https://www.youtube.com/watch?v=XISqvBVyASo
https://www.youtube.com/watch?v=XISqvBVyASo

Diamonds
http://www.youtube.com/watch?v=EissQ331WI8
http://www.youtube.com/watch?v=EXec-tHeRXI
http://www.youtube.com/watch?v=kIX5WL07Uss
http://www.youtube.com/watch?v=HLLYLCN-ma8

Water and Waves
https://www.shadertoy.com/view/Ms2SD1
http://madebyevan.com/webgl-water/

Conclusion

- Conceptually simple, yet powerful rendering method
- Produces many advanced effects by modelling the nature of light
- Use is limited by resource requirements
- Resource requirements become less of an issue over time with advancing technology

Software

- Brazil http://www.splutterfish.com
- Mental Ray http://www.mentalimages.com
- Final Render http://www.finalrender.com
- Cinema 4D http://www.maxon.net
- Skeleton ray tracers

  http://www.raytracegroundup/downloads.html
- Iray http://www.nvidia.com/object/iray-features.html

References

- “Fundamentals of Computer Graphics” by Peter Shirley
- “Computer Graphics C Version” by Donald Hearn, M. Pauline Baker
- “Computer Graphics with OpenGL” by Donald Hearn, M. Pauline Baker
- “Real-time Rendering” by Tomas Akenine-Möller, Eric Haines
- “Ray Tracing from the Ground Up” by Kevin Suffern, A K Peters, 2007.