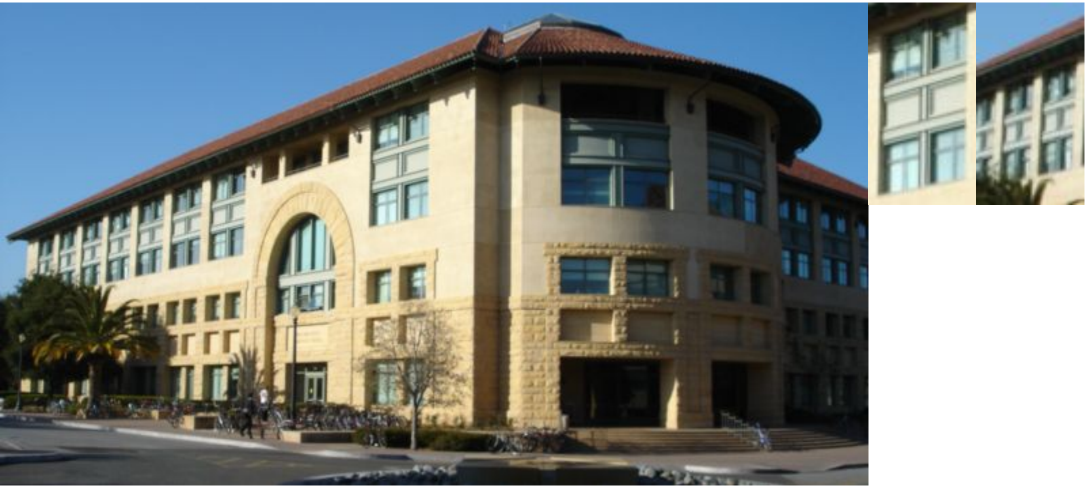
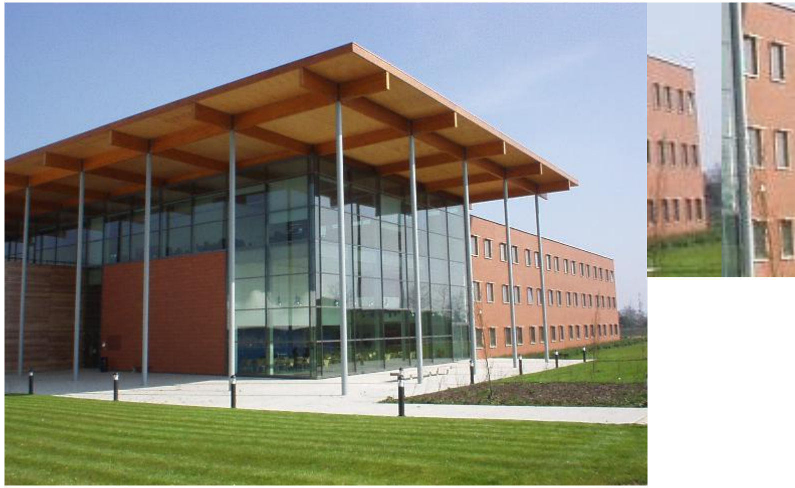
Introduction to Computer Graphics

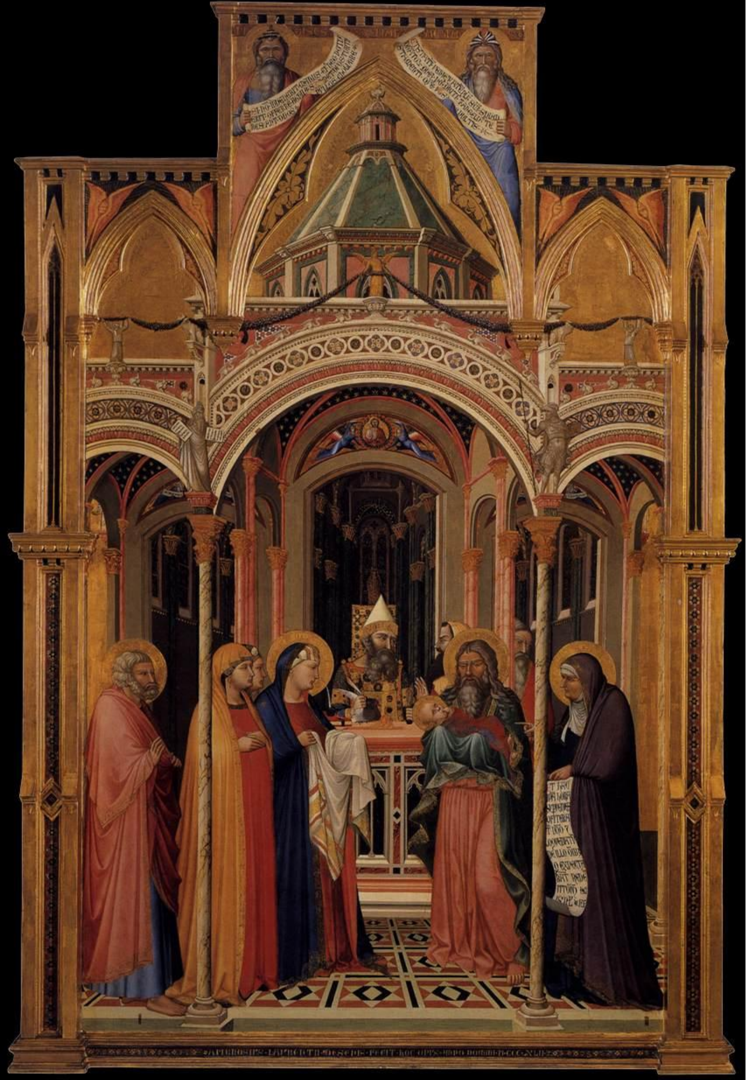
- Background
- **Rendering**
  - **the process of generating realistic or stylish images from a 3D model of digital scenes**
  - Perspective
  - Ray tracing
  - Reflection of light from surfaces and shading
- Graphics pipeline
- Graphics hardware and modern OpenGL
- Human vision and colour & tone mapping

Perspective in photographs


Gates Building – the rounded version (Stanford)


Gates Building – the rectilinear version (Cambridge)

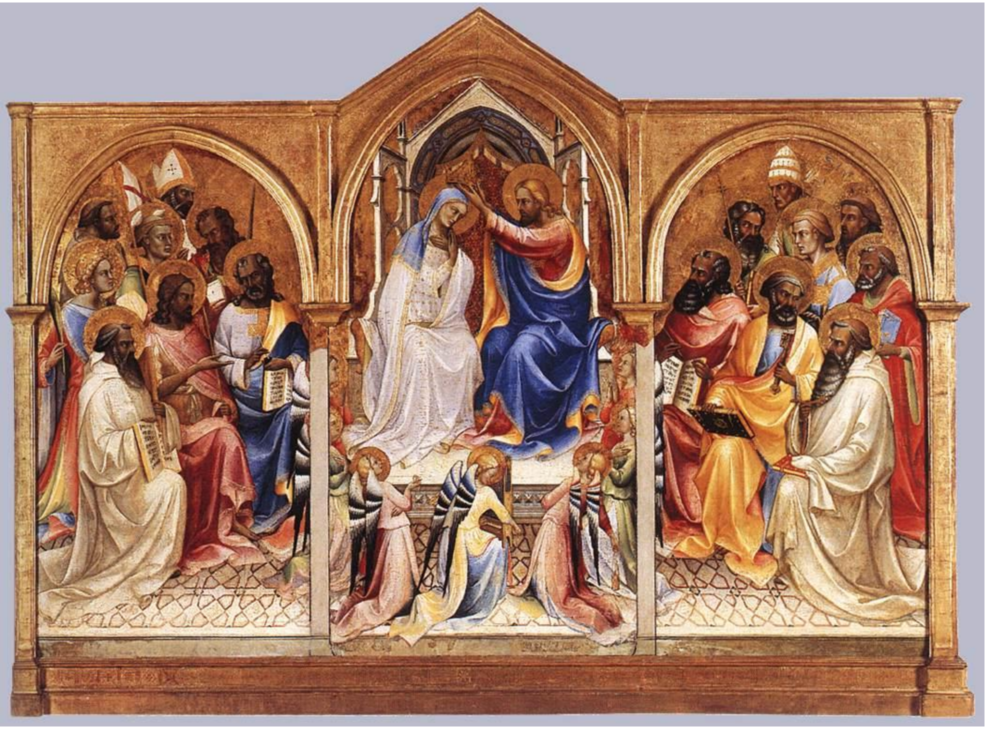
Linear perspective — represent parallel lines converging as they recede into the distance, typically towards a vanishing point on the horizon line, creating the illusion of depth and three-dimensionality on 2D images.



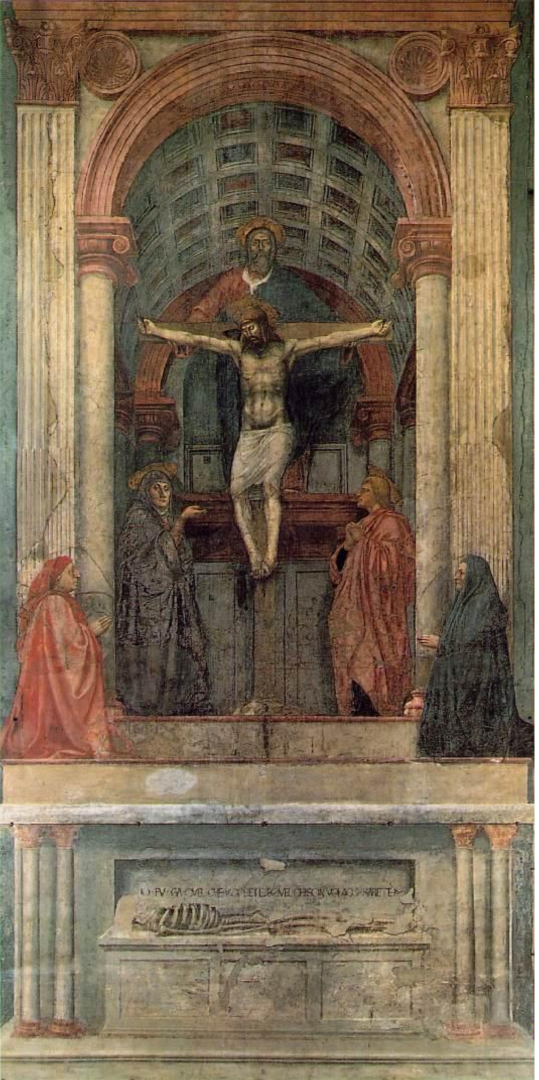
Early perspective

- Presentation at the Temple
- Ambrogio Lorenzetti 1342
- Uffizi Gallery
  Florence

Wrong perspective



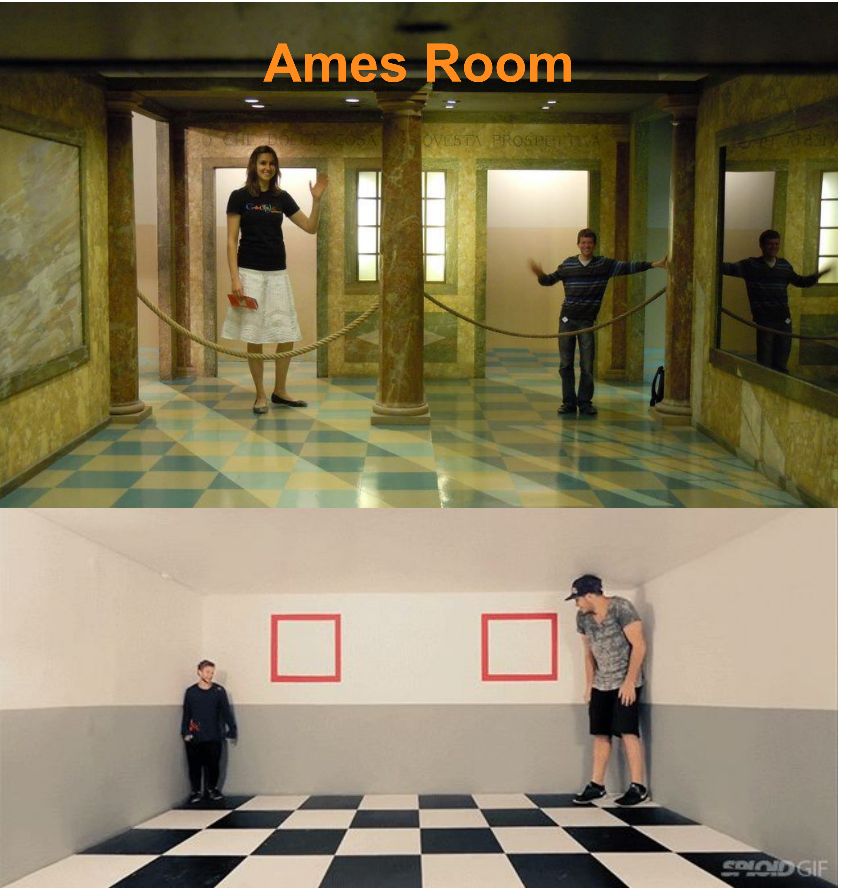
- Adoring saints
- Lorenzo Monaco 1407-09
- National Gallery London

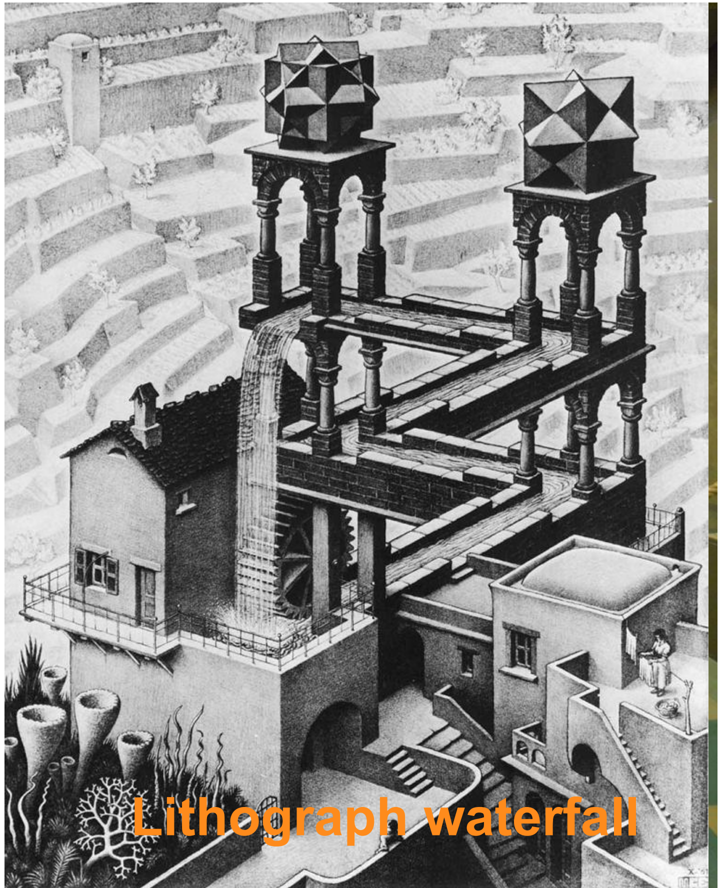


Renaissance perspective

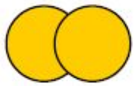
- Geometrical perspective
  Filippo Brunelleschi 1413
- Holy Trinity fresco
- Masaccio (Tommaso di Ser Giovanni di Simone) 1425
- Santa Maria Novella
  Florence
- *De pictura* (On painting)
  textbook by Leon Battista Alberti
  1435

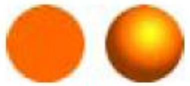
False perspective



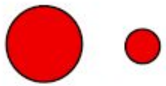


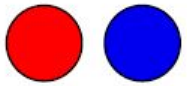
Depth cues

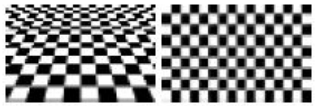

Occlusion

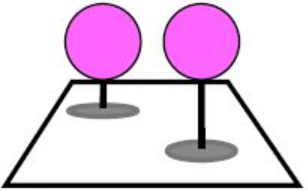

Shading

Familiar Size


Relative Size


Colour

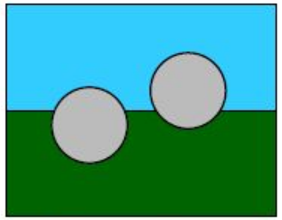

Texture Gradient


Shadow and Foreshortening

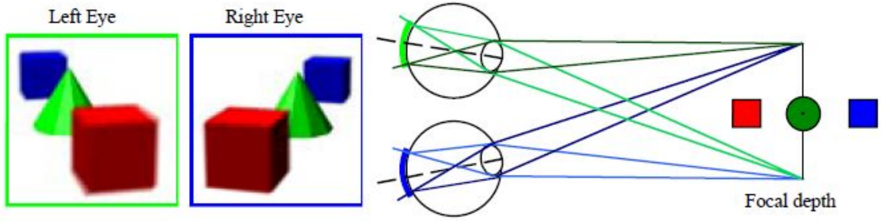
Relative Brightness

Atmosphere

Focus

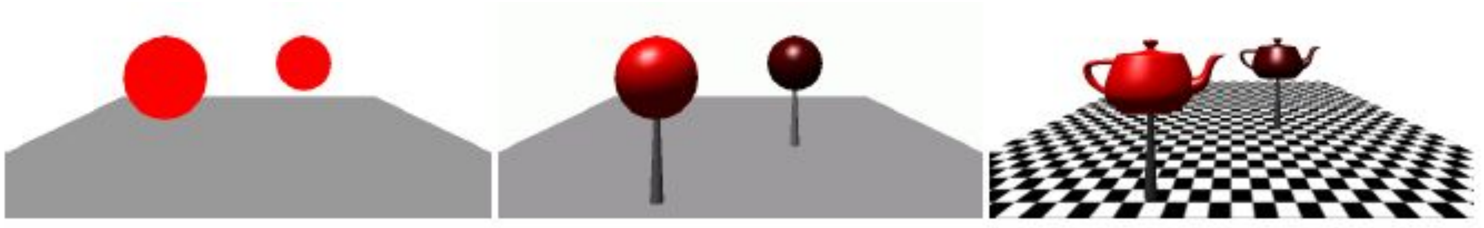

Distance to Horizon

Monocular depth cues

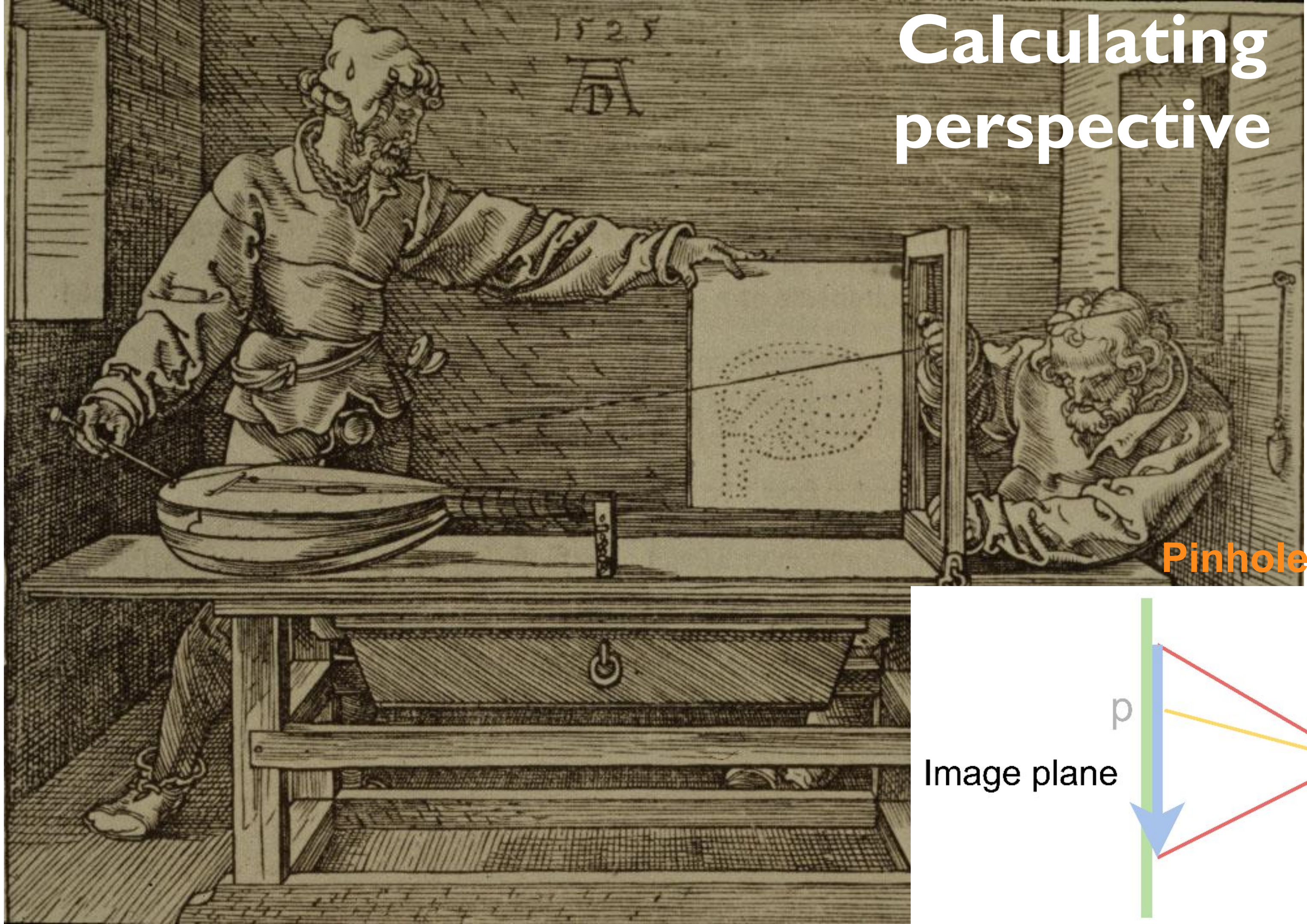


Binocular depth cues

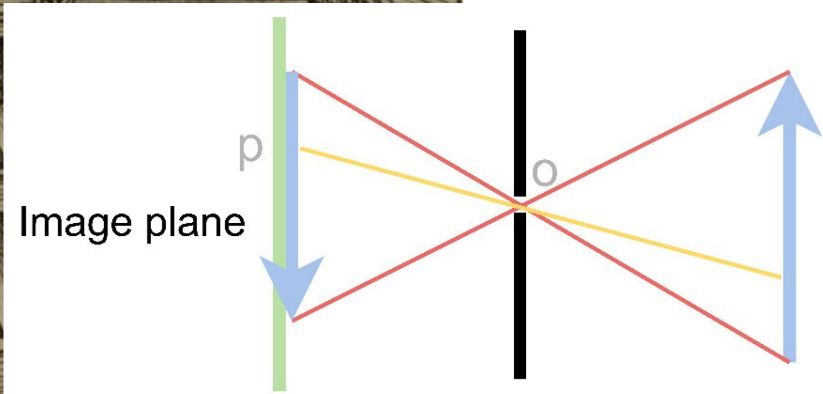
Rendering depth



Calculating perspective

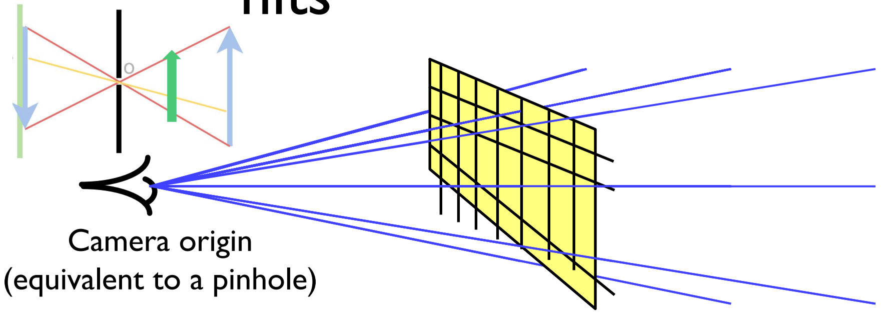


Pinhole camera model

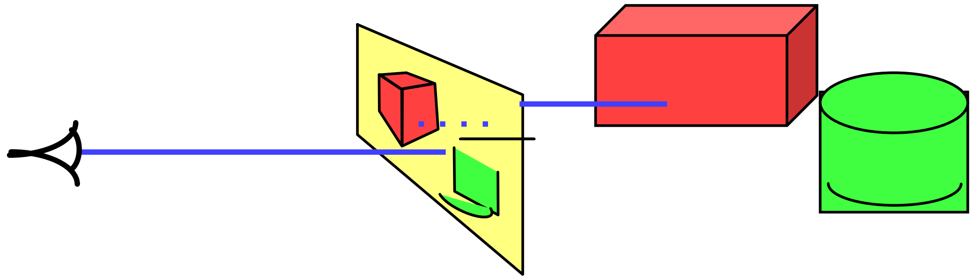


Ray tracing

- Identify point on surface and calculate illumination
- Given a set of 3D objects, shoot a ray from the eye through the centre of every pixel and see what surfaces it hits



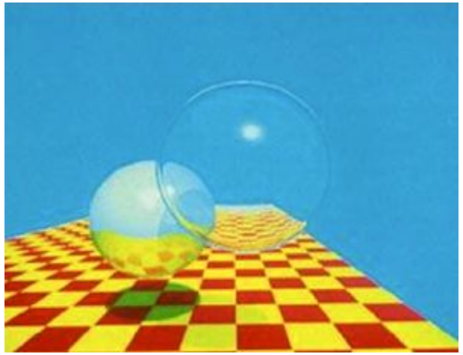
shoot a ray through each pixel

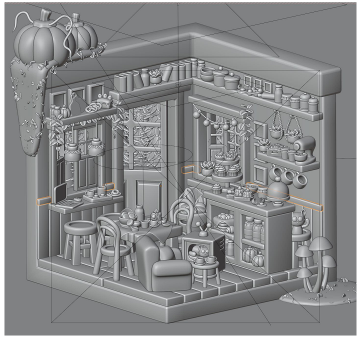


whatever the ray hits determines the colour of that pixel

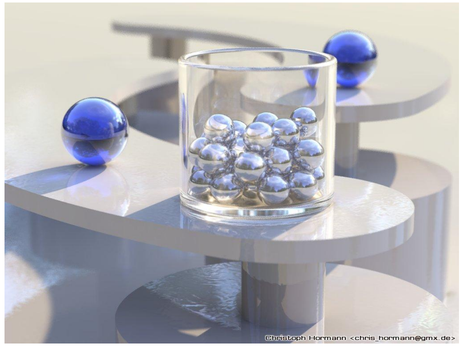
[FCG 4/4]

Ray tracing: examples


Turner Whitted 1979



Cozy Kitchen by Nicole Morena



Ray tracing produces mathematically correct perspective while easily handling reflection, refraction, shadows and blur (due to motion and optics)

Ray tracing is computationally expensive

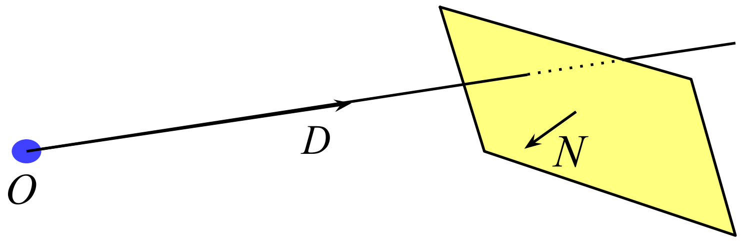
Ray tracing algorithm

```
select a camera origin (eye point) and a screen plane

FOR every pixel in the screen plane
    determine the ray from the eye through the pixel's centre
    FOR each object in the scene
        IF the object is intersected by the ray
            IF the intersection is the closest (so far) to the eye
                record intersection point and object
            END IF ;
        END IF ;
    END FOR ;
    calculate colour for the closest intersection point (if any)
END FOR ;
```

Intersection of a ray with an object I

- plane



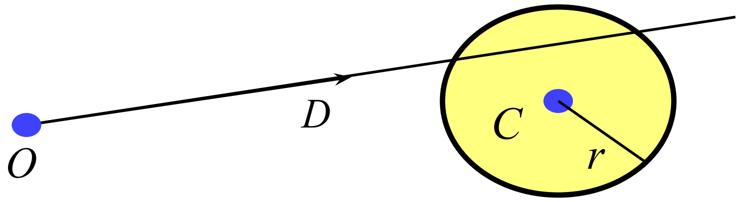
ray: $P = O + sD\ ,\ \ s \geq 0$
plane: $P \cdot N + d = 0$

$$s = -\frac{d + N \cdot O}{N \cdot D}$$

- polygon or disc
  - intersection the ray with the plane of the polygon
    - as above
  - then check to see whether the intersection point lies inside the polygon
    - a 2D geometry problem (which is simple for a disc)

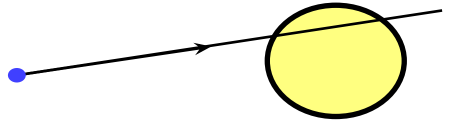
Intersection of a ray with an object 2

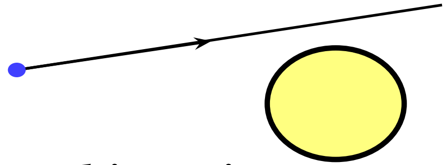
- sphere



ray: $P = O + sD, \quad s \geq 0$
sphere: $(P - C) \cdot (P - C) - r^2 = 0$

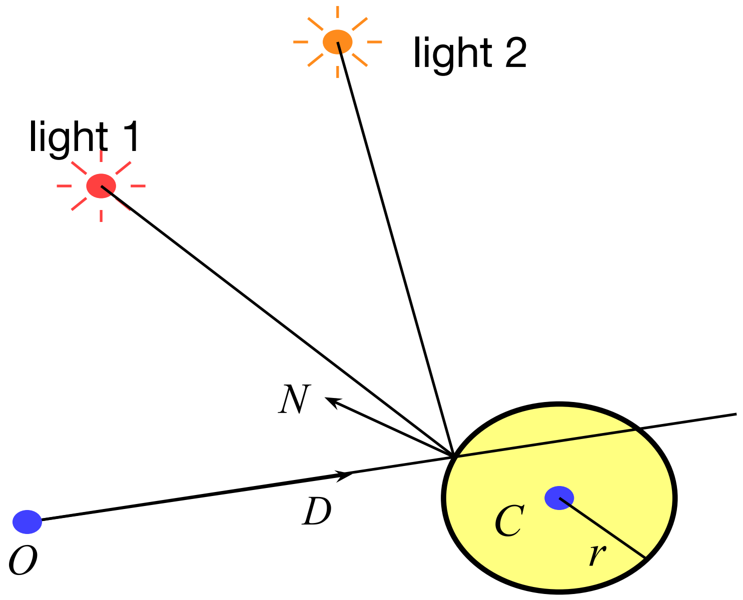
$$
\begin{aligned}
a &= D \cdot D \\
b &= 2D \cdot (O - C) \\
c &= (O - C) \cdot (O - C) - r^2 \\
d &= \sqrt{b^2 - 4ac} \\
s_1 &= \frac{-b + d}{2a} \\
s_2 &= \frac{-b - d}{2a}
\end{aligned}
$$


d real


d imaginary

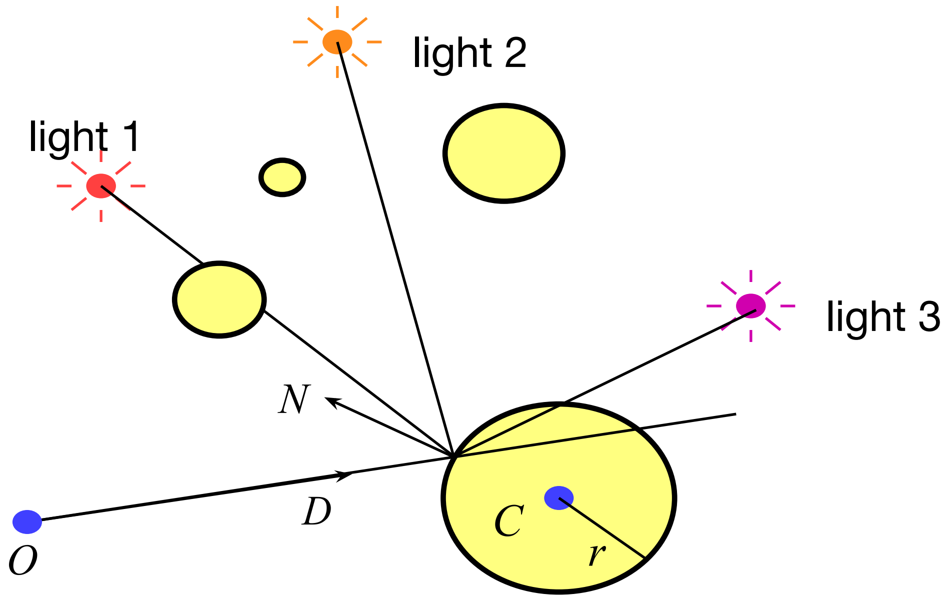
- cylinder, cone, torus
  - all similar to sphere
  - try them as an exercise

Ray tracing: shading



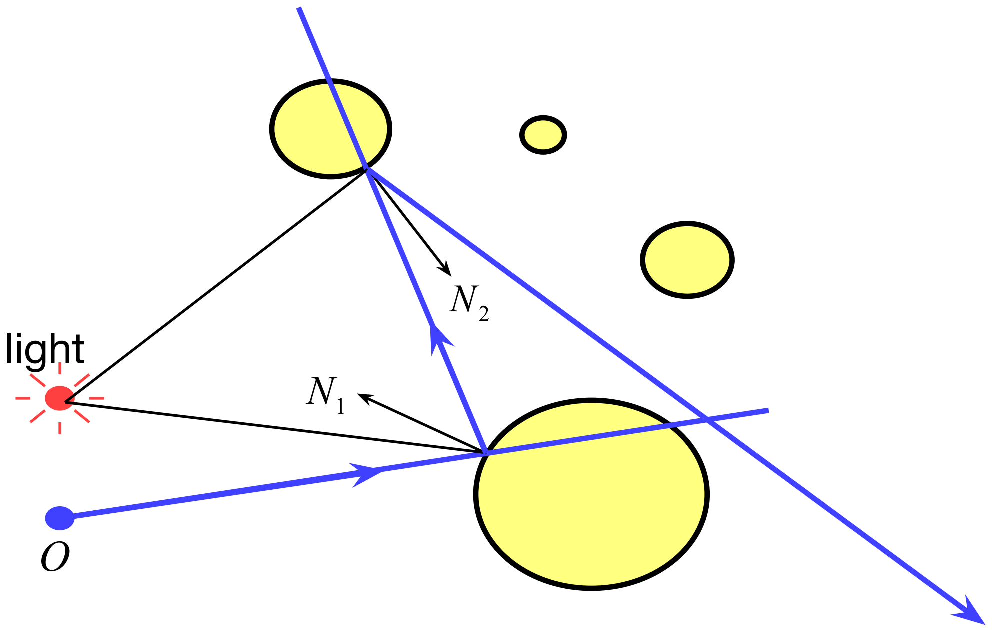
- **Shading**
  - computing the appearance of the surface under an illumination based on how light interacts with the surface
- once you have the intersection of a ray with the nearest object you can:
  - calculate the normal to the object at that intersection point
  - shoot rays from that point to all of the light sources, and calculate the **diffuse** and **specular** reflections off the object at that point
    - this (plus **ambient** illumination) gives the colour of the object (at that point)

Ray tracing: shadows



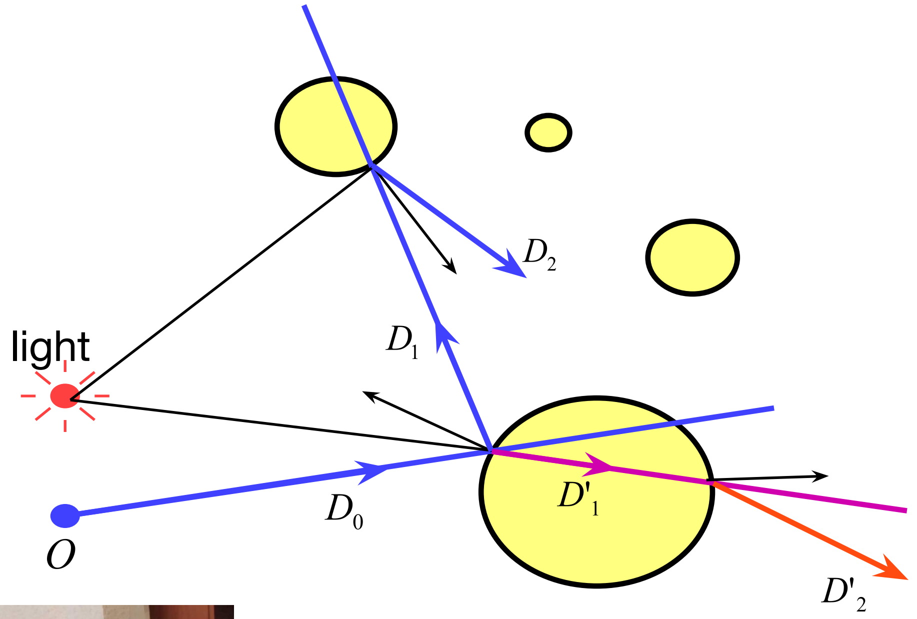
- because you are tracing rays (**shadow rays**) from the intersection point to the light, you can check whether another object is between the intersection and the light and is hence casting a shadow
  - also need to watch for self-shadowing

Ray tracing: reflection



- if a surface is totally or partially reflective then new rays can be spawned to find the contribution to the pixel's colour given by the reflection
  - this is perfect (mirror) reflection

Ray tracing: transparency & refraction



Example of a refraction

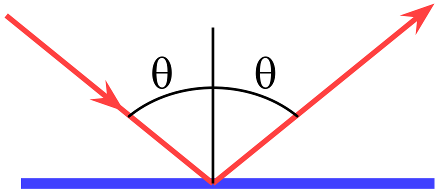
- objects can be totally or partially transparent
  - this allows objects behind the current one to be seen through it
- transparent objects can have refractive indices
  - bending the rays as they pass through the objects
- transparency + reflection means that a ray can split into two parts

Illumination and shading

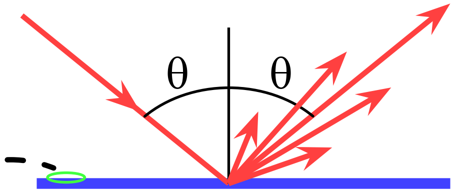
- Visibility check (Dürer’s method) allows us to calculate what part of the scene is visible in each pixel
- But what colour should it be?
- Depends on:
  - normal
  - lighting
  - shadows
  - properties of surface material

[FCG 4.5-4.8/5]

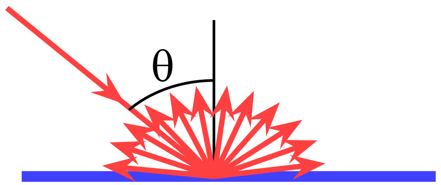
How do surfaces reflect light?



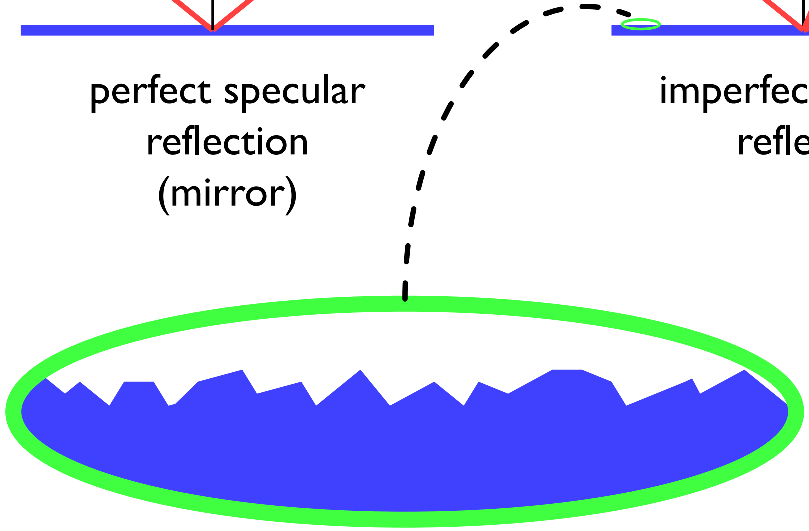
perfect specular reflection (mirror)



imperfect specular reflection



diffuse reflection (Lambertian reflection)

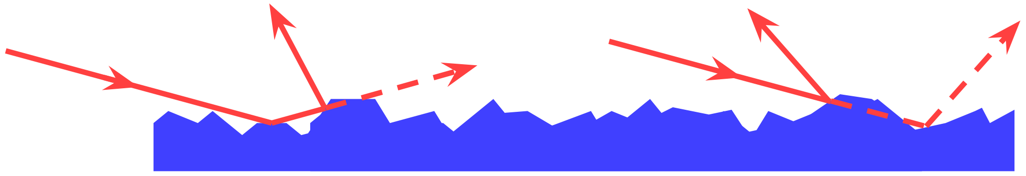


the surface of a specular reflector is facetted, each facet reflects perfectly but in a slightly different direction to the other facets

Johann Lambert, 18th century German mathematician

Comments on reflection

- surface can absorb/reflect certain wavelengths of light to exhibit **colours**
  - e.g. shiny gold or shiny copper
- specular reflection has "interesting" properties at glancing angles due to occlusion of micro-facets by one another

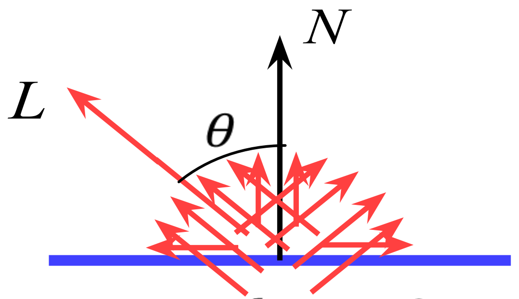


- plastics are good examples of surfaces with:
  - specular reflection in the light's colour
  - diffuse reflection in the plastic's colour
- simplified model
  - physics-based reflection model in Further Graphics

Phong shading with direct illumination

- Phong reflection model
  - diffuse (Lambertian) reflection + (imperfect) specular reflection + ambient reflection (approximating indirect illuminations)
- Direct illumination
  - all light falling on a surface comes directly from a light source
    - there is no interaction between objects
  - no object casts shadows on any other
    - so can treat each surface as if it were the only object in the scene
  - light sources are considered to be infinitely distant from the object (sunlight)
    - light intensity is the same across the whole scene

Diffuse shading calculation



$$\begin{aligned} I &= I_l k_d \cos\theta \\ &= I_l k_d (N \cdot L) \end{aligned}$$

L is a normalised unit vector pointing in the direction of the light source

N is the normal to the surface

I_l is the intensity of the light source

k_d is the proportion of light which is diffusely reflected by the surface

I is the intensity of the light reflected by the surface

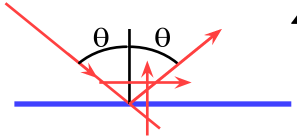
use this equation to calculate the diffuse colour of a pixel

- observation:
  - the colour of a flat surface will be uniform across it, dependent only on the colour & position of the object and the colour & position of the light sources

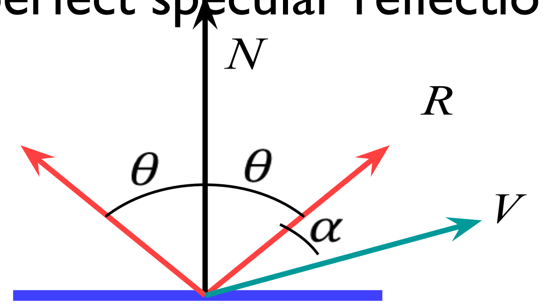
Diffuse shading: comments

- not depending on the viewing angle!
- can have different I_l and different k_d for different wavelengths (colours)
- watch out for $\cos\theta < 0$
  - implies that the light is behind the polygon and so it cannot illuminate this side of the polygon
- do you use one-sided or two-sided surfaces?
  - one sided: only the side in the direction of the normal vector can be illuminated
    - if $\cos\theta < 0$ then both sides are black
  - two sided: the sign of $\cos\theta$ determines which side of the polygon is illuminated
    - need to invert the sign of the intensity for the back side
- this is essentially a simple one-parameter (θ) BRDF
  - Bidirectional Reflectance Distribution Function — Further Graphics

Imperfect specular reflection



- Phong developed an easy-to-calculate *approximation* to imperfect specular reflection



$$I = I_l k_s \cos^n \alpha$$
$$= I_l k_s (R \cdot V)^n$$

L is a normalised unit vector pointing in the direction of the light source

R is the vector of perfect reflection

N is the normal to the surface

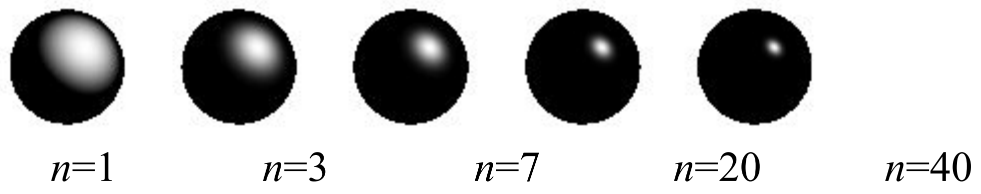
V is a unit vector pointing at the viewer

I_l is the intensity of the light source

k_s is the proportion of light which is specularly reflected by the surface

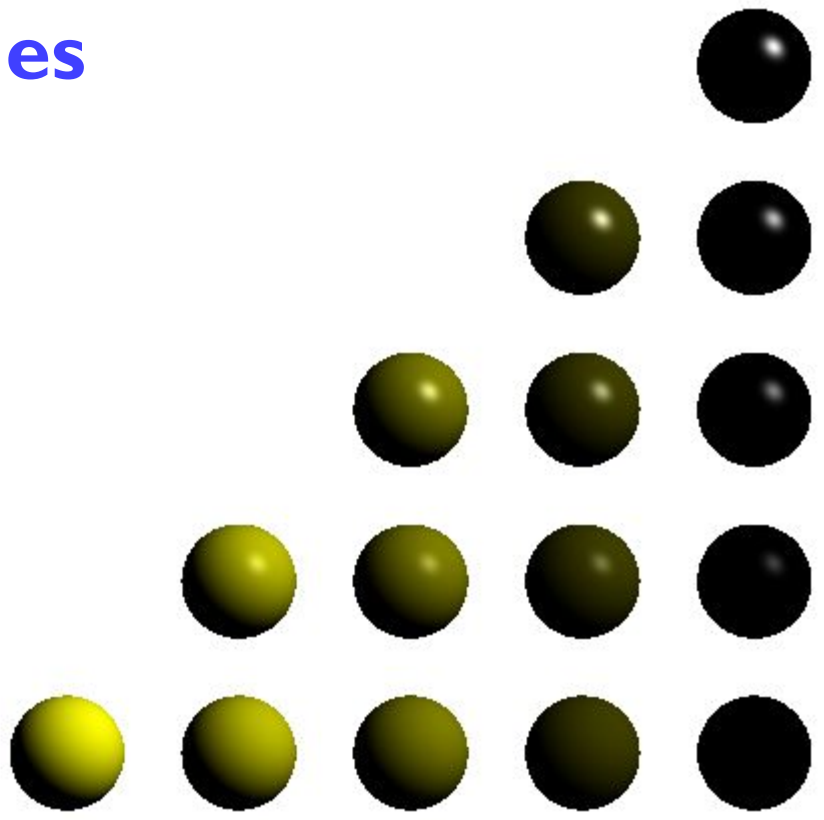
n is Phong's *ad hoc* "roughness" coefficient

I is the intensity of the specularly reflected light



Phong Bui-Tuong, "Illumination for computer generated pictures", *CACM*, **18**(6), 1975, 311–7

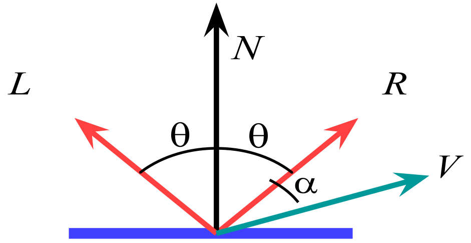
Examples


100%
75%
50%
25%
0%
specular reflection
100%
75%
50%
25%
0%
diffuse reflection

Phong shading: overall equation

- The overall shading equation can thus be considered to be the ambient illumination plus the diffuse and specular reflections from each light source

$$I = I_a k_a + \sum_i I_i k_\mathrm{d}(L \cdot N) + \sum_i I_i k_\mathrm{s}(R \cdot V)^n$$



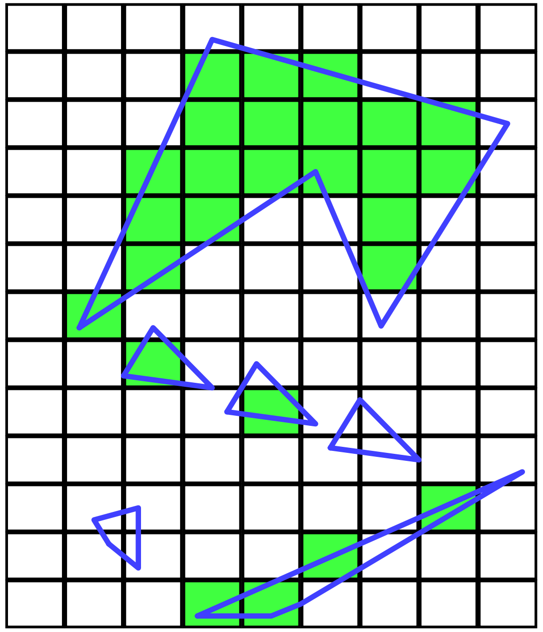
  - The equation above is computed for each colour channel
  - The more lights there are in the scene, the longer this calculation will take

The gross assumptions revisited

- Phong: diffuse reflection and approximate specular reflection
- no shadows
  - need to cast shadow rays (ray tracing) or shadow mapping (rasterization) to obtain shadows
- lights at infinity
  - can add local lights at the expense of more calculation
- no interaction between surfaces
  - cheat!
    - assume that all light reflected off all other surfaces onto a given surface can be amalgamated into a single constant term: "ambient illumination", add this onto the diffuse and specular illumination

Sampling

- we have assumed so far that each ray passes through the centre of a pixel
  - i.e. the value for each pixel is the colour of the object which happens to lie exactly under the centre of the pixel
- this leads to:
  - stair step (jagged) edges to objects
  - small objects being missed completely
  - thin objects being missed completely or split into small pieces

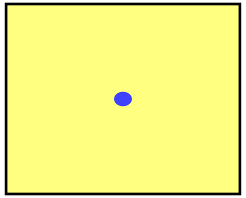


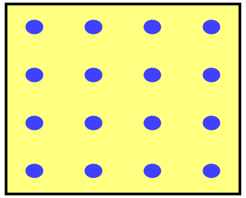
Anti-aliasing

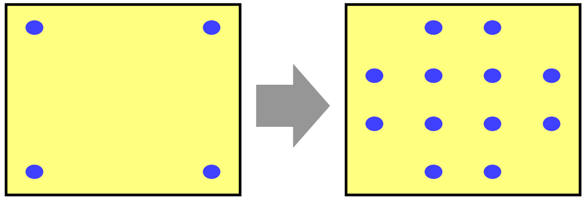
- these artefacts (and others) are jointly known as **aliasing**
  - smooth edges or details appear to be discontinuous or distorted due to insufficient sampling rate in digital images
  - in signal processing aliasing is universally considered for all types of signals, e.g., images, audio, and more
- methods of ameliorating the effects of aliasing are known as **anti-aliasing**

Sampling in ray tracing

- single point
  - shoot a single ray through the pixel's centre
- super-sampling for anti-aliasing
  - shoot multiple rays through the pixel and average the result
  - regular grid, random, jittered, Poisson disc
- adaptive super-sampling
  - shoot a few rays through the pixel, check the variance of the resulting values, if similar enough stop, otherwise shoot some more rays

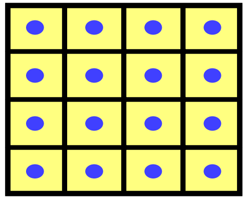
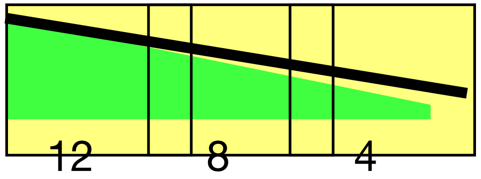


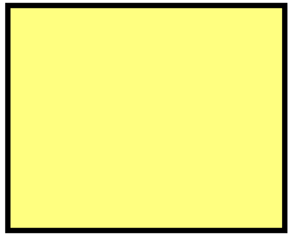




Types of super-sampling 1

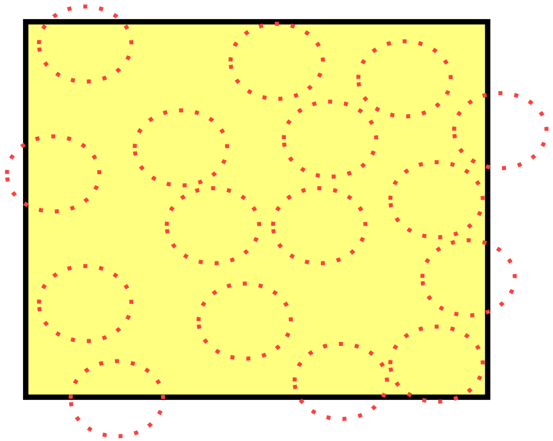
- regular grid
  - divide the pixel into a number of sub-pixels and shoot a ray through the centre of each
  - problem: can still lead to noticable aliasing unless a very high resolution sub-pixel grid is used
- random
  - shoot N rays at random points in the pixel
  - replaces aliasing artefacts with noise artefacts
    - the eye is far less sensitive to noise than to aliasing

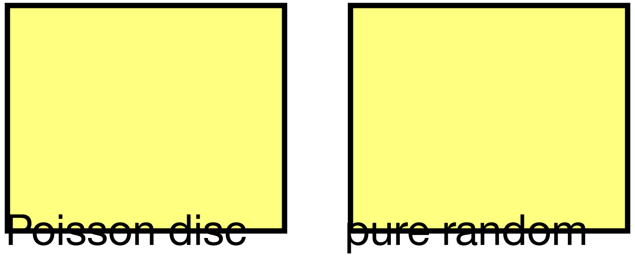





Types of super-sampling 2

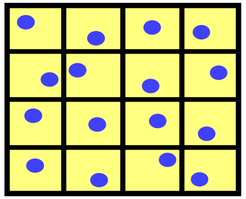
- Poisson disc
  - shoot N rays at random points in the pixel with the proviso that no two rays shall pass through the pixel closer than ε to one another
  - for N rays this produces a better looking image than pure random sampling
  - very hard to implement properly

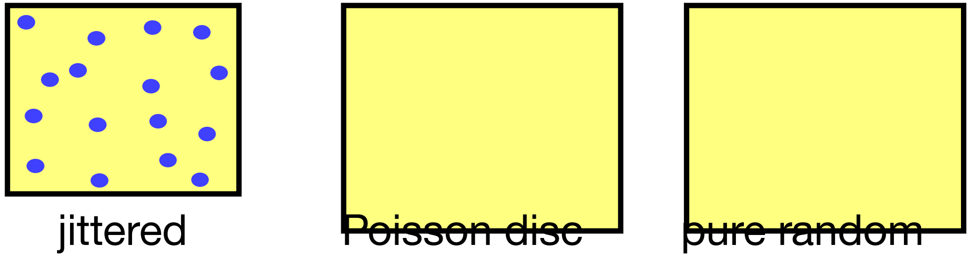




Types of super-sampling 3

- jittered
  - divide pixel into N sub-pixels and shoot one ray at a random point in each sub-pixel
  - an approximation to Poisson disc sampling
  - for N rays it is better than pure random sampling
  - easy to implement

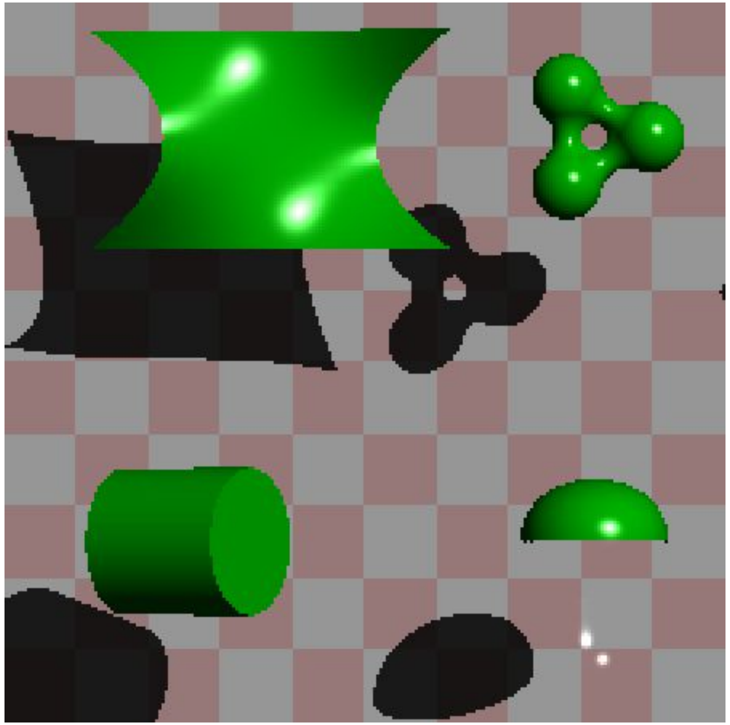




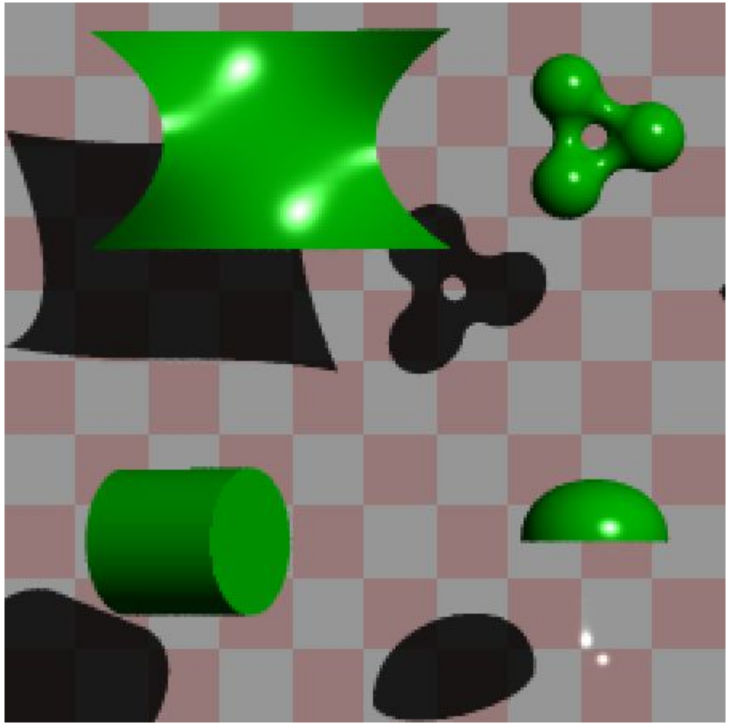
Distributed ray tracing

- super-sampling is only one reason why we might want to take multiple samples per pixel
- many effects can be achieved by distributing multiple samples over certain range
  - called *distributed* ray tracing
    - N.B. *distributed* means distributed over a range of values
- can work in two ways
  1. each of the multiple rays shot through a pixel is allocated a random value from the relevant distribution(s)
     - all effects can be achieved this way with sufficient rays per pixel
  2. each ray spawns multiple rays when it hits an object
     - this alternative can be used, for example, for area lights

Anti-aliasing



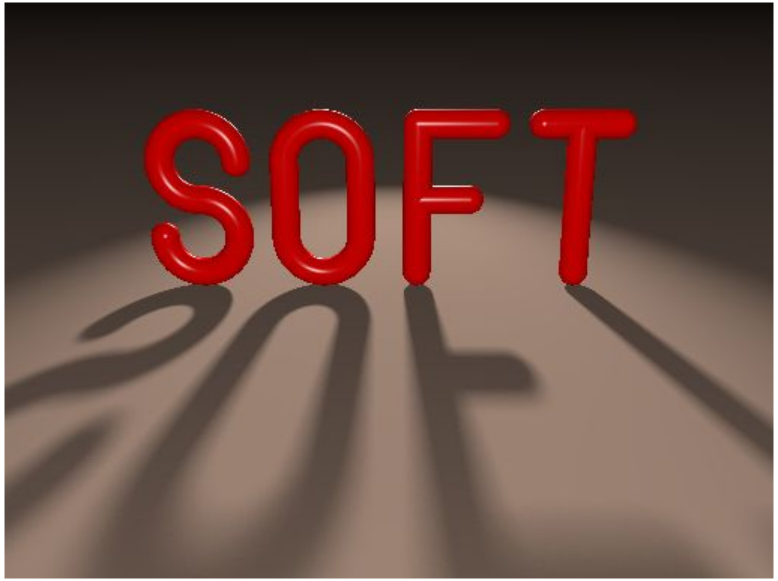
one sample per pixel



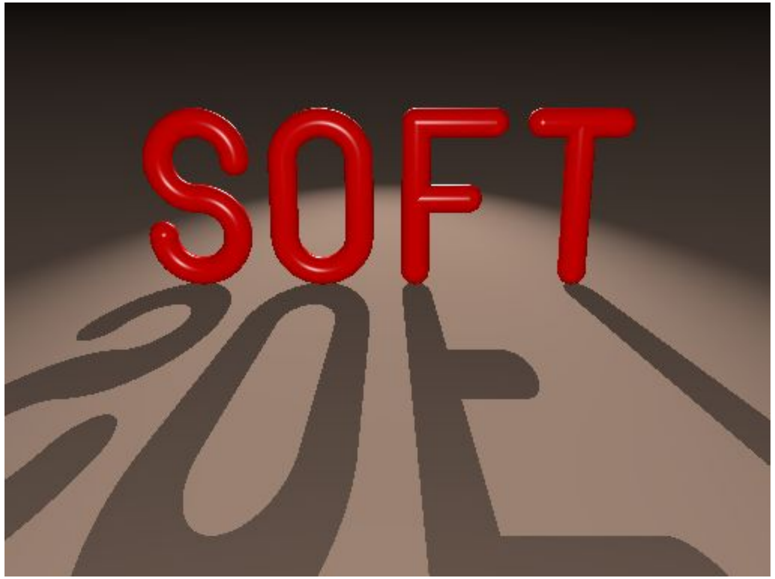
multiple samples per pixel

- distribute the samples for a pixel over the pixel area

Area *vs* point light source



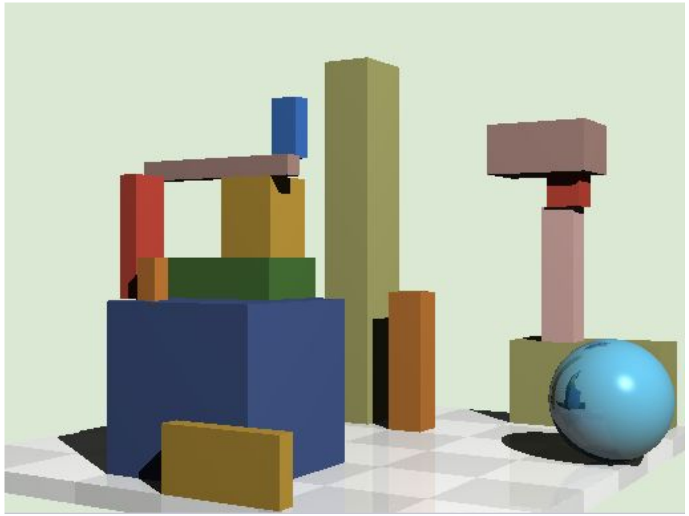
an area light source produces soft shadows



a point light source produces hard shadows

- distribute the rays going to a light source over some area
  - allows area light sources in addition to point and directional light sources
  - produces soft shadows with penumbrae

Finite aperture



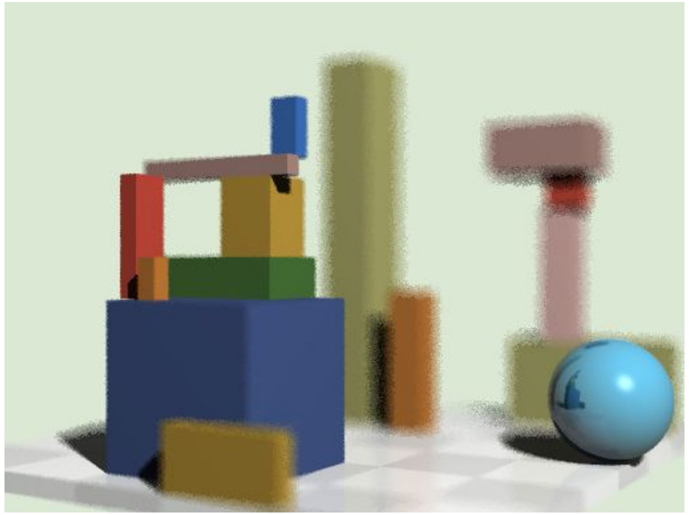
left, a pinhole camera

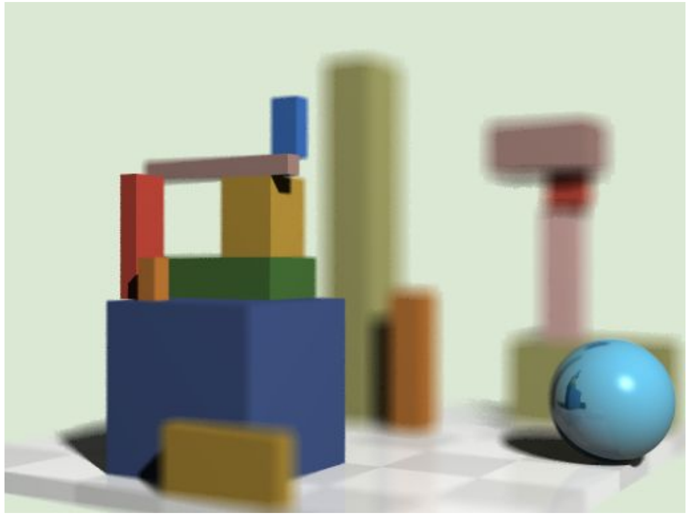
below, a finite aperture camera

below left, 12 samples per pixel

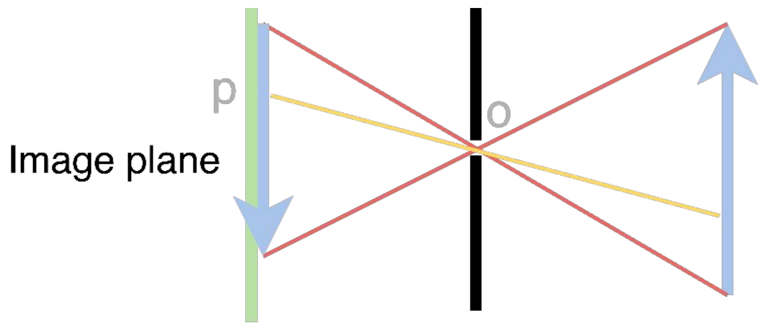
below right, 120 samples per pixel

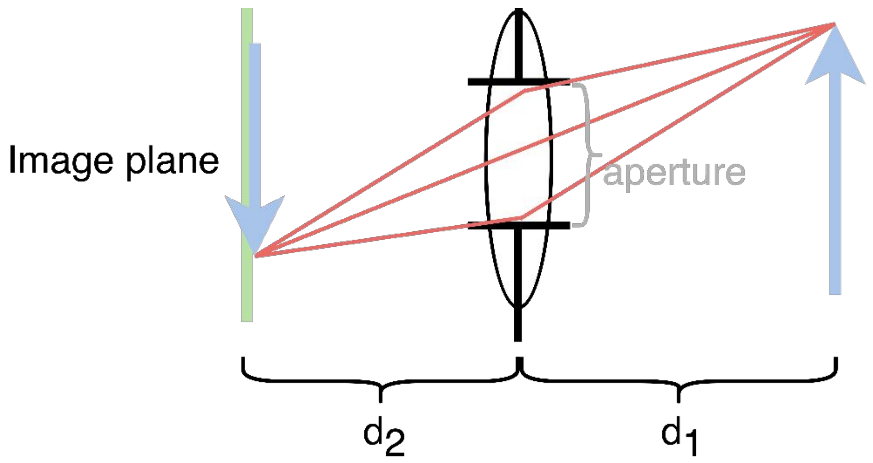
note the depth of field blur: only objects at the correct distance are in focus





Finite aperture





Pinhole camera

Thin-lens camera

- distribute the camera position over an aperture
  - allows simulation of a camera with a finite aperture lens
  - produces depth of field effects