

# Global Illumination and Ray Tracing

**CSE606: Computer Graphics**  
**Jaya Sreevalsan Nair, IIT Bangalore**  
**April 21-23, 2025**

# Agenda

- Global Illumination
  - Desirable Effects
  - Shadows
- Image-based rendering
  - Ray Casting
  - Ray Tracing
  - Path Tracing
- Aliasing and Anti-aliasing

# Global Illumination

# Global Illumination

- Object-centric rasterization process (with hardware assist) enables fast rendering of complex scenes with reasonable realism
- However, fundamentally limited in being able to reproduce many common optical effects, thus limiting realism
  - Computes contribution of a portion of a polygon (fragment) to the final image, based on **local information**: material, lights etc.
  - A pragmatic approximation to principles of optics
  - Cannot account for impact of other polygons on the rendering of a given fragment/polygon

# Global Illumination

Some of the effects that need to be rendered:

- Shadows (hard and soft)
- Reflections and refractions
- Color bleeding
- Caustics
- Light attenuation and ambient occlusion

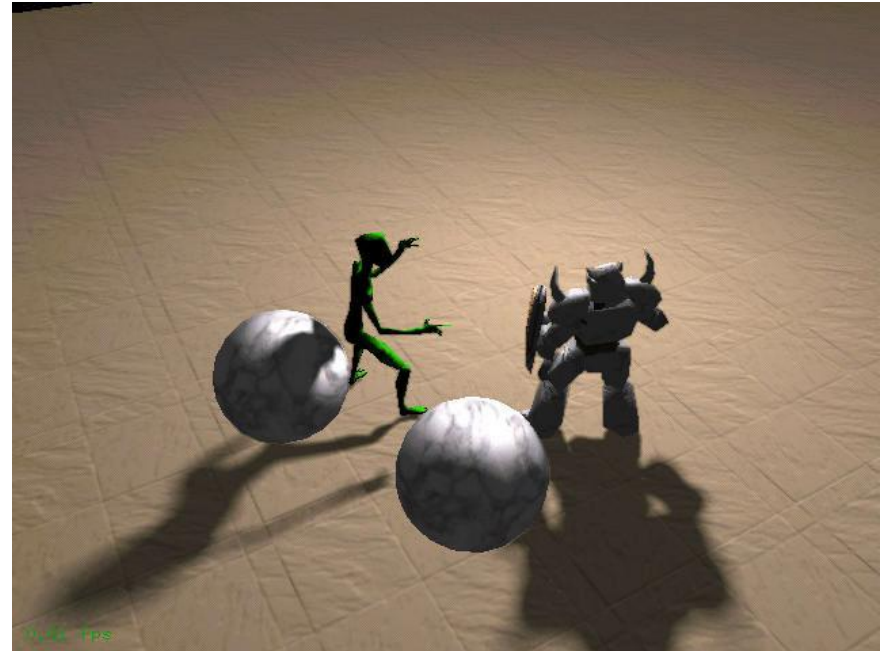
...

# Global Illumination

Some of the effects that need to be rendered:

- Shadows (hard and soft)
- Reflections and refractions
- Color bleeding
- Caustics
- Light attenuation and ambient occlusion

...



[graphics.cs.lth.se/research/shadows/](http://graphics.cs.lth.se/research/shadows/)

# Global Illumination

Some of the effects that need to be rendered:

- Shadows (hard and soft)
- Reflections and refractions
- Color bleeding
- Caustics
- Light attenuation and ambient occlusion

...



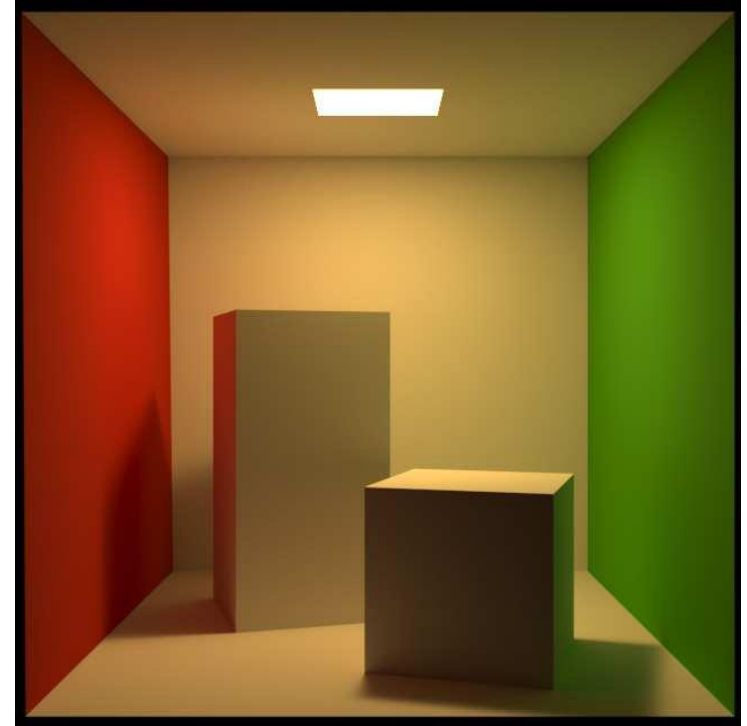
Giles Tran; Wikipedia

# Global Illumination

Some of the effects that need to be rendered:

- Shadows (hard and soft)
- Reflections and refractions
- Color bleeding
- Caustics
- Light attenuation and ambient occlusion

...



Jaroslav Krivanek

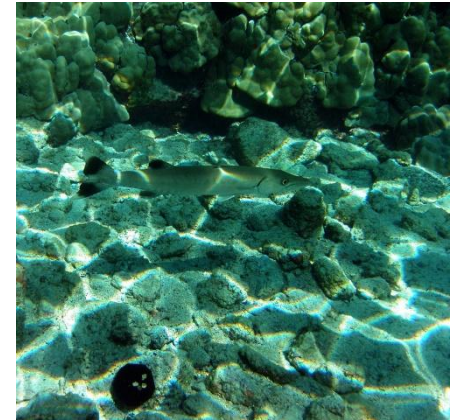
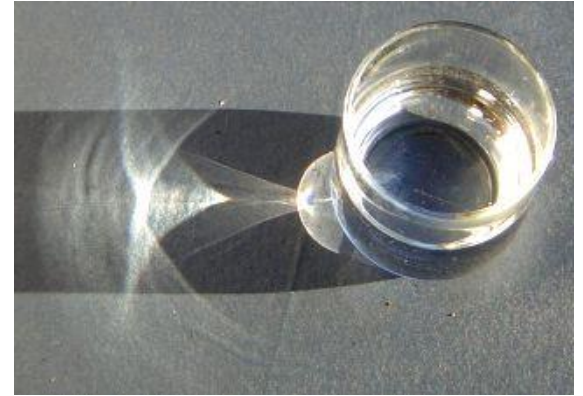


# Global Illumination

Some of the effects that need to be rendered:

- Shadows (hard and soft)
- Reflections and refractions
- Color bleeding
- Caustics
- Light attenuation and ambient occlusion

...



# Global Illumination

Some of the effects that need to be rendered:

- Shadows (hard and soft)
- Reflections and refractions
- Color bleeding
- Caustics
- Light attenuation and ambient occlusion

...

Light energy falls with distance from source – inverse square law

Ambient light is not necessarily uniform in all directions and points – since it is actually a combination of light from “infinite” sources including reflections

# Shadow Maps

- Multi-pass techniques can be used to generate shadows in the rasterization process – within the normal OpenGL pipeline, for instance
- First pass: compute distances of each point of scene objects from the light source. Can be computed by positioning camera at light source and storing depth in additional z-buffer
- Second pass: perform normal rendering. For each fragment, check distance from light source (using z-buffer stored earlier), and use fragment color only if not in shadow
- Can be extended for multiple light sources – one pass and buffer for each light source
- Soft shadows harder to achieve. Approximate “area light source” as multiple point light sources
- Aliasing a major issue. Needs post-filtering or other anti-aliasing processing

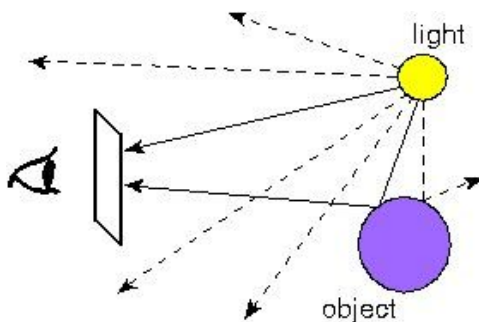
# Global Illumination - Implementation Approach

Follow all rays of light from all sources of light to the scene, and trace their paths as they reflect/refract off objects in the scene, accumulating color/intensity along the path

Those rays that enter the eye (through the pixels of the screen) form the final image

Computationally very expensive.

Also, how do we process “all rays”?



Albrecht Dürer, 1532

recreated by Drummyfish - wikipedia

[https://en.wikipedia.org/wiki/Ray\\_tracing\\_\(graphics\)](https://en.wikipedia.org/wiki/Ray_tracing_(graphics))

# Image-based Rendering

# Ray Casting

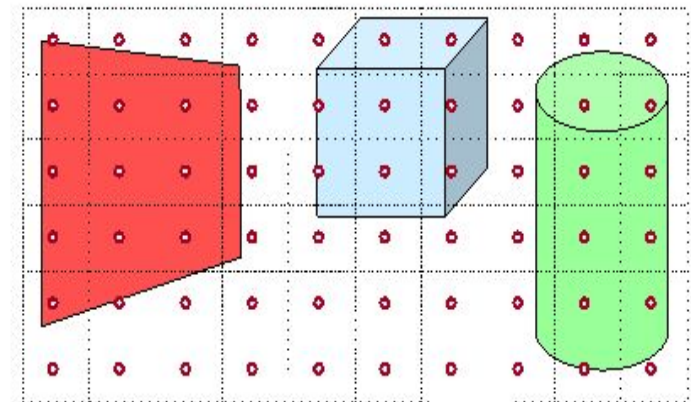
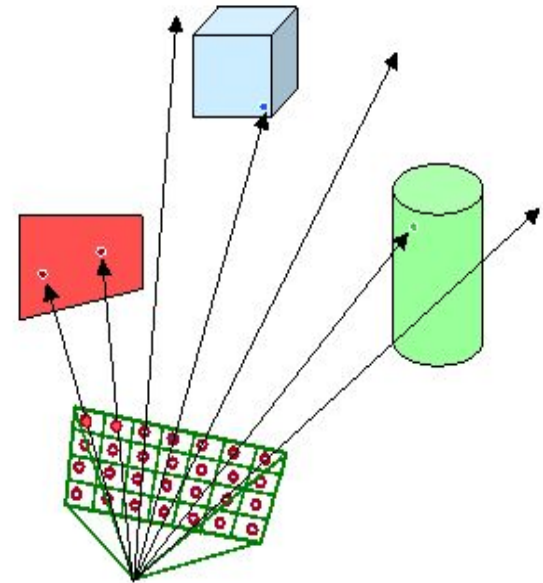
Need to compute rays entering the eye/camera for each pixel

Reverse the process:

- Shoot rays from camera to pixels on the screen.
- Extend till the first point of intersection with an element of the scene. Compute color at that point using any of the shading techniques.

Can be done for orthographic or perspective views.

Produces results similar to direct lighting techniques. Shadows etc. are not captured

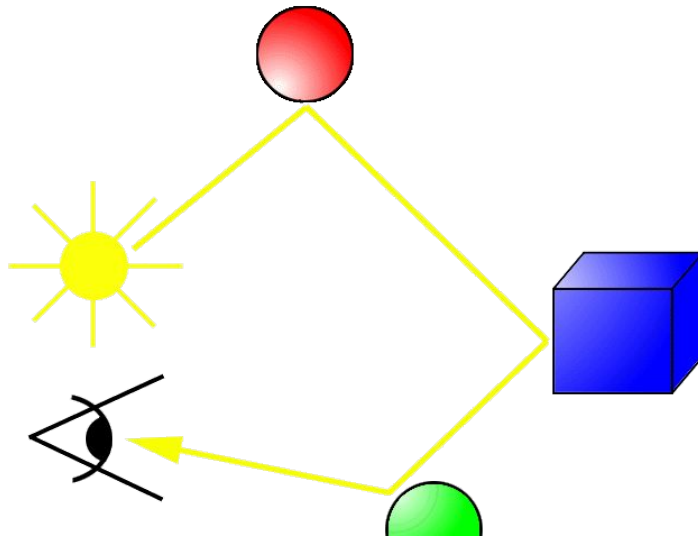


cs.princeton.edu

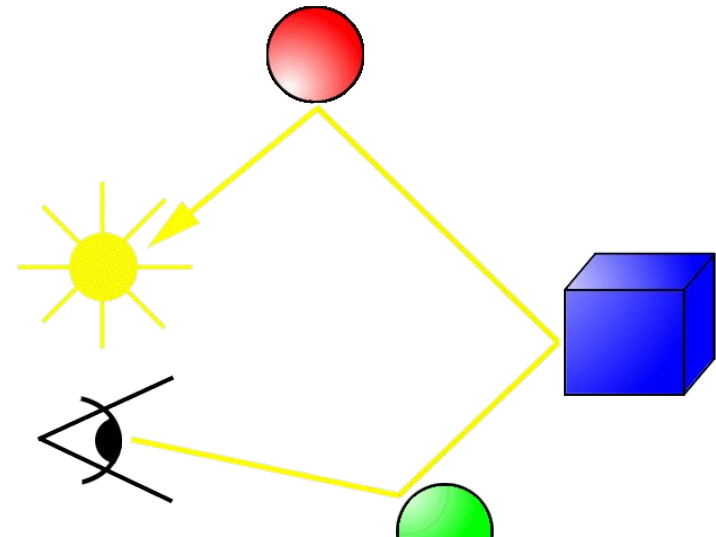
# Ray Tracing

As in ray casting, reverse the tracing process

Only need those rays that enter the eye – one for each pixel (or part of a pixel).



Forward or light-based ray  
tracing



Backwards or eye-based ray  
tracing

# Ray Tracing – Reflections

- At intersection point, generate a *reflection ray*: the ray that would be produced at the intersection point under normal laws of reflection
- Compute intersection (if any) of this ray with scene objects. If intersection exists, use the colour of that point on the object.
- Hence, can produce reflection effects



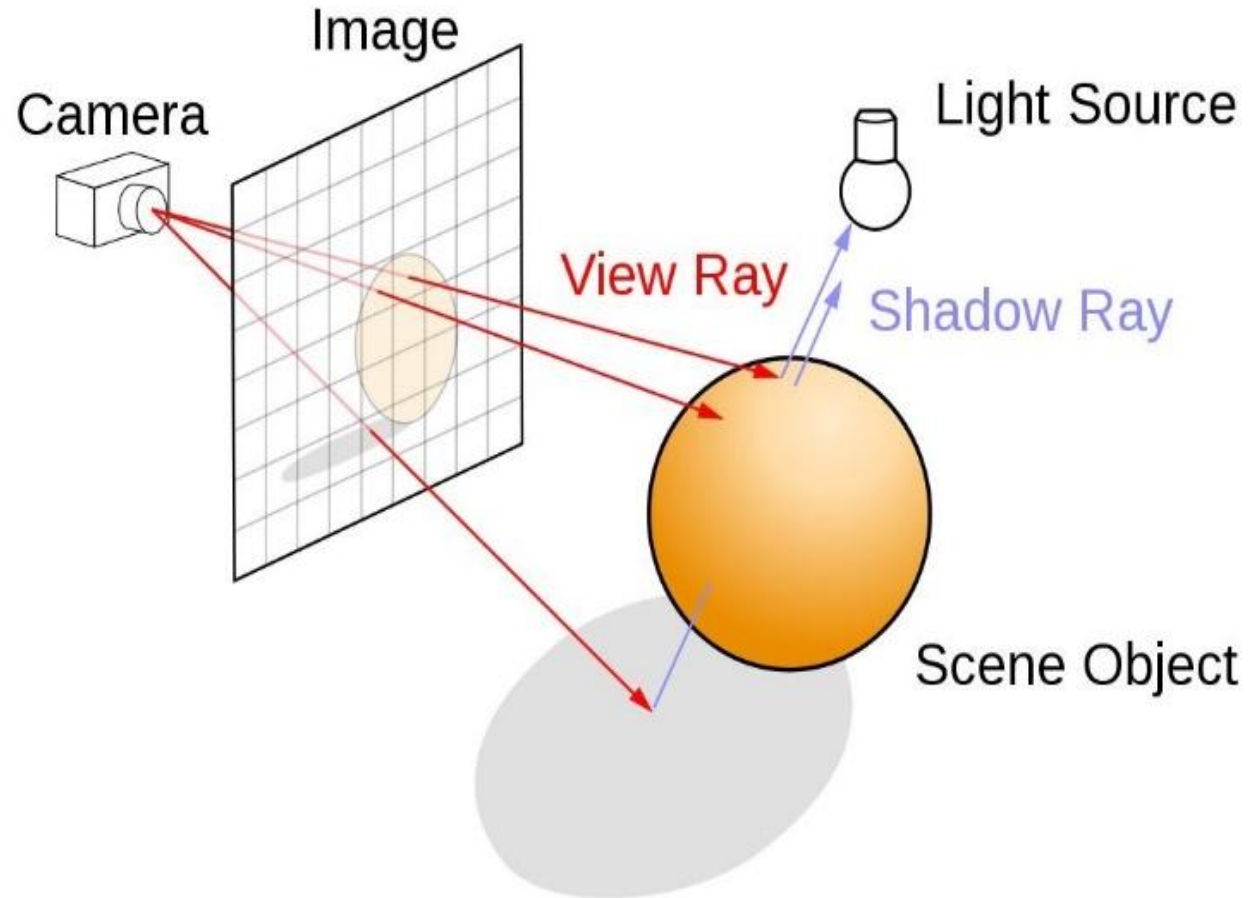
# Ray Tracing - Reflections



# Ray Tracing - Shadows

Shoot ray from eye through each pixel into the scene. At each intersection point

- Check if this ray is reachable from light source(s) or is occluded by some other object.
- Essentially, shoot a new ray from intersection point of first ray towards light source. Called a *shadow ray*
- If occluded, point is in the shadow and has no contribution from that light source. Thus, shadows are correctly rendered.

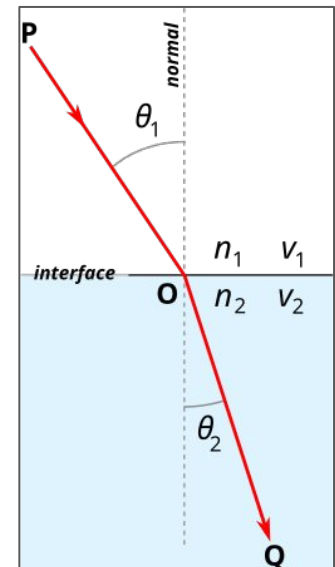


# Ray Tracing - Refractions

- For translucent objects, at the first intersection point, also generate a *refraction ray* – the ray that would be produced by refraction – Snell's Law
- Process as before – compute intersection of refraction ray with scene objects, and add that color (with appropriate transmission coefficient)

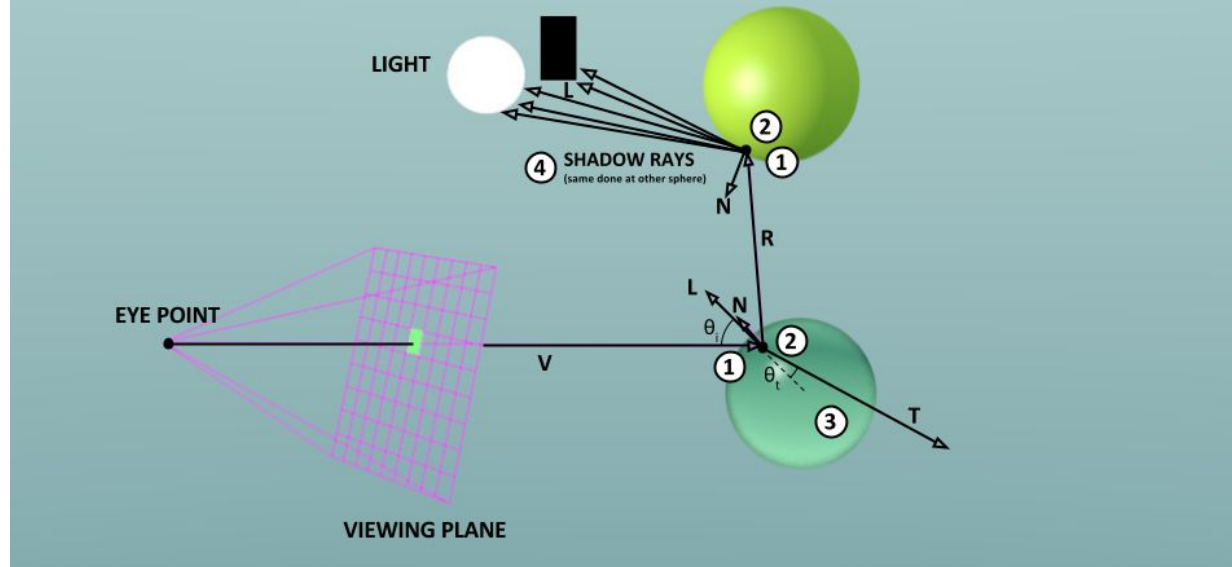
Snell's law states that, for a given pair of media, the ratio of the sines of angle of incidence  $\theta_1$  and angle of refraction  $\theta_2$  is equal to the relative refractive index of the second medium with regard to the first  $n_{21}$  which is equal to the ratio of the refractive indices ( $n_2/n_1$ ) of the two media, or equivalent to ratios of phase velocities ( $v_1/v_2$ )

$$\frac{\sin \theta_1}{\sin \theta_2} = n_{21} = \frac{n_2}{n_1} = \frac{v_1}{v_2}$$

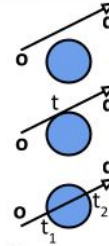


# RAY TRACING

(for one pixel up to first bounce)



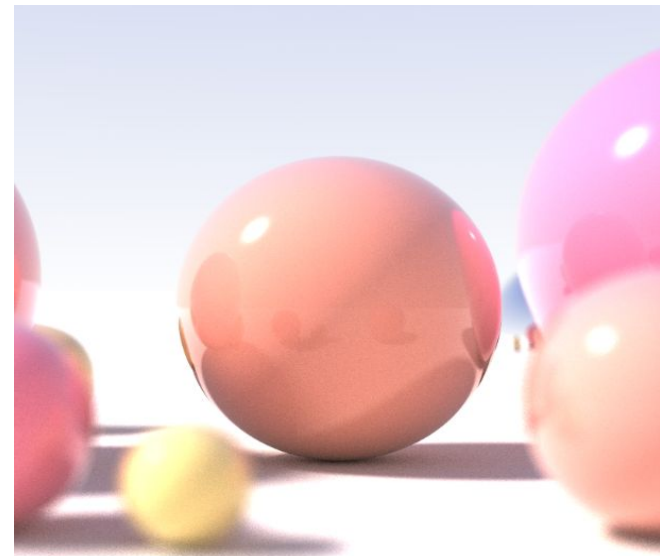
- ① Sphere equation:  $(\vec{p} - \vec{c}) \cdot (\vec{p} - \vec{c}) = r^2$  Intersection:  $(\vec{o} + t\vec{d} - \vec{c}) \cdot (\vec{o} + t\vec{d} - \vec{c}) = r^2$   
Ray equation:  $\vec{r}(t) = \vec{o} + t\vec{d}$   $t^2(\vec{d} \cdot \vec{d}) + 2(\vec{o} - \vec{c}) \cdot \vec{d} + (\vec{o} - \vec{c}) \cdot (\vec{o} - \vec{c}) - r^2 = 0$


- ② Illumination Equation (Blinn-Phong) with recursive Transmitted and Reflected Intensity:

$$I = k_a I_a + I_i \left( k_d \left( \vec{L} \cdot \vec{N} \right) + k_s \left( \vec{V} \cdot \vec{R} \right)^n \right) + \underbrace{k_t I_t + k_r I_r}_{\text{recursion}}$$
- ③ Snell's law:  $\frac{\sin \theta_1}{\sin \theta_2} = \frac{v_1}{v_2} = \frac{n_2}{n_1}$   $n_{air} \sin \theta_i = n_{glass} \sin \theta_t$  refraction coefficients:  $n_{air} = 1, n_{glass} = 1.5$
- ④ Area Light Simulation:  $I_{light} \frac{\#(\text{visible shadow rays})}{\#(\text{all shadow rays})}$

# Ray Tracing: Recursive Process

- Strictly speaking, the reflection and refraction process need to be continued for ever, or until the ray exits the scene.
- Hence, can be set up as a recursive process: each step of the recursion returns the color contribution for the entire path from one intersection point to the end of the ray – a light source or outside of the scene
- Modelled as a *ray tree*
- Clearly, computationally very expensive
- Need to have a termination criteria:
  - Number of recursion steps
  - Attenuation threshold, since some attenuation at every step



Tim Babb - wikipedia

# Ray Tracing – Intersection Computations

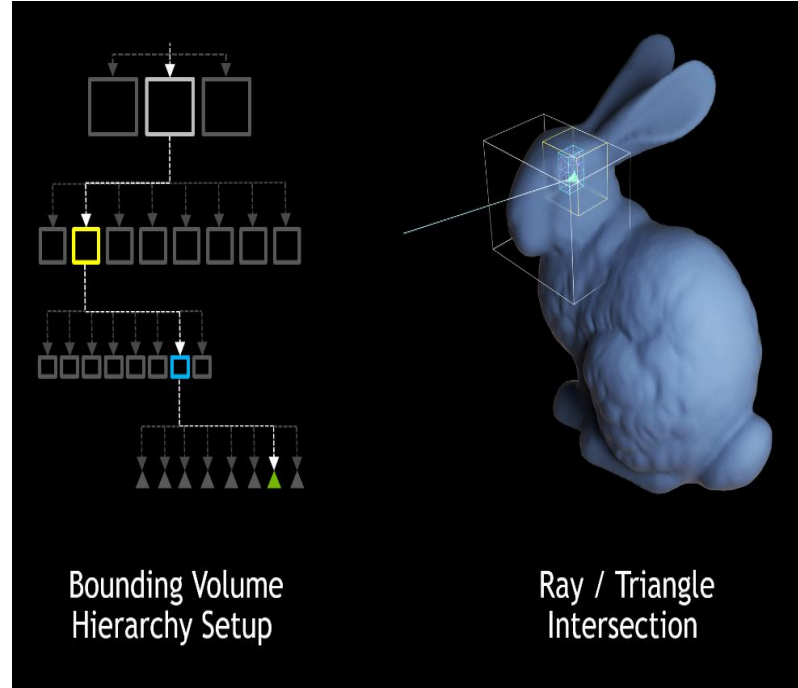
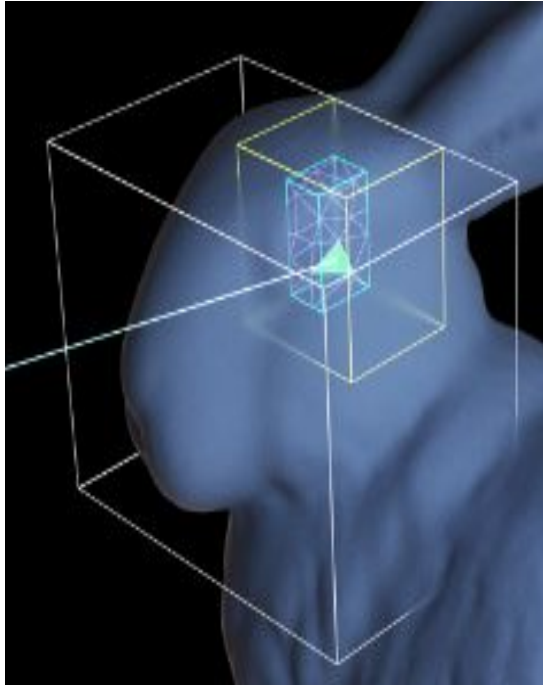
- Special cases of geometries such as spheres and cubes can be processed with special intersection modules
  - Line-sphere intersection
  - Line-plane intersection
- For more general objects and/or complex scenes, use culling techniques to minimize actual intersection computations
  - BSP trees, bounding boxes etc
  - Find subset of elements that could intersect a ray

# Ray Tracing - Aliasing

- Since we shoot one ray per pixel, aliasing effects are to be expected:
  - Missing small features
  - Sharp changes across adjacent pixels

# Ray Tracing in Hardware

- Can exploit massive parallelism, since each ray can be traced independently. However, each ray needs access to the entire scene
- Hardware support for:
  - Bounding Volume Hierarchy tests
  - Ray/Triangle intersection

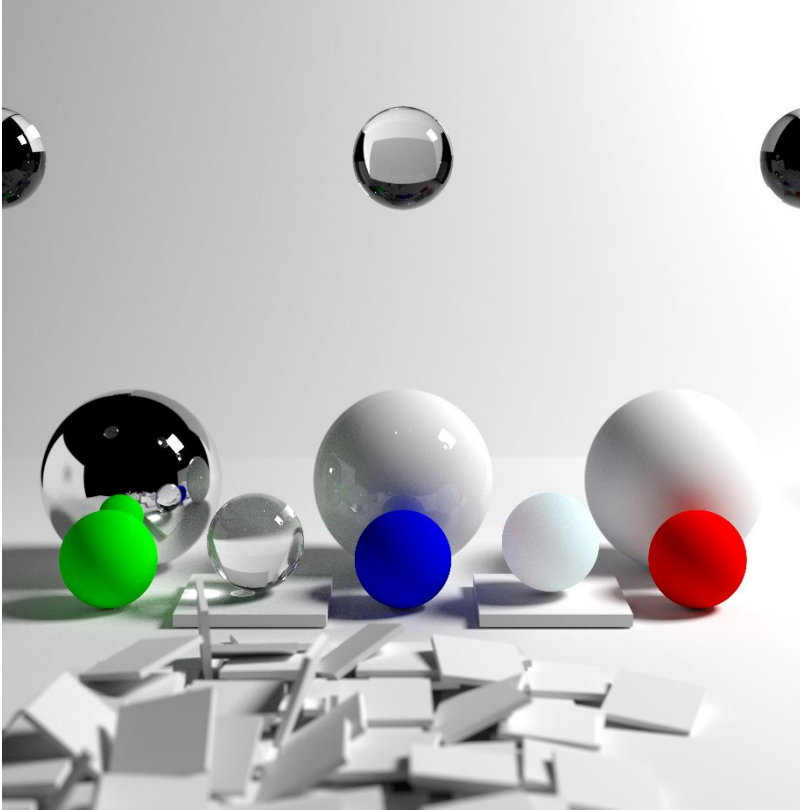




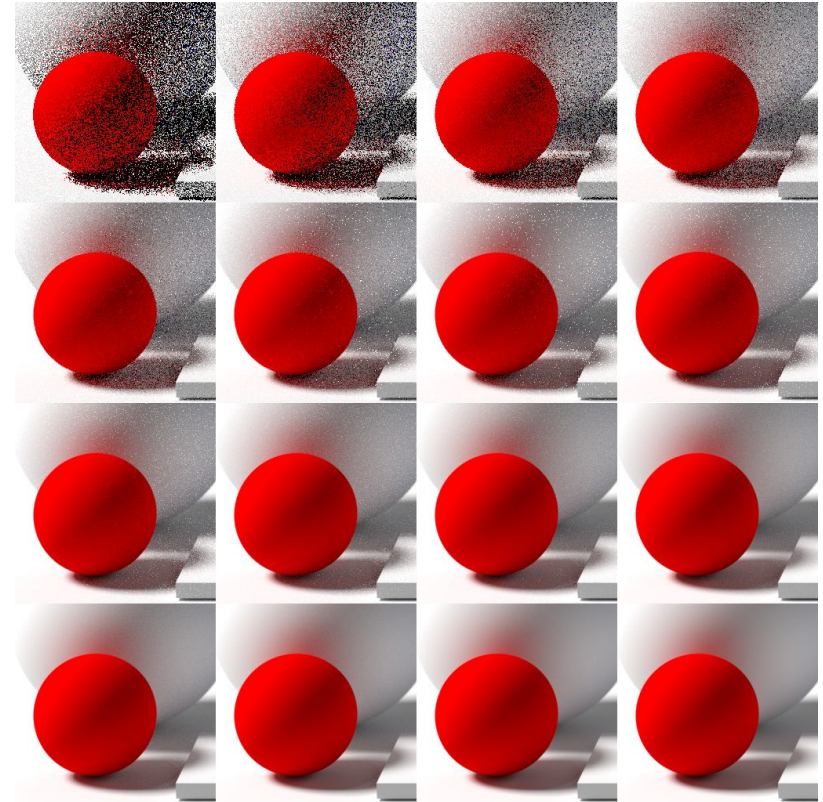
# Path Tracing

- To correctly model diffuse reflections/refractions, need to generate multiple reflection/refraction rays at each step. Further increases computation cost
- Instead, generate only one ray at each step – in a random direction. Hence one path per pixel sample
- Produces a very noisy image
- Improved by increasing the number of samples per pixel (at random sampling points within the pixel)
  - Image quality improves at every step

# Path Tracing



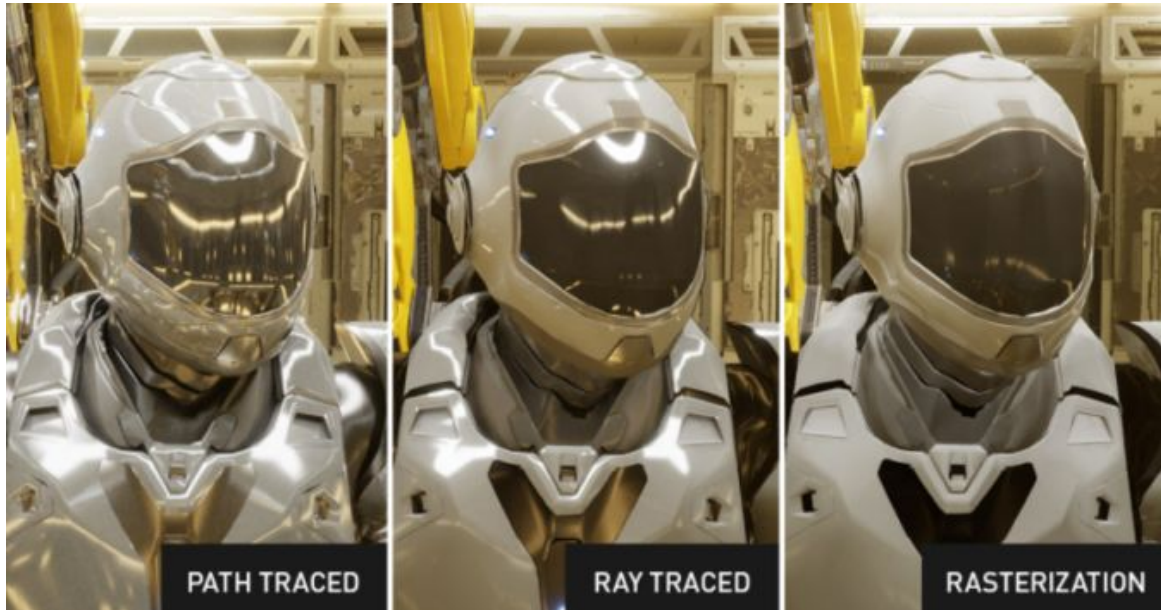
An image rendered using path tracing, demonstrating notable features of the technique



Noise decreases as the number of samples per pixel increase. The top left shows 1 sample per pixel, and doubles from left to right each square.

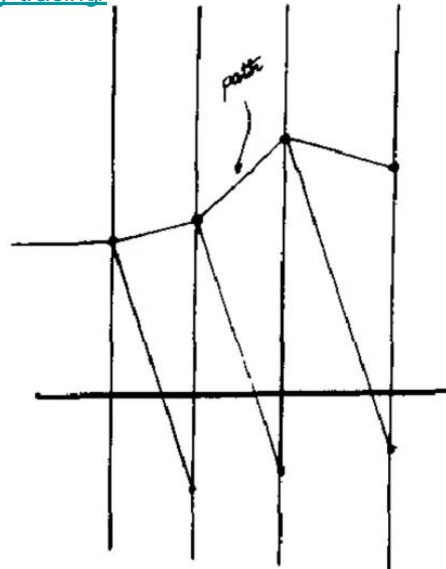
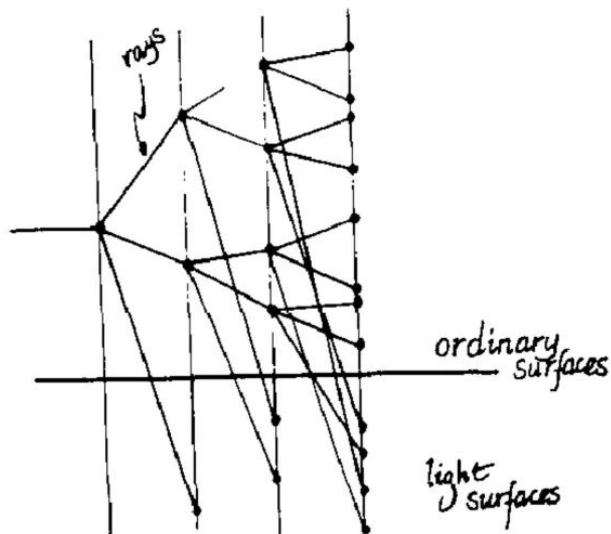
# Path Tracing vs Ray Tracing

Kajiya's hand-drawn diagram of ray- vs path-tracing ↓↓

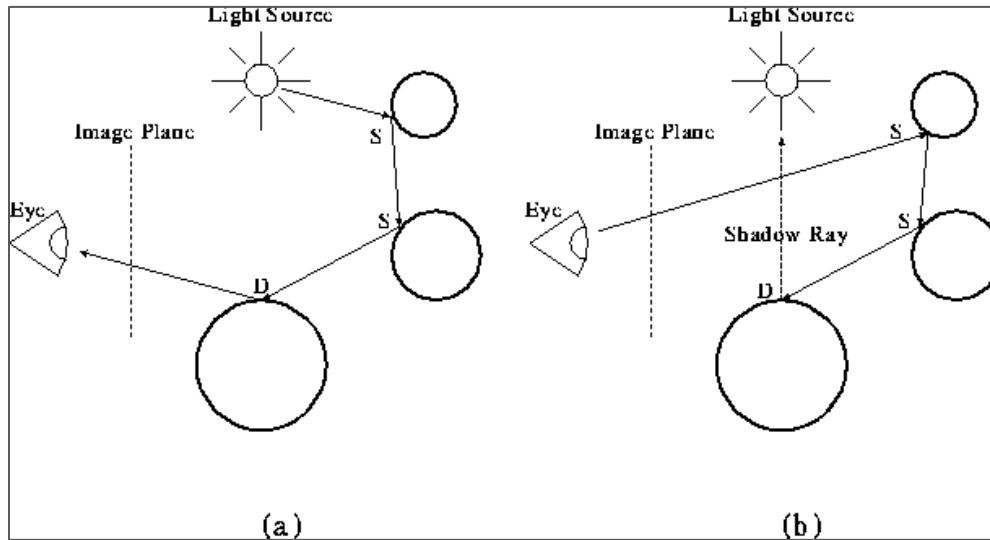


<https://www.techspot.com/article/2485-path-tracing-vs-ray-tracing/>

<https://blogs.nvidia.com/blog/what-is-path-tracing/>



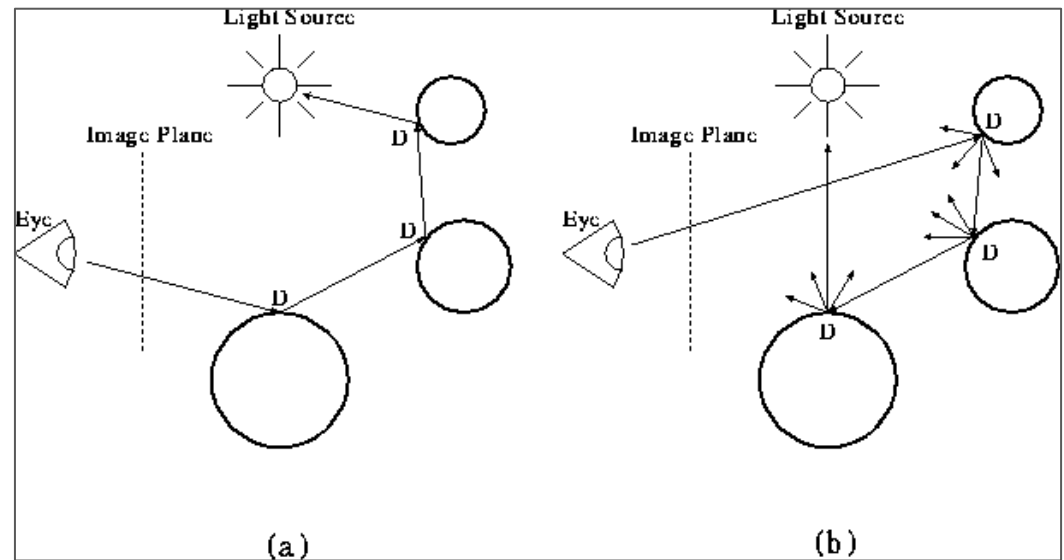
# Comparison of Methods



Classical ray tracing methods (a: photon tracing, b: visibility ray tracing)

**Bidirectional ray tracing** = photon tracing + visibility ray tracing

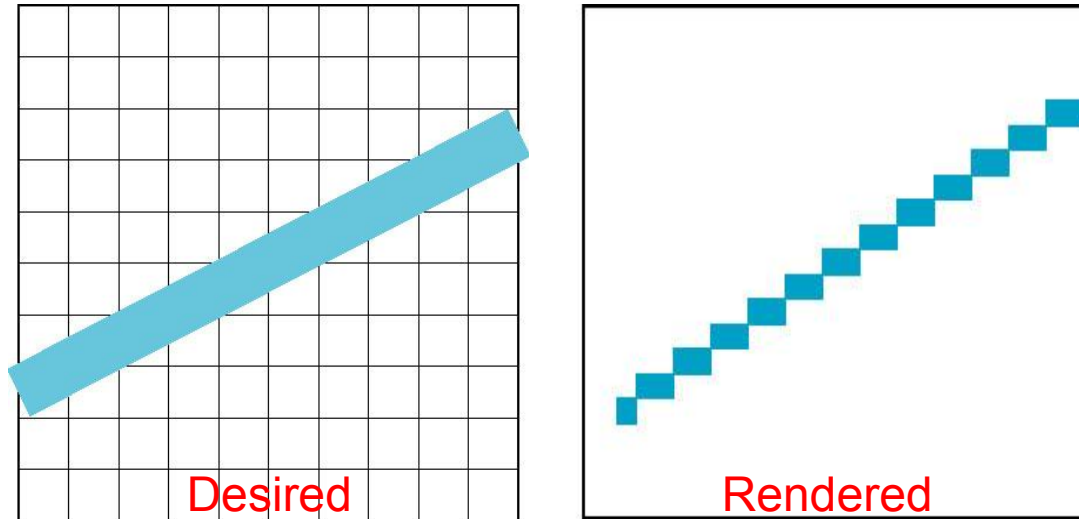
Monte-Carlo ray tracing methods (a: path tracing, b: distributed ray tracing)



# Aliasing and Anti-aliasing

# Aliasing

- Ideal rasterized line should be 1 pixel wide



- Choosing best y for each x (or visa versa) produces aliased raster lines (multiple lines map to the same set of pixels)

## Sources

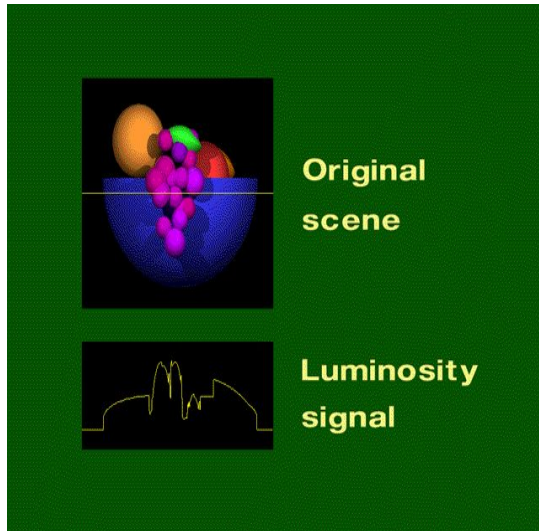
- SIGGRAPH Education Slide Set: *Overview of Aliasing in Computer Graphics* at <https://education.siggraph.org/static/HyperGraph/aliasing/alias2a.htm>
- Notes/slides by Tomas Akenine-Möller on the book “*Real Time Rendering*”

# Aliasing

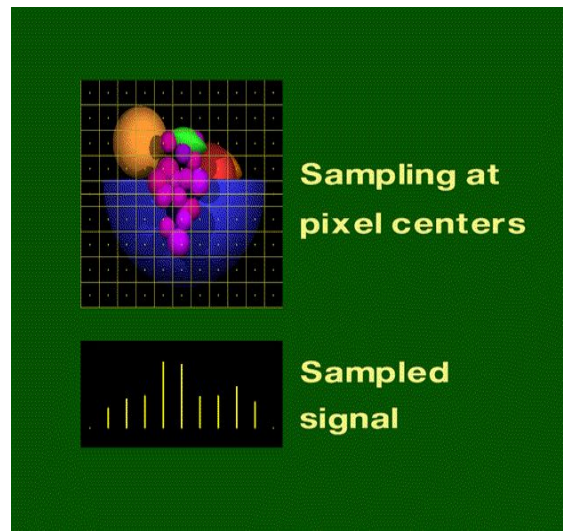
- Why is this an issue?
- Need high quality rendering, especially as the size of the screen increases or where higher quality is needed
  - Example, game vs. movie
- Problem is that we need to subsample a high-resolution object to be able to render into a lower resolution screen:
  - Lines, textures
  - Time (for animation)



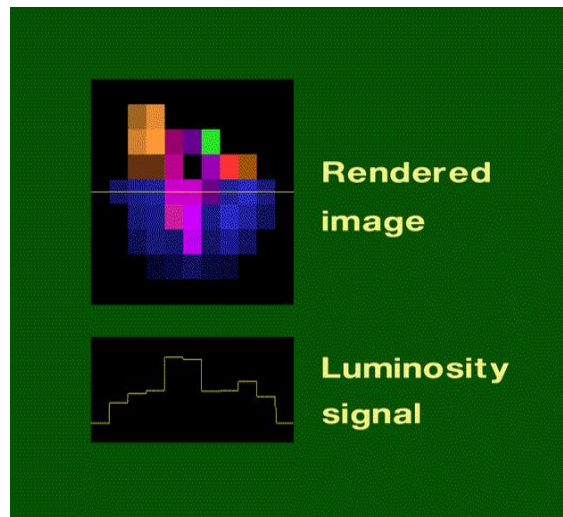
# Sampling an Image



Luminosity signal across one scan line in the original image



Significant loss of information if sampled once per pixel



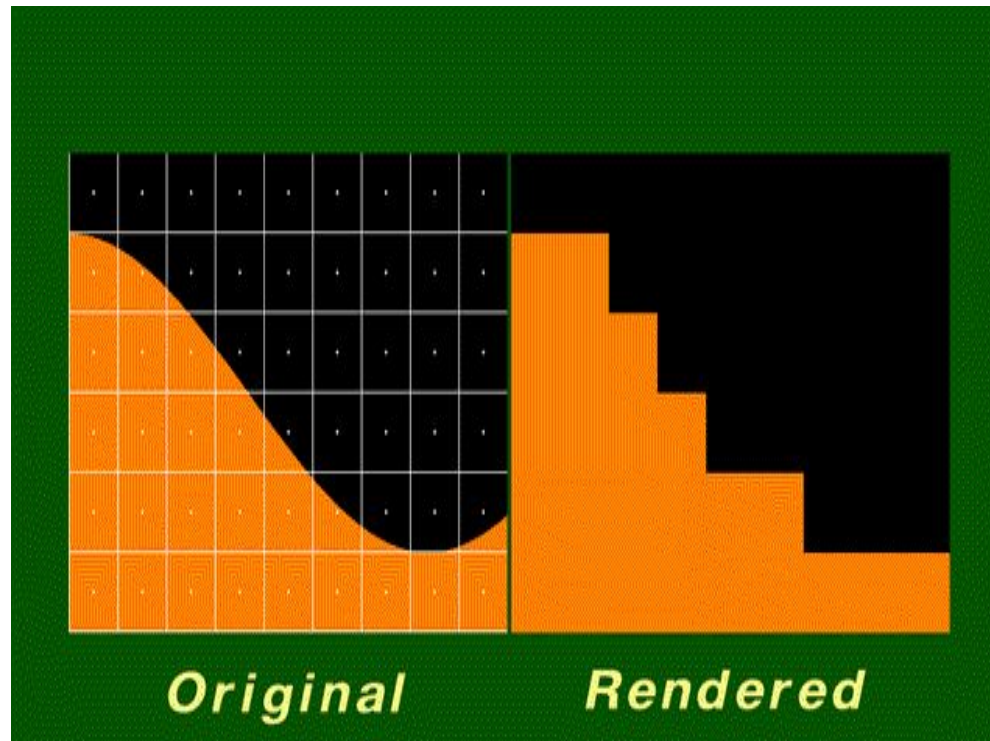
Source: ACM SIGGRAPH Education Slide Set

# Aliasing

- A general problem in signal processing:
  - Need to sample from the original signal that is continuous
  - Reconstruct to recover the original signal
  - Can produce loss of information and aliasing
- Nyquist theorem: sampling rate must be at least twice the max frequency in the signal
- But max frequency may not be known or is infinite

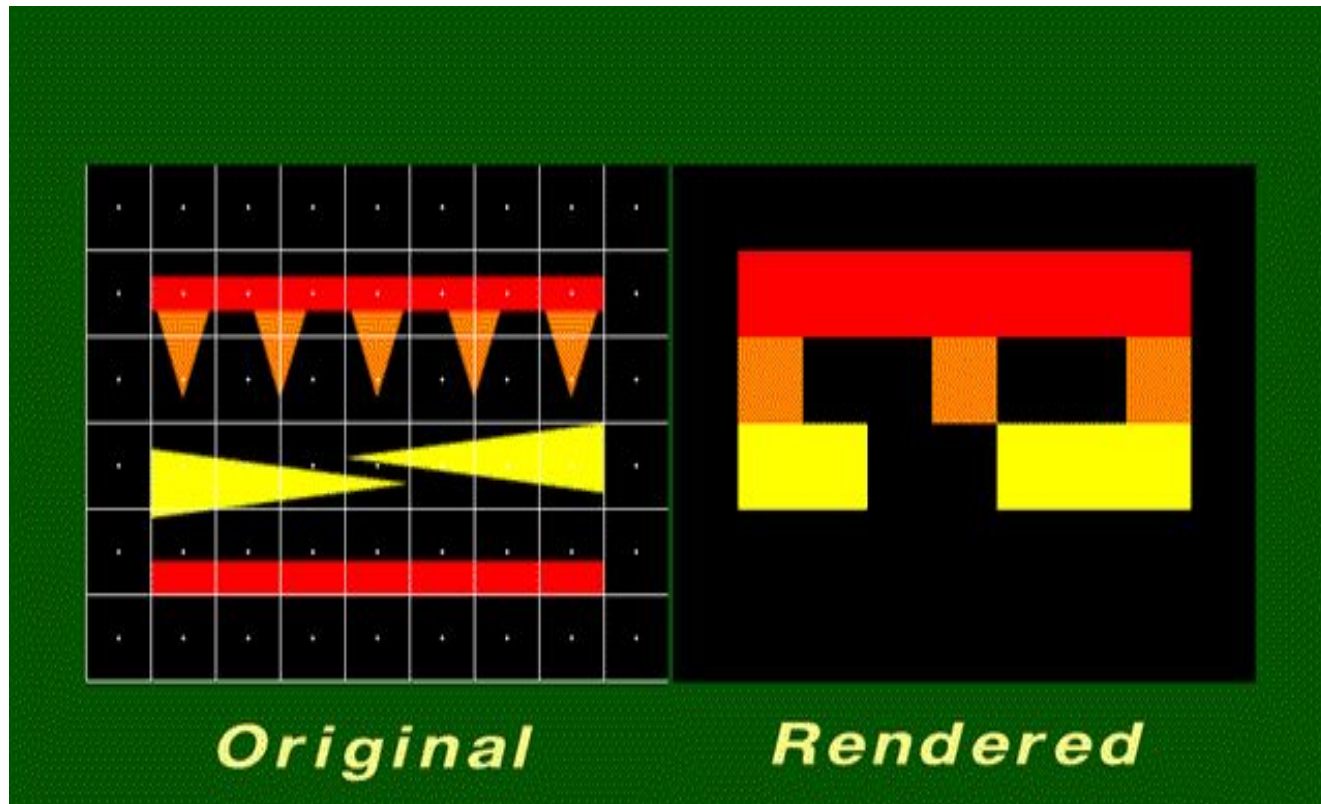
# Aliasing Artefacts

- If we assume sampling at the center of the pixel, jagged profiles or “jaggies” can be produced
  - Especially visible at silhouettes, and where there is high contrast between adjacent regions/polygons



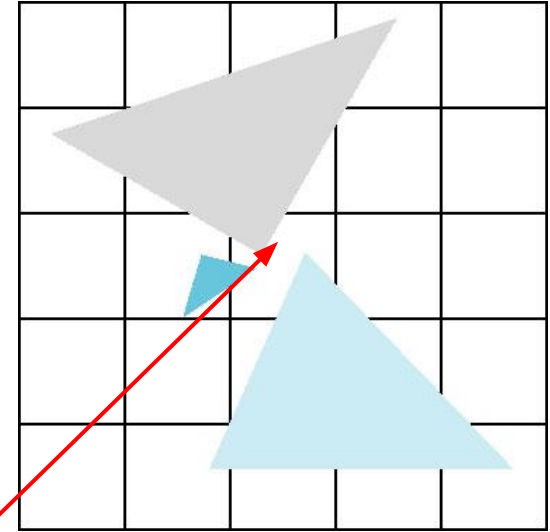
## Aliasing Artefacts – Loss of Detail

- Entire polygons can be lost, or rendered such that the shape and/or size are incorrect



# Rendering Polygons

- Aliasing problems can be serious for polygons
  - Jaggedness of edges
  - Small polygons neglected
  - Need compositing so color of one polygon does not totally determine color of pixel



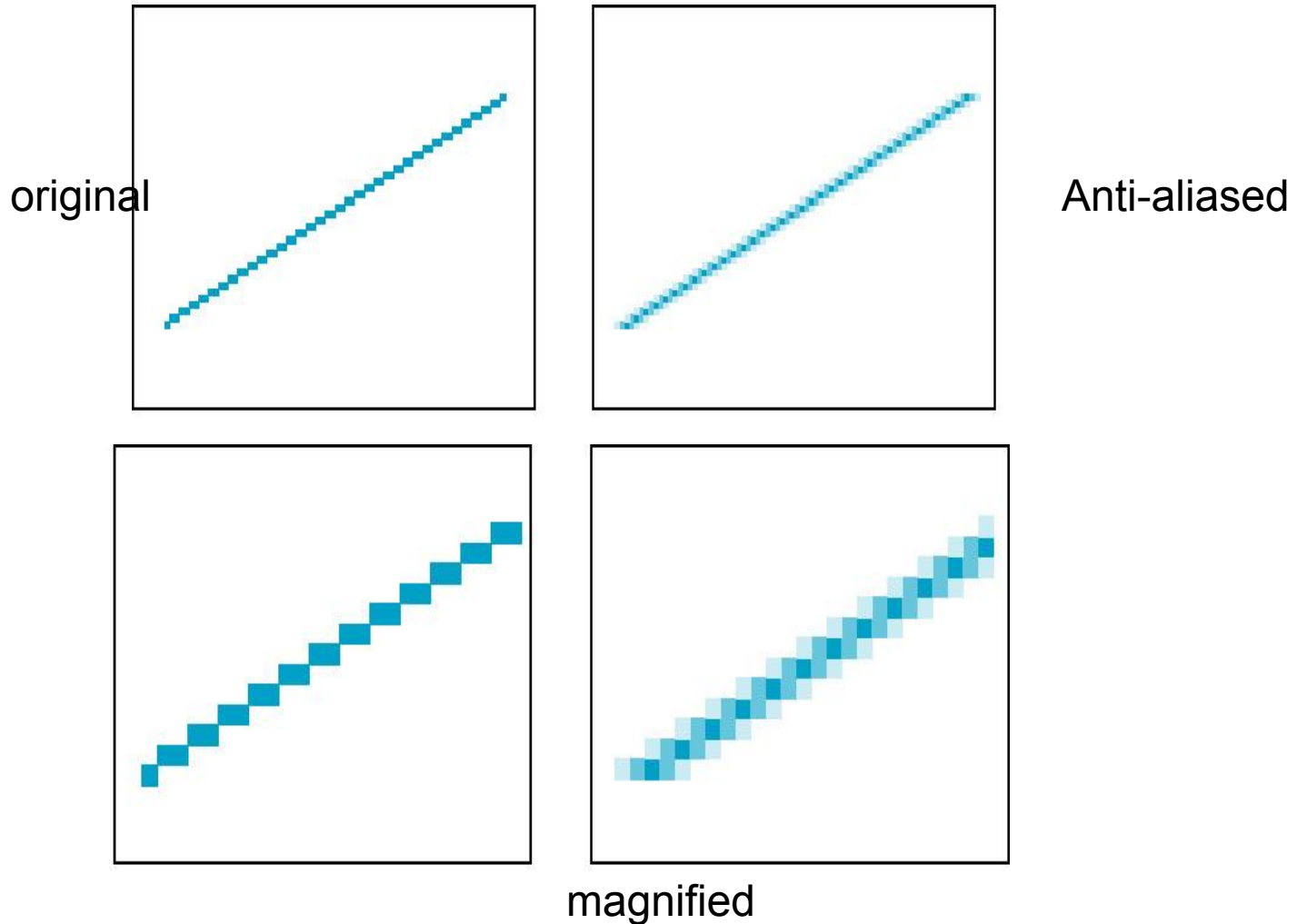
All three polygons should contribute to color

# Anti-aliasing

- Artefacts of aliasing can be corrected by “anti-aliasing” techniques
  - Achieved broadly by increased sampling and appropriate filtering
- Screen-based anti-aliasing
  - Difficult, since an edge has infinite frequency
  - Gets harder for polygons
- Approaches:
  - Pre-filtering: treat a pixel as an area, compute contribution of a fragment based on its overlap with the pixel. Add up all such contributions
  - Post-filtering or Super-sampling: use multiple samples to compute colour of a pixel. Use weighted average to combine samples.

# Anti-aliasing by Area Averaging

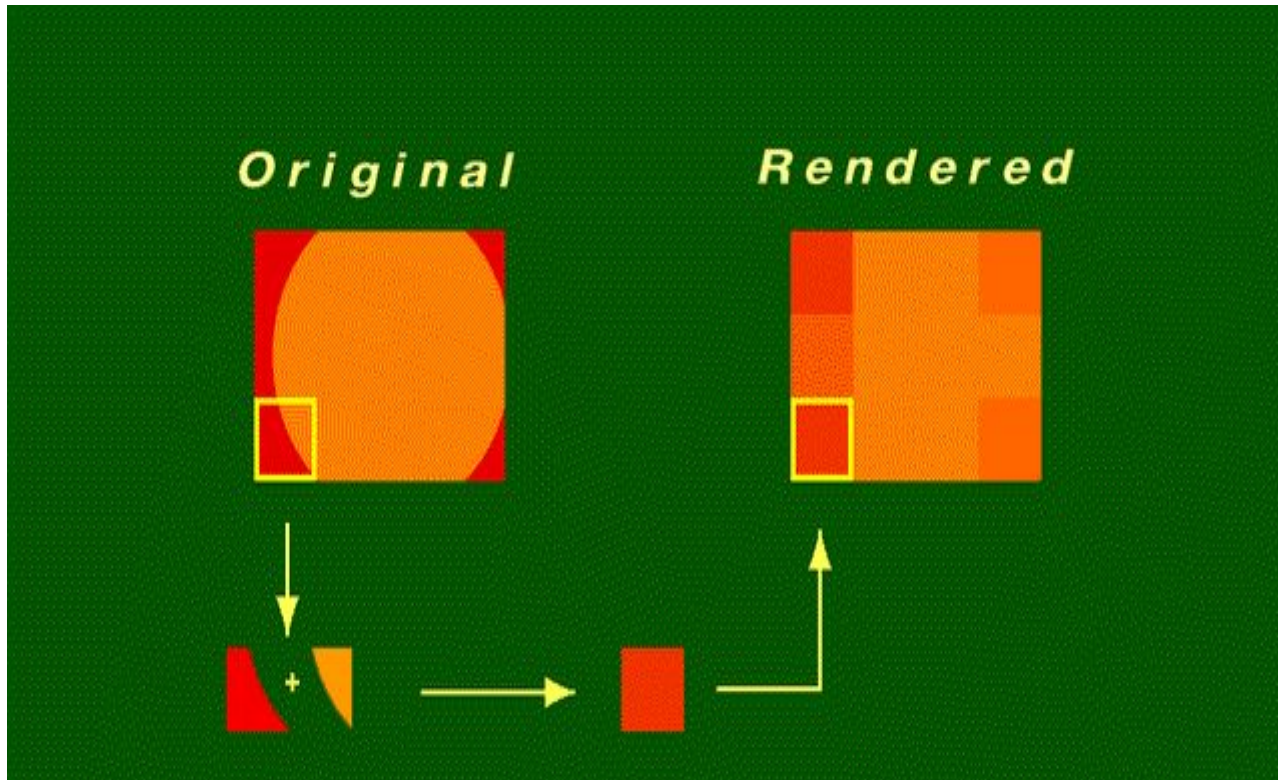
- Color multiple pixels for each x depending on coverage by ideal line





# Pre-filtering

- Pixel colour based on fraction of contribution of each fragment/polygon that overlaps the pixel (e.g area of polygon overlap \* colour of polygon)





## Pre-filtering



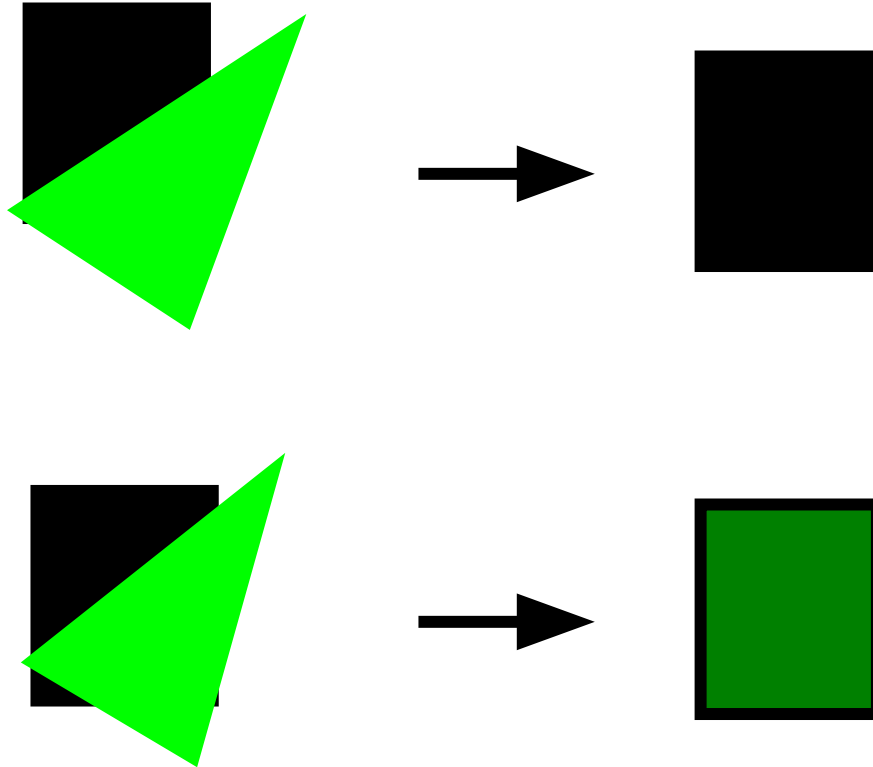
Original



Pre-filtered

# Polygon – Anti-aliasing

- One sample per pixel not sufficient
- Sample at multiple points, apply weighted average of sampled values



# Sampling Pixels

- One sample per pixel not sufficient
- Sample at multiple points, apply weighted average of sampled values
- The pixel value at  $x, y$  :

$$p(x, y) = \sum_{i=1}^n w_i \cdot c(i, x, y)$$

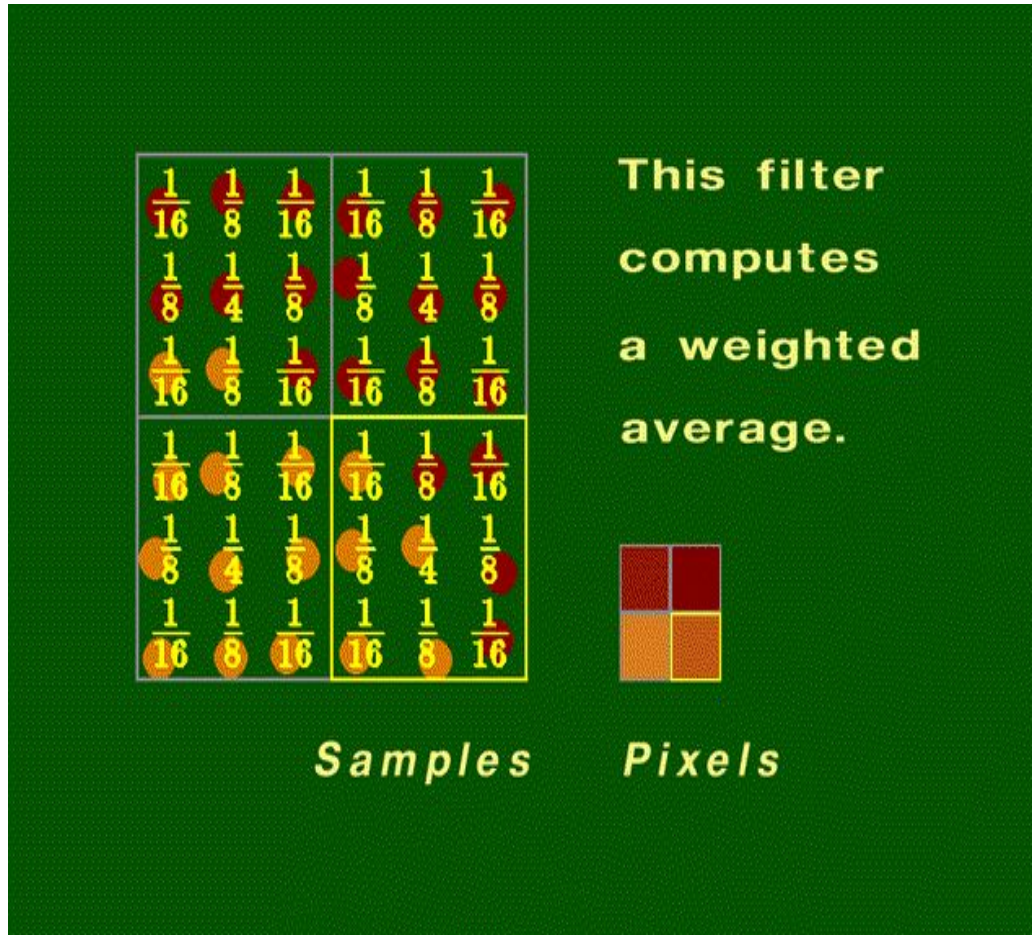
where  $w_i$  are weights and

$c(i, x, y)$  is the colour of sample  $i$  at location  $x, y$

- Many sampling schemes used

# Super-sampling

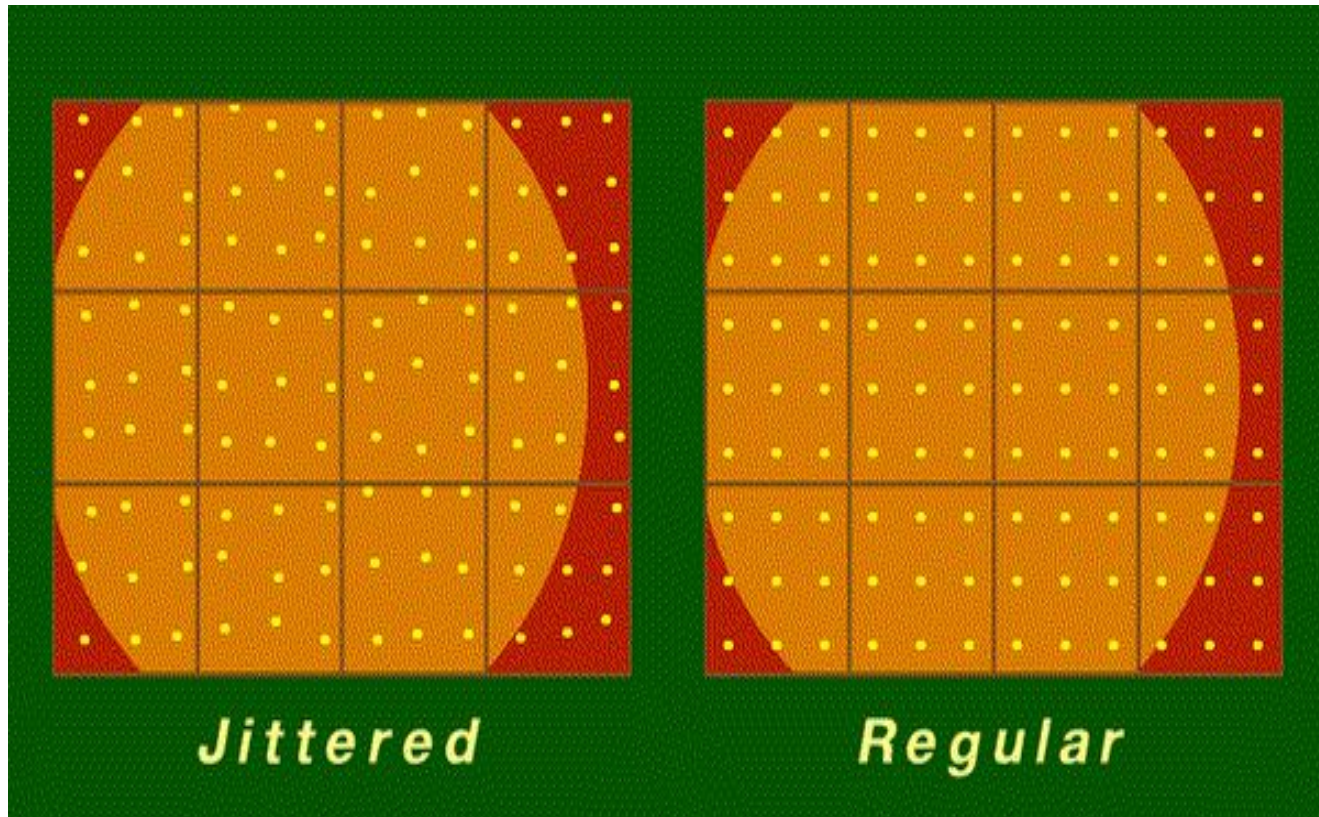
Sample 9 points per pixel – and a possible filter for averaging





# Jittered Sampling

- Use jittered sampling: random positioning of sample points within a pixel
- Essentially replace aliasing with noise
- Eye more tolerant to noise than to aliasing artefacts



# Ray Tracing - Aliasing

- Since we shoot one ray per pixel, aliasing effects are to be expected:
  - Missing small features
  - Sharp changes across adjacent pixels
- Anti-aliasing techniques
  - Super-sampling – multiple rays per pixel. Only reduces the scale of aliasing artefacts
  - Adaptive supersampling/Monte Carlo sampling: shoot more rays per pixel only if rays within a pixel produce sufficiently different colours
  - Stochastic Ray Tracing – pick reflection rays at random from a set of possible “incoming” rays

# Anti-aliasing in OpenGL

- Super-sampling in OpenGL (SSAA)
  - run the fragment shader for each sample point. Essentially create buffers that are  $n$  times larger, if  $n$  sample points per pixel
    - Significant increase in buffer size and computations
- Multi-sampling Anti-aliasing (MSAA)
  - More commonly used.
  - Rather than compute the colour at each sample point, the colour is computed for the fragment as usual, but multiple sample points are generated, and tested to see if they lie within the polygon. This is multiplied by the fraction of sample points that lie within the polygon.

# Summary

- Global Illumination
- Ray Tracing
- Aliasing/Antialiasing