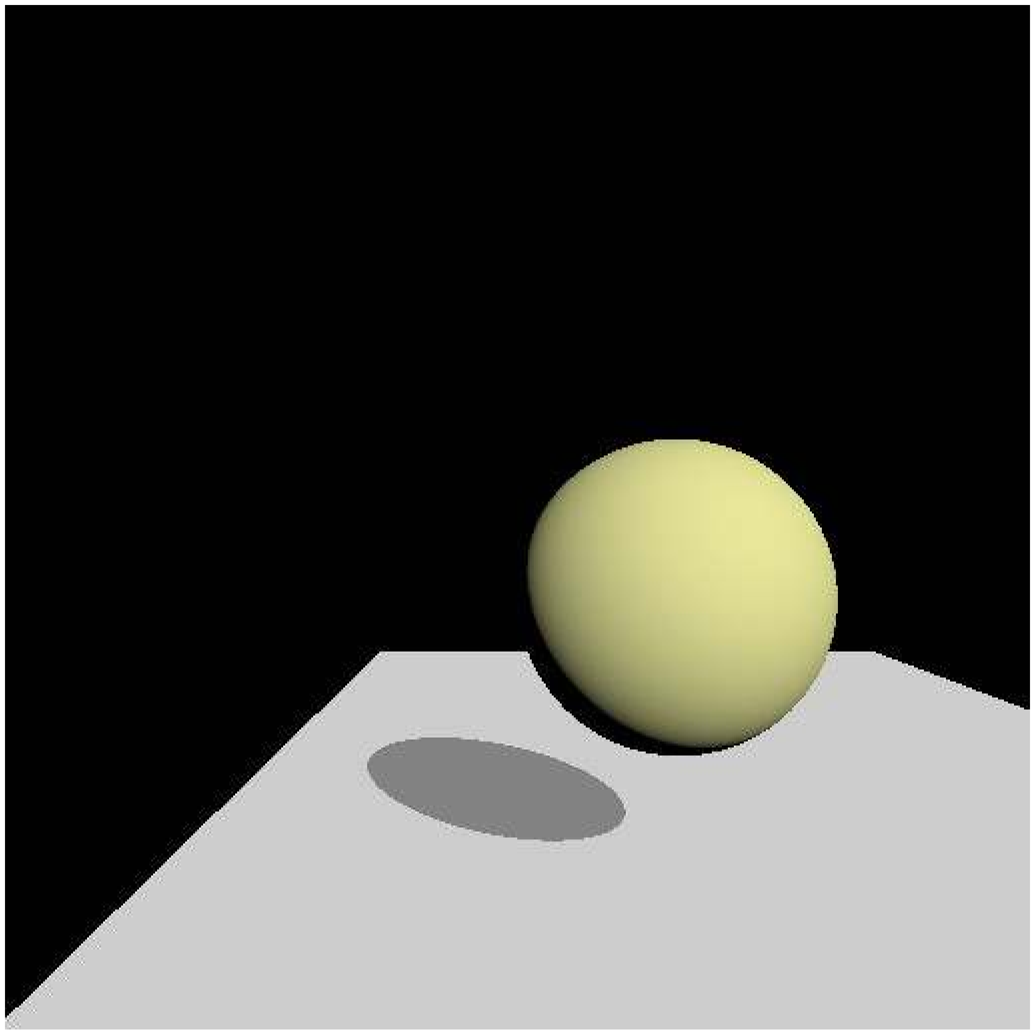
CS 428: Fall 2009

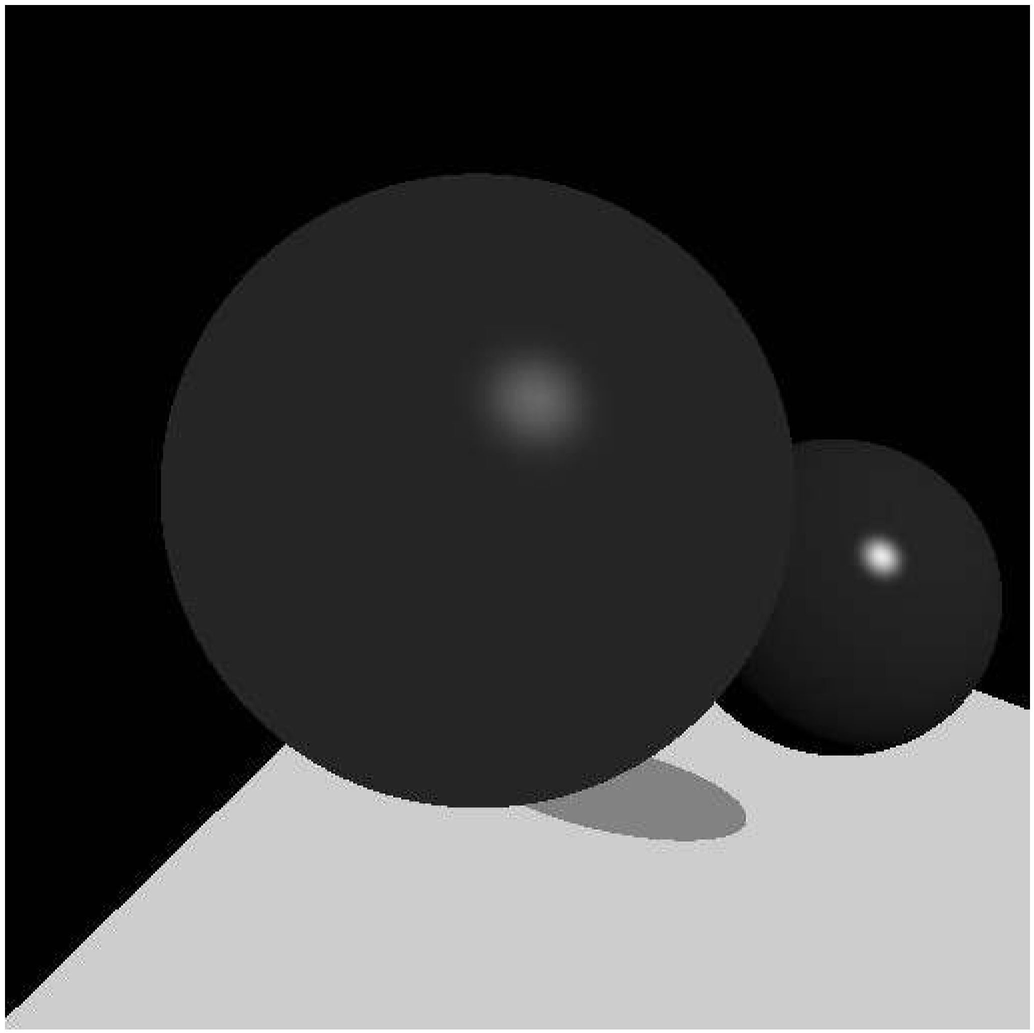
Introduction to Computer Graphics

Raytracing topics

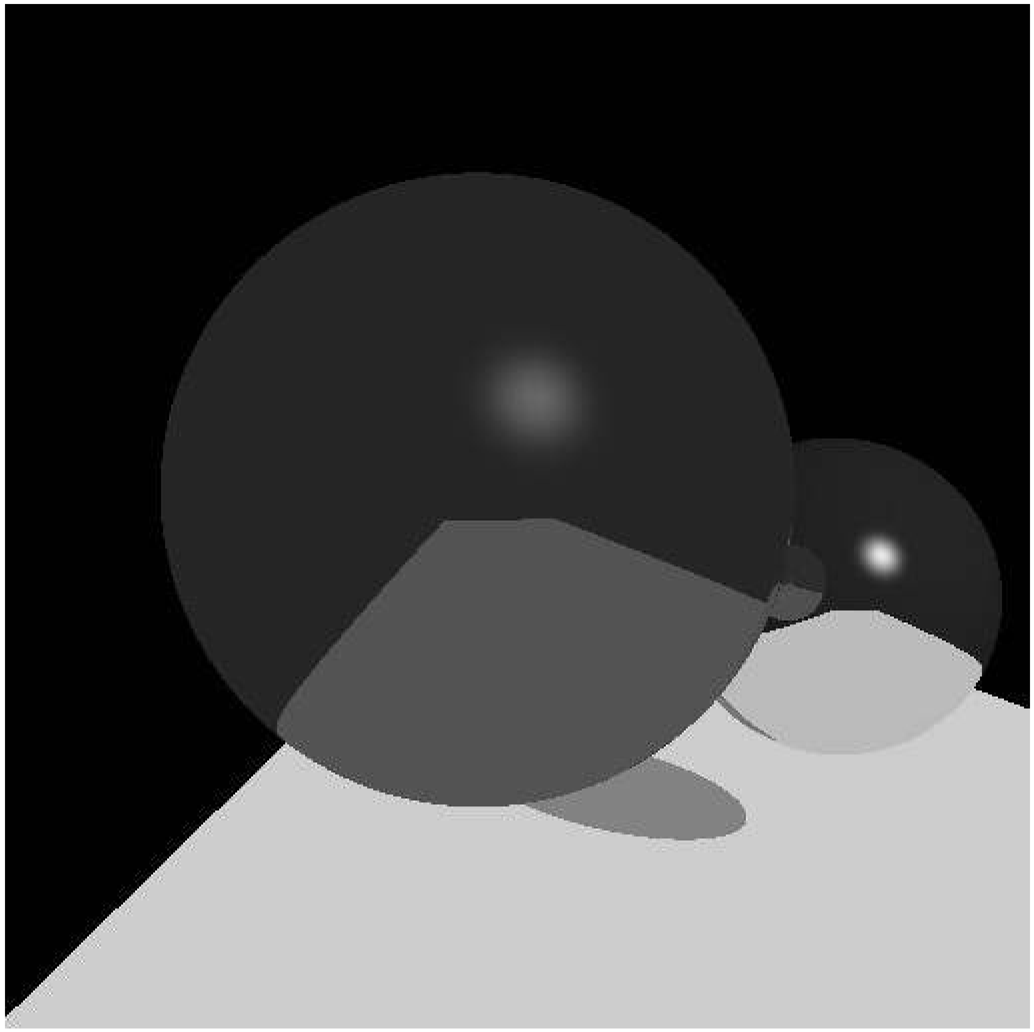
Opaque and non-reflective scene



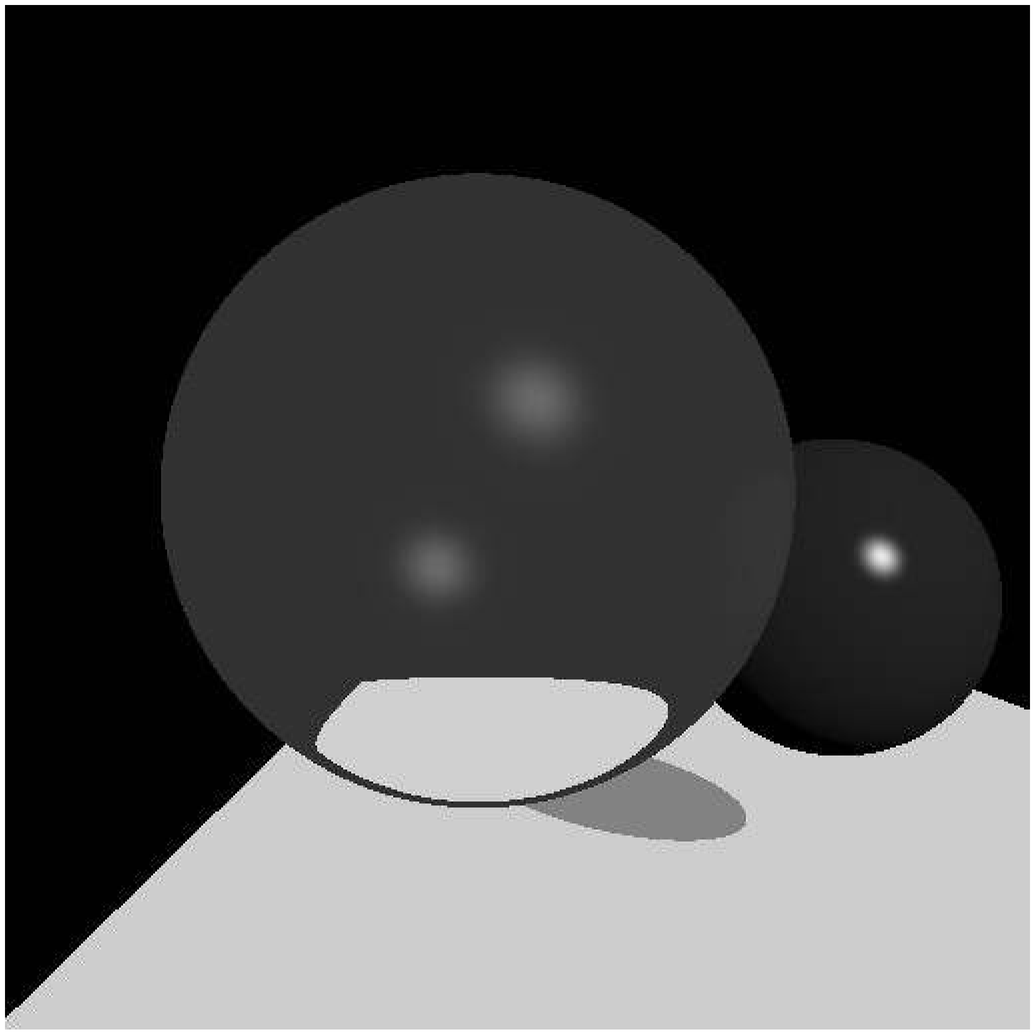
Transparent and reflective scene (non-recursive ray tracer)



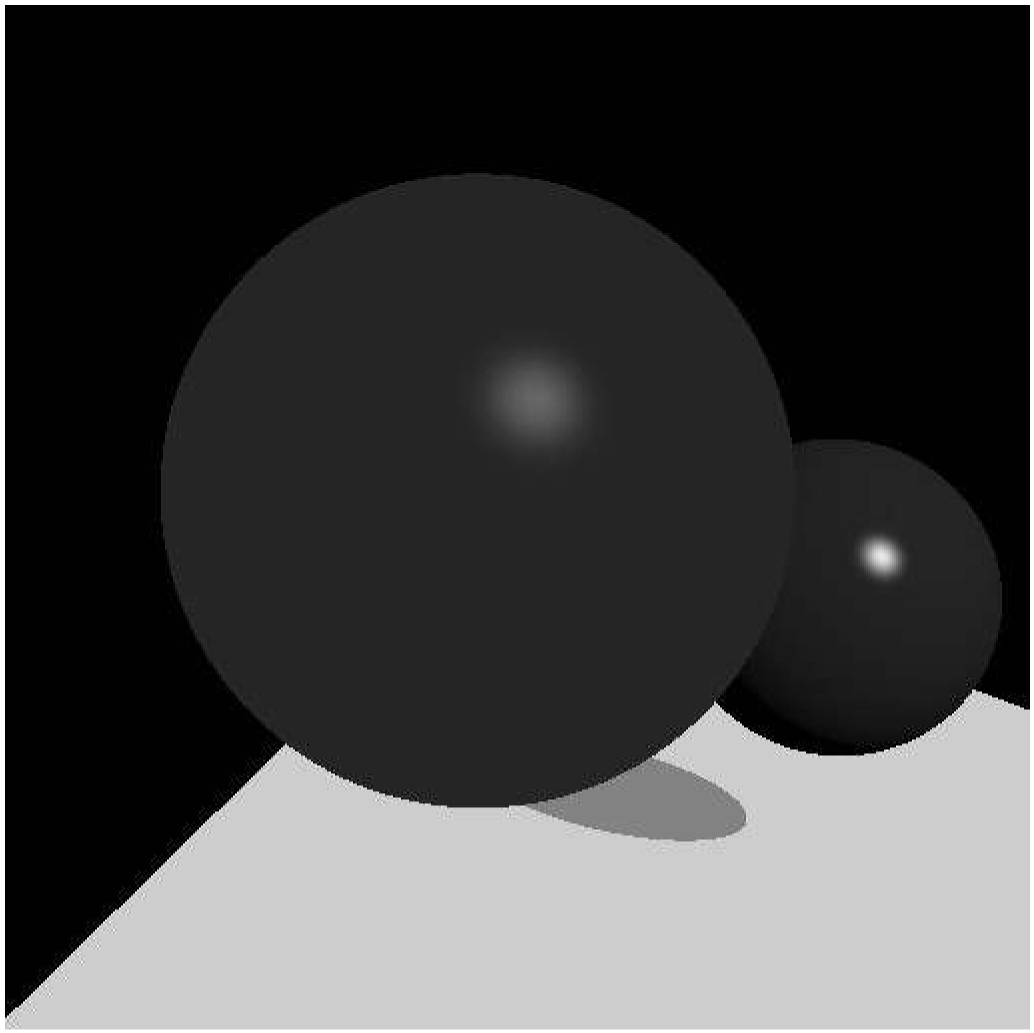
Transparent and reflective scene (reflections only)



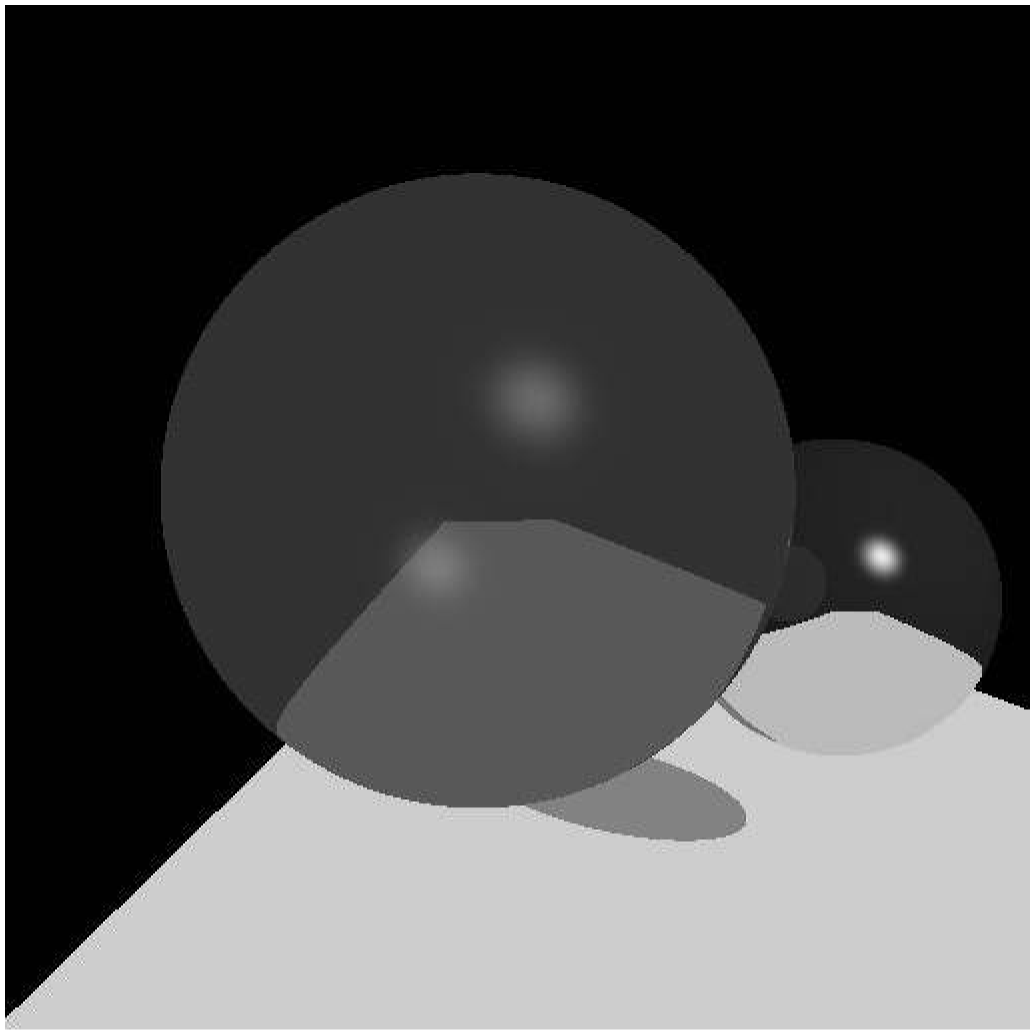
Transparent and reflective scene (refractions only)



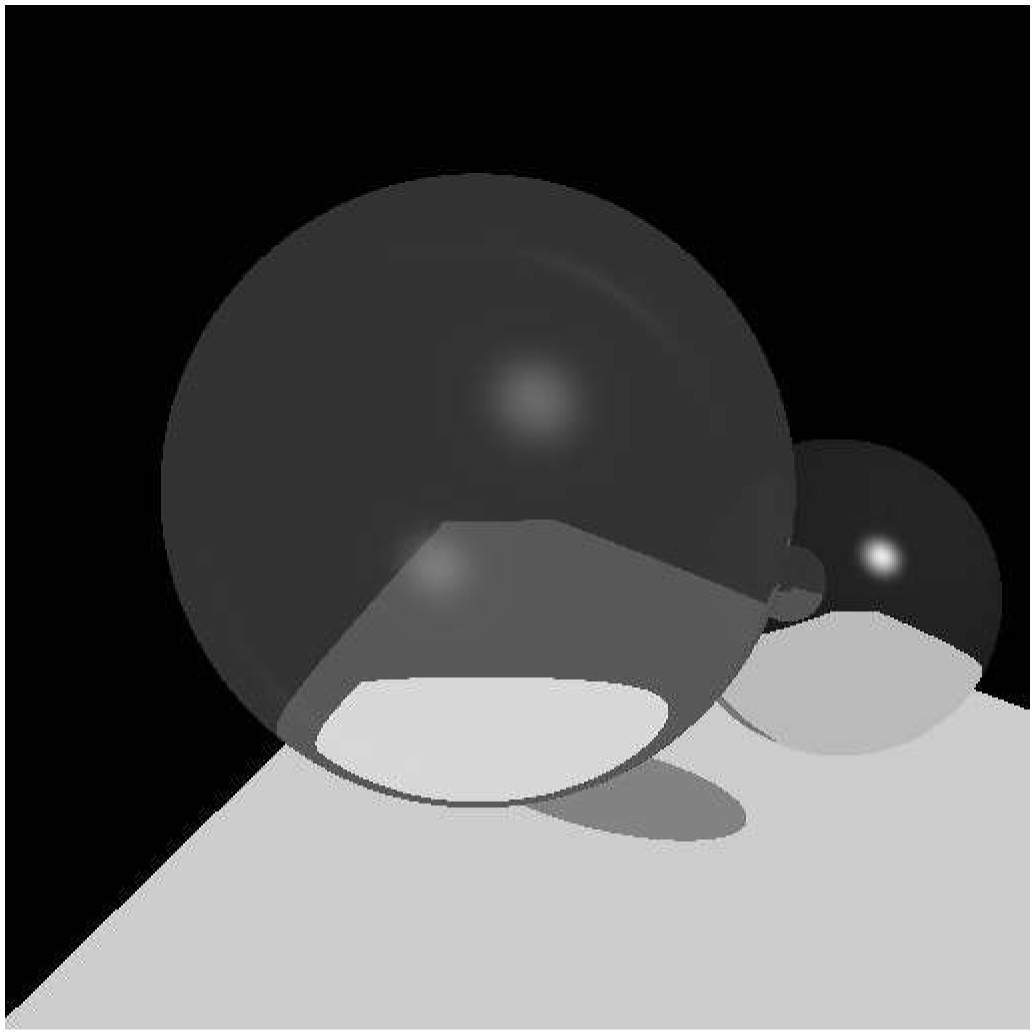
Transparent and reflective scene (recursion level 0)



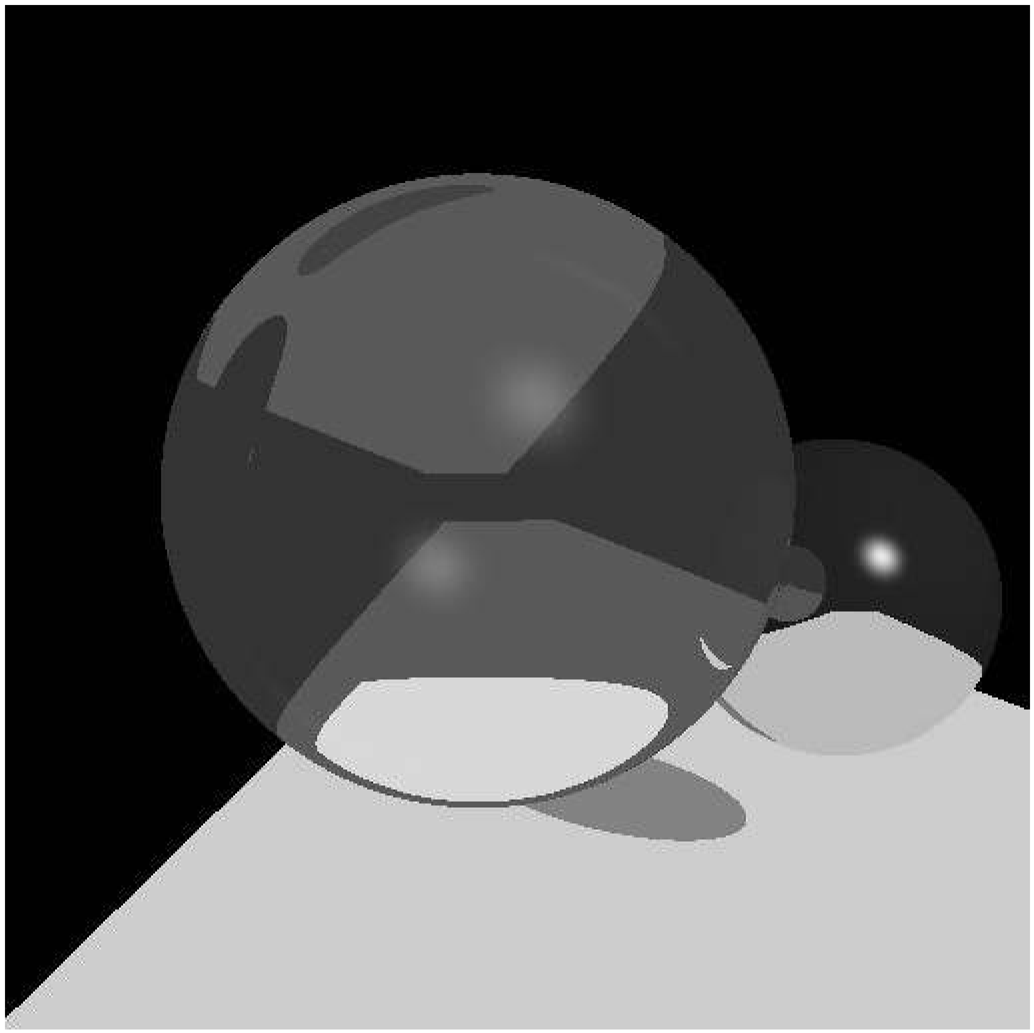
Transparent and reflective scene (recursion level 1)



Transparent and reflective scene (recursion level 2)

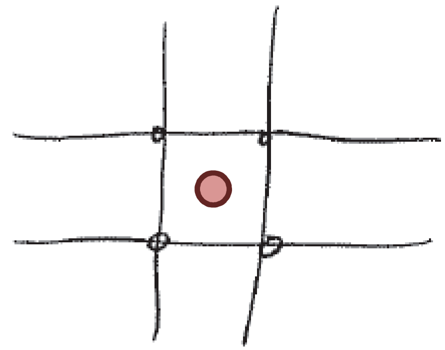


Transparent and reflective scene (recursion level 3)

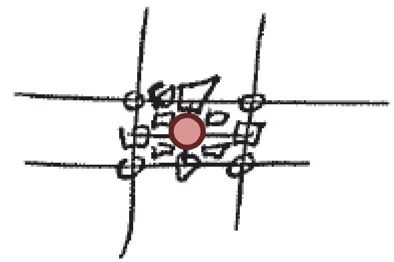


Anti-aliasing

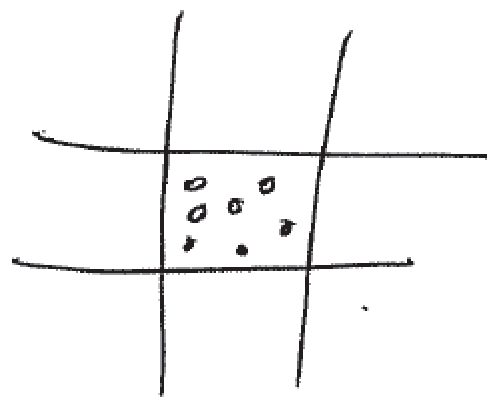
- Subdivision
  - Average 5 rays per pixel (sharing)



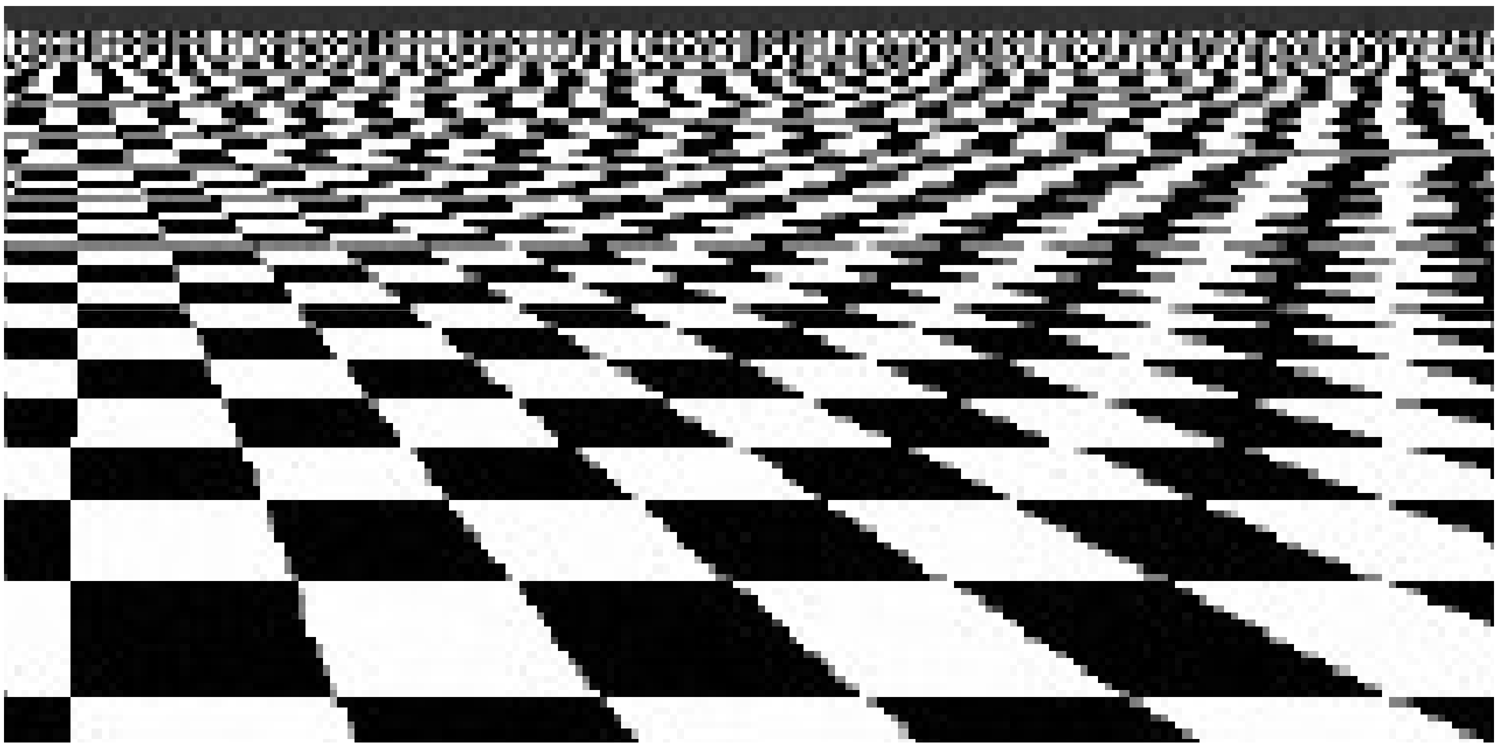
- Adaptive subdivision
  - If 5 rays vary strongly, add samples
  - 8 more rays per split



- Stochastic
  - Evenly spaced, randomly placed
  - Trade quality for noise



Texture aliasing example



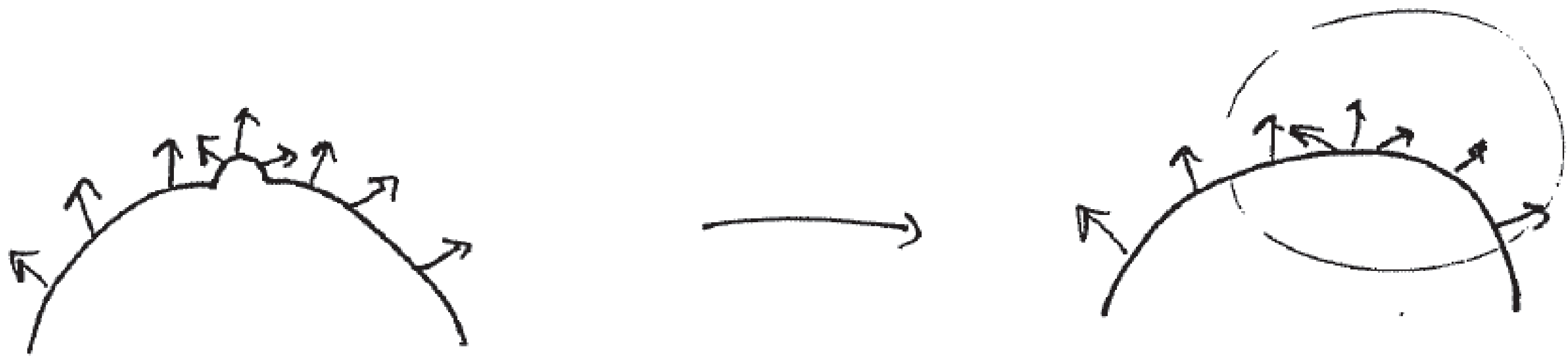
Bump mapping

- Add fine surface detail to enhance realism

- Representing this using implicit functions f(x,y,z) = 0 is possible, but **expensive**
- **Trick:** don't represent actual surfaces, but just be able to find normal vectors
  - Use fake normals in lighting computation

Bump mapping

- **Trick:** don’t represent actual surfaces, but just be able to find normal vectors
  - Use fake normals in lighting computation



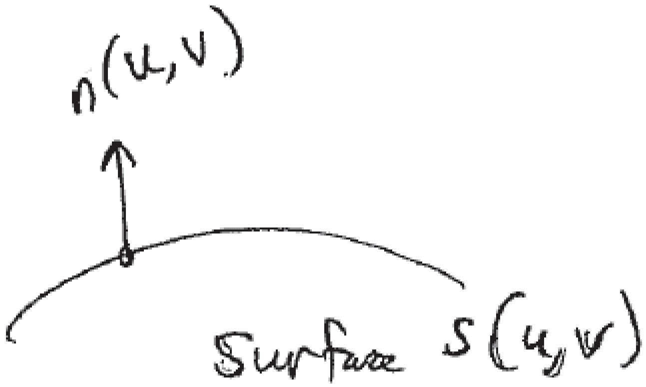
- Looks ok, but the silhouette is not bumpy

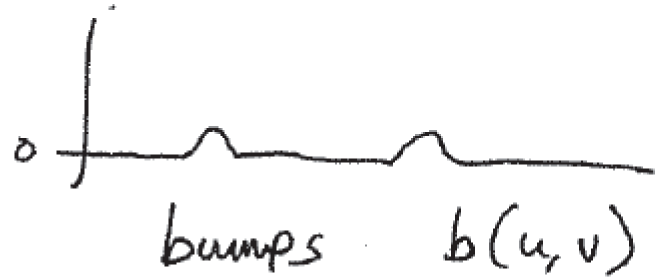
Bump mapping

- Use texture to specify "offset" from surface caused by bumps (in normal direction)
  - This way, bumps stick to the surface

- Find normals using simple approximation
  - Finite differencing on a regular grid

Bump mapping details





$$s'(u,v) = s(u,v) + b(u,v)\,\hat{n}(u,v)$$

$$n(u,v) = \frac{\partial s}{\partial u} \times \frac{\partial s}{\partial v}$$

$$\hat{n}(u,v) = \frac{n(u,v)}{\|n\|}$$

$$n'(u,v) = \frac{\partial s'}{\partial u} \times \frac{\partial s'}{\partial v}$$

Bump mapping details

$$n'(u,v) = \frac{\partial s'}{\partial u} \times \frac{\partial s'}{\partial v}$$

$$n'(u,v) = n(u,v) + \underline{\qquad\qquad}$$

(unnormalized)

approximate by:

$$\frac{\partial b}{\partial u}\left(n(u,v) \times \frac{\partial s}{\partial v}\right) - \frac{\partial b}{\partial v}\left(n(u,v) \times \frac{\partial s}{\partial u}\right)$$

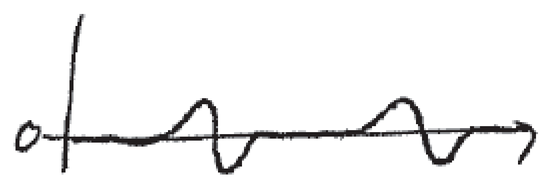
a	b	c

$\frac{c-a}{2}$

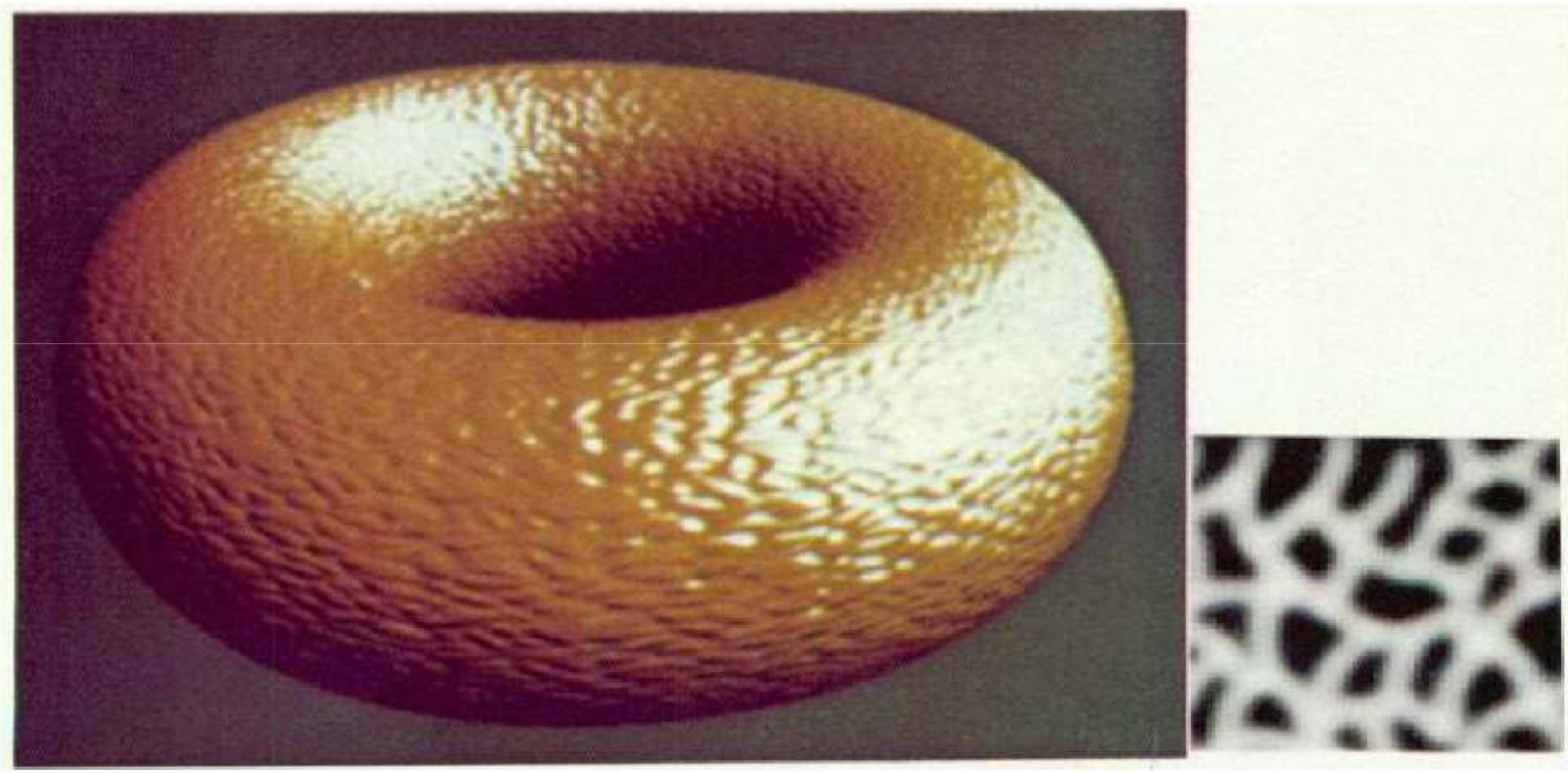
"image" derivatives (filtering)

surface tangents in u + v directions

$\frac{\partial b}{\partial u}$



Bump mapping results



[Blinn 1978]

Distributed ray tracing

$$I \approx I_L + I_{reflect} + I_{refract}$$

- Illumination from just one (of a few) directions
  - On of the approximations in backward ray tracing

$L_{in}(w_{in})$ = radiance from direction w_{in}

$R(w_{in}, w_{out})$ = light reflected in w_{out} direction incoming as w_{in}

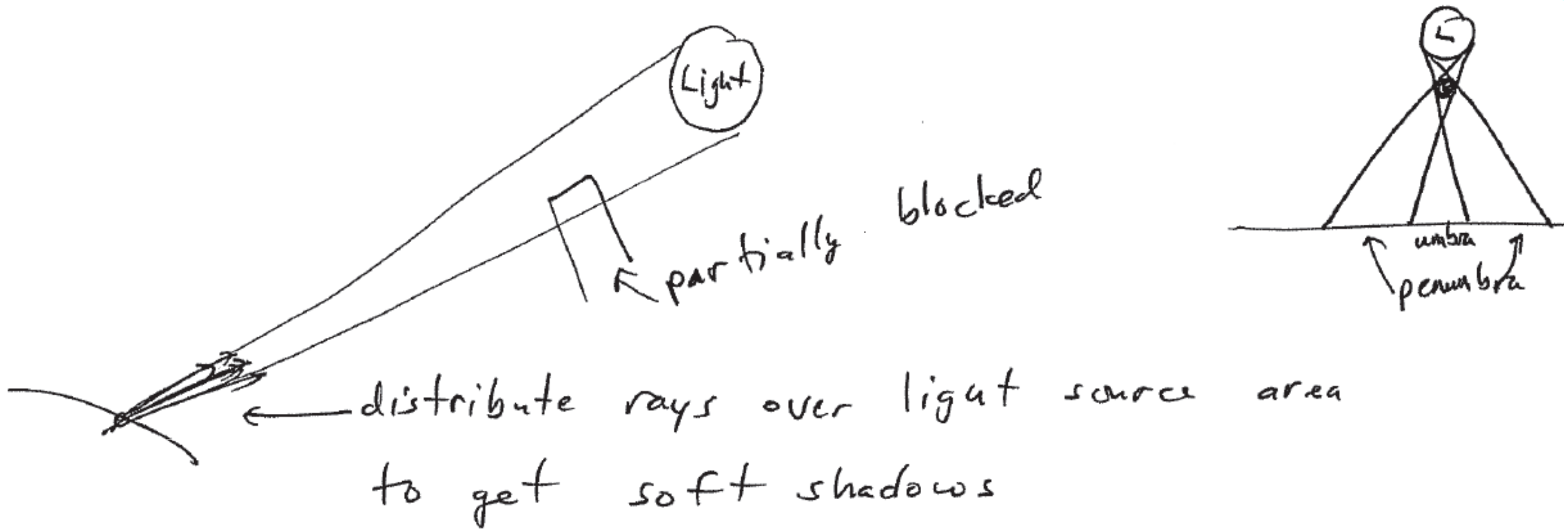
$$I(w_{out}) = \int L_{in}(w_{in})\, R(w_{in}, w_{out})\, d\, w_{in}$$

↑ intensity of reflected light in w_{out} direction

Distributed ray tracing

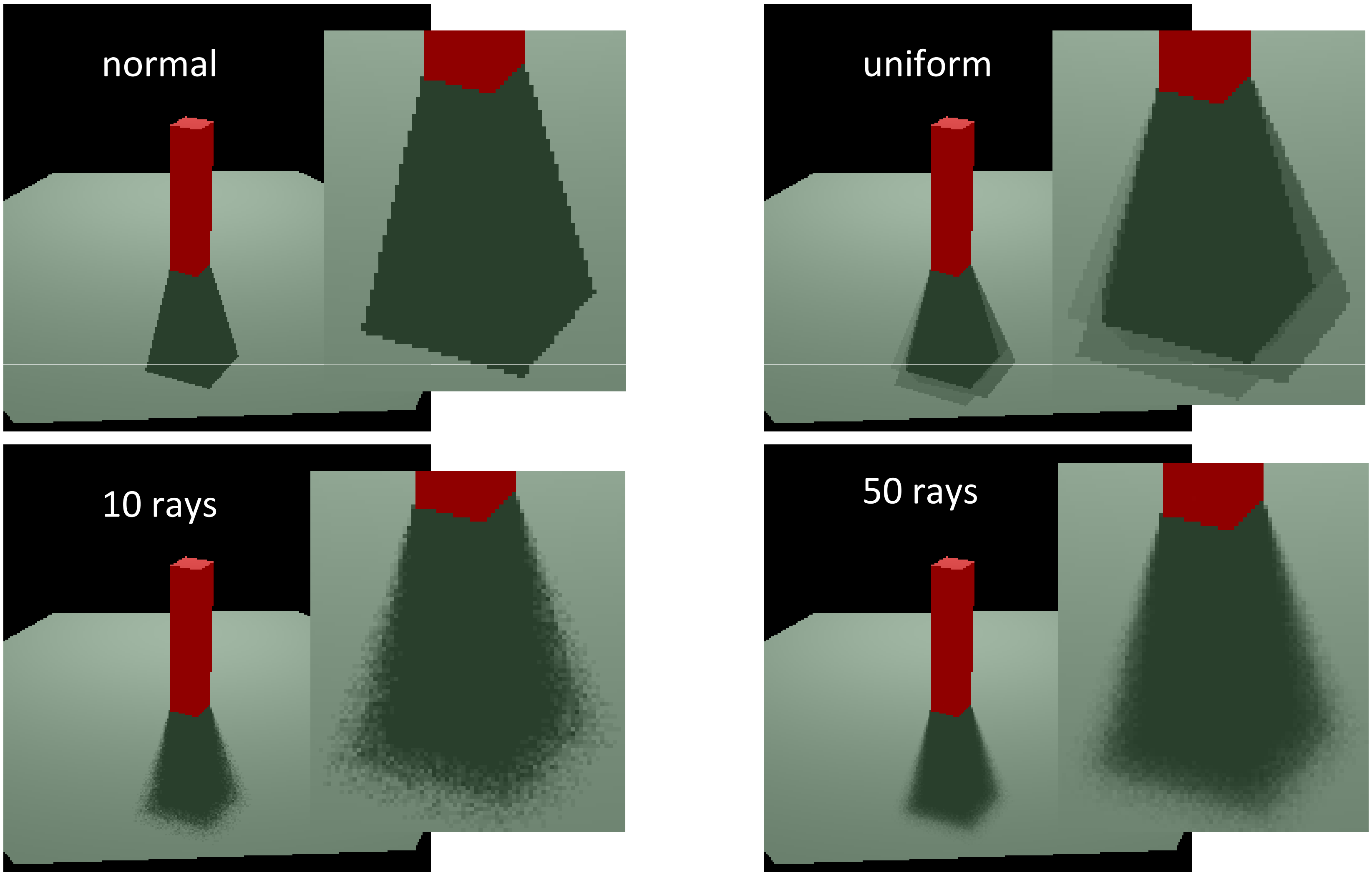
- Prior assumptions
  1. L is a δ function (light from point light source)
  2. Elsewhere, L is independent of ω_{in} (ambient light)
  3. R is a δ function (mirrored reflections)
- 1 and 3 now change
  - Relax 1 → get soft shadows
  - Relax 3 → get fuzzy/glossy reflections

Penumbras / soft shadows

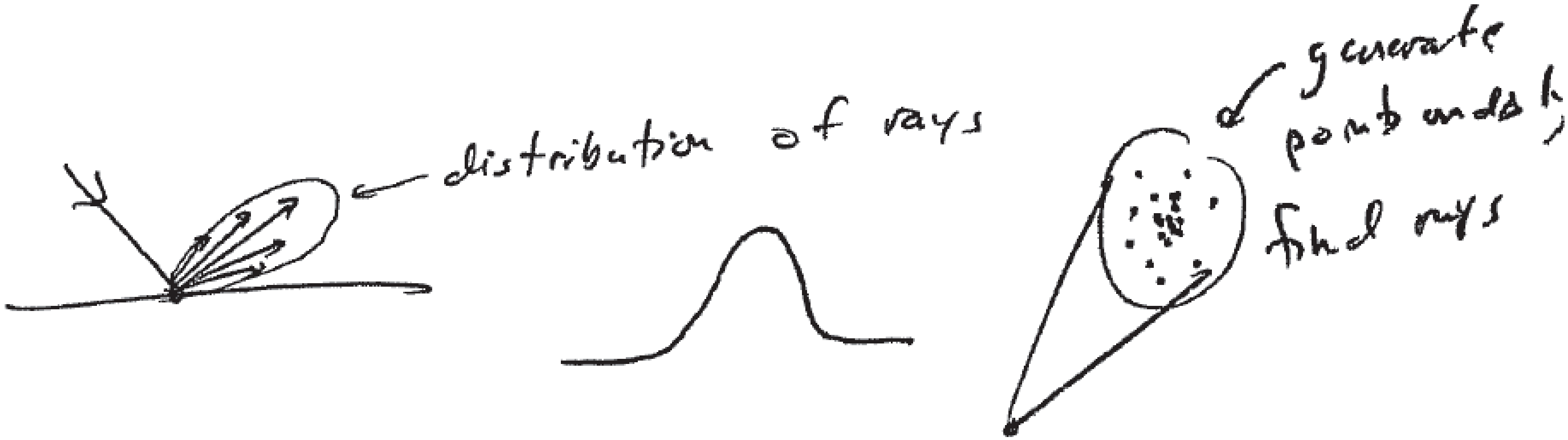


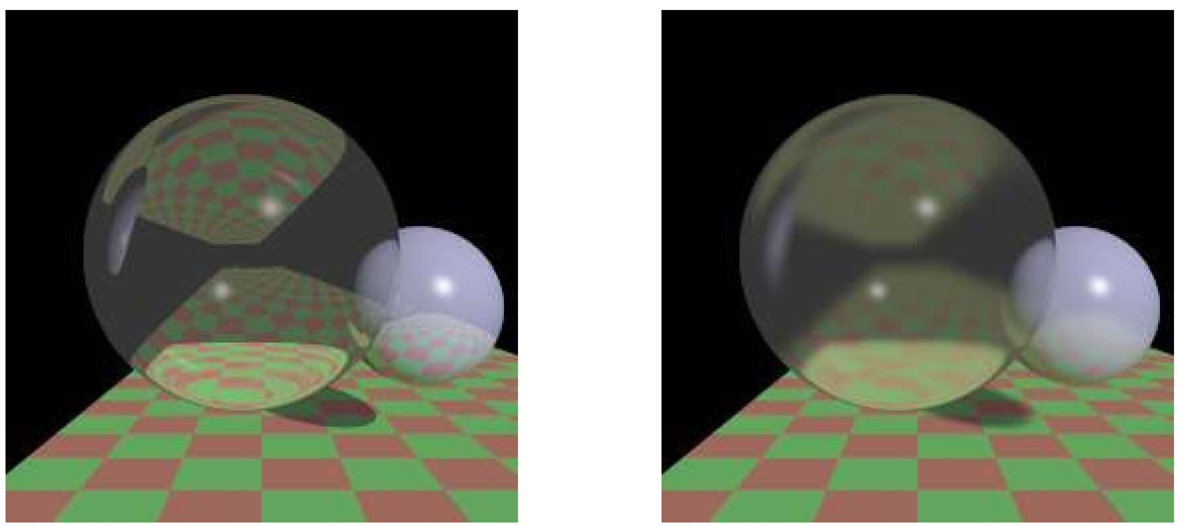
- Shoot a distribution of rays toward light + add together (and normalize)
  - Sampling the area light on a regular grid results in visible artifacts
  - Stochastic sampling (importance, Poisson) better

Penumbras / soft shadows

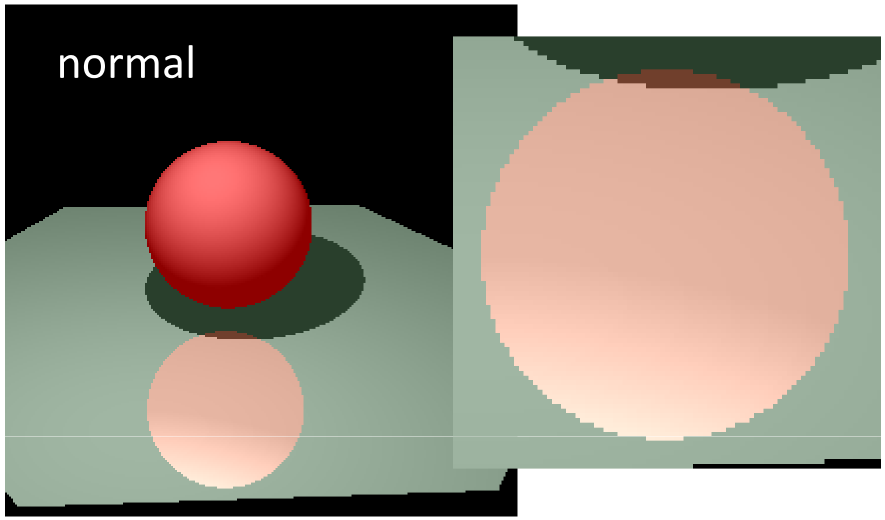


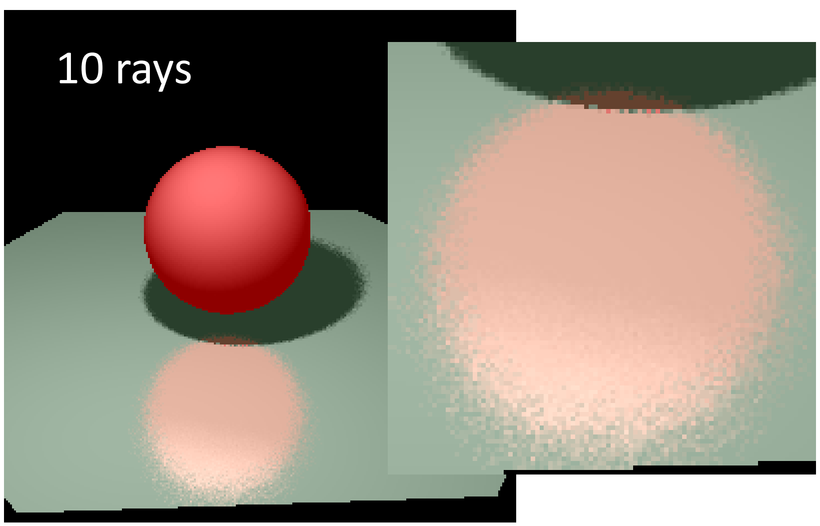
Glossy reflections

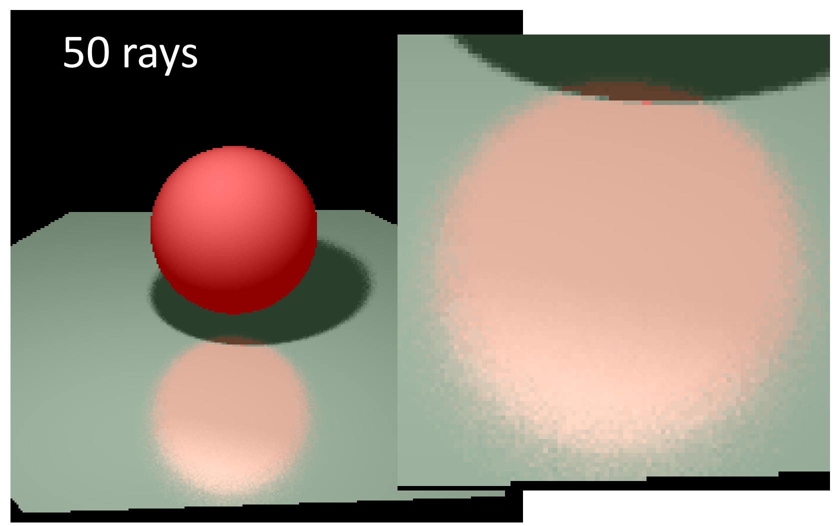




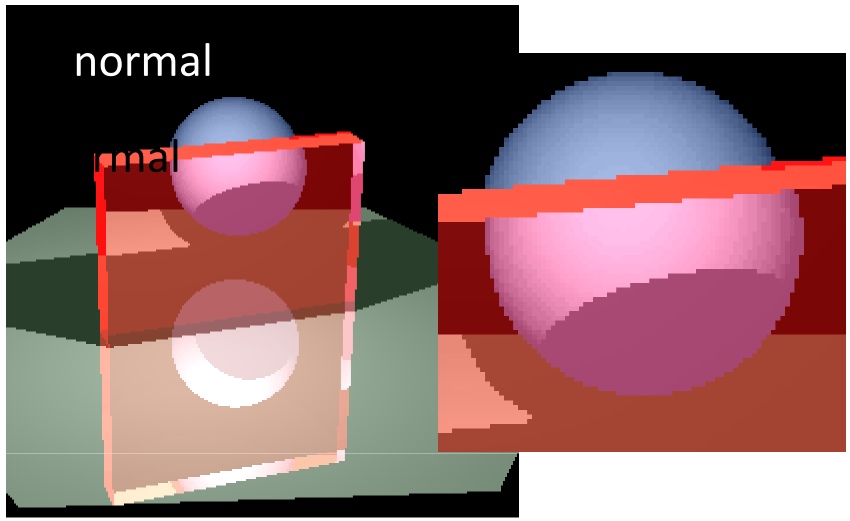
Glossy reflections

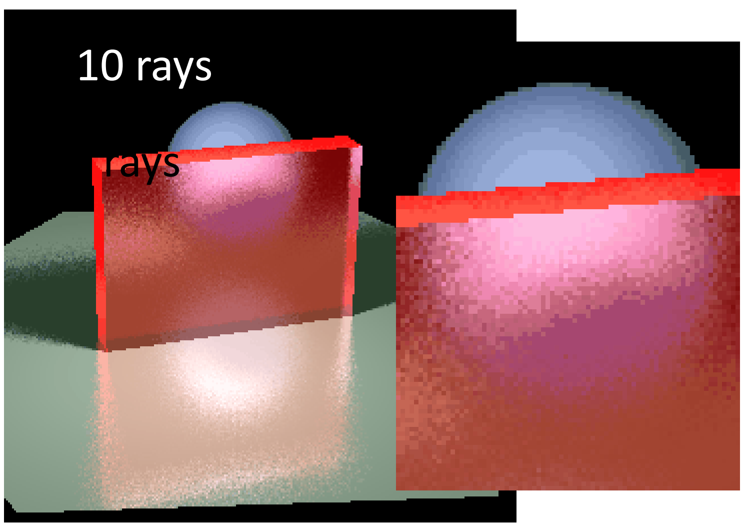


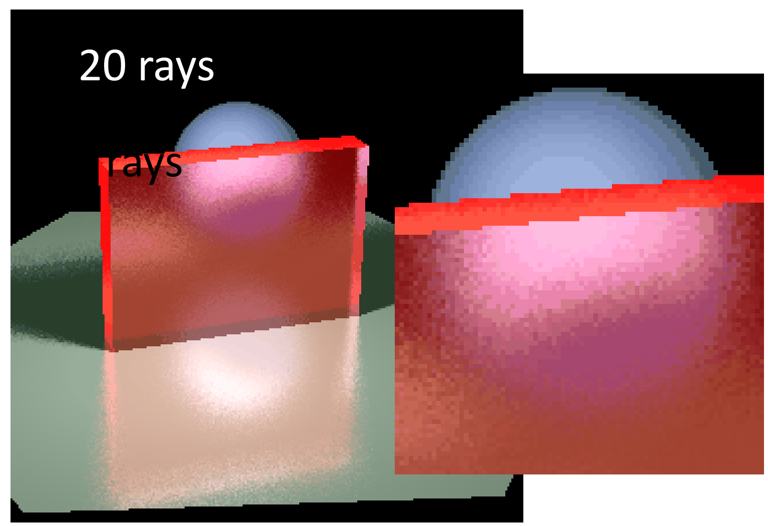




Glossy refractions

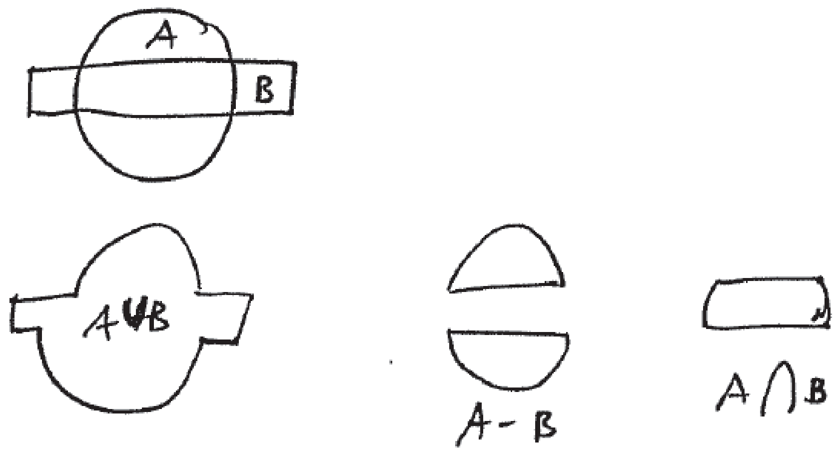






Constructive solid geometry (CSG)

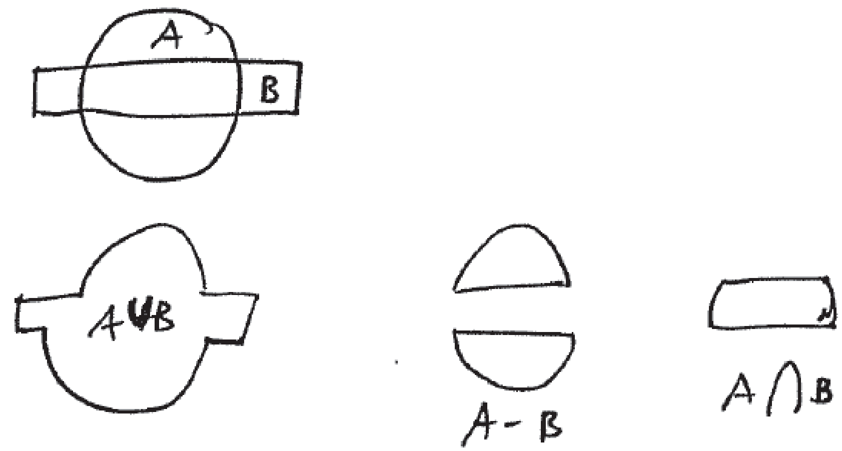
- Boolean operations on solids
  - Union, intersection and difference

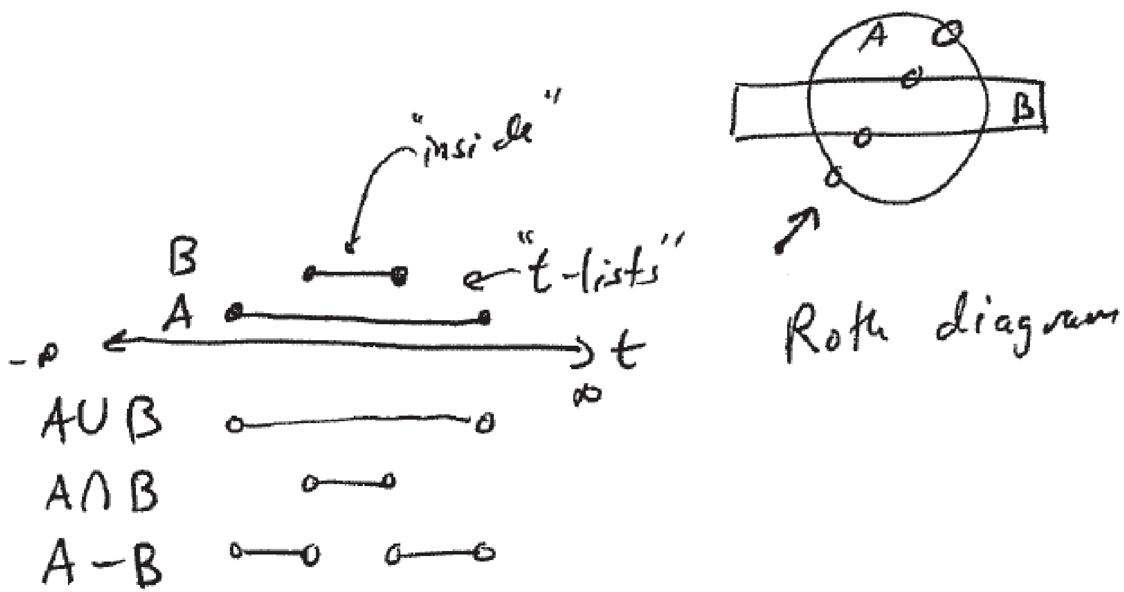


- Adding CSG to a raytracer is not difficult
- Just need to be able to
  - Find all intersection points with a ray + object, not just closest, and know whether ray is **entering** or **exiting**
  - If "skimming" then skip it

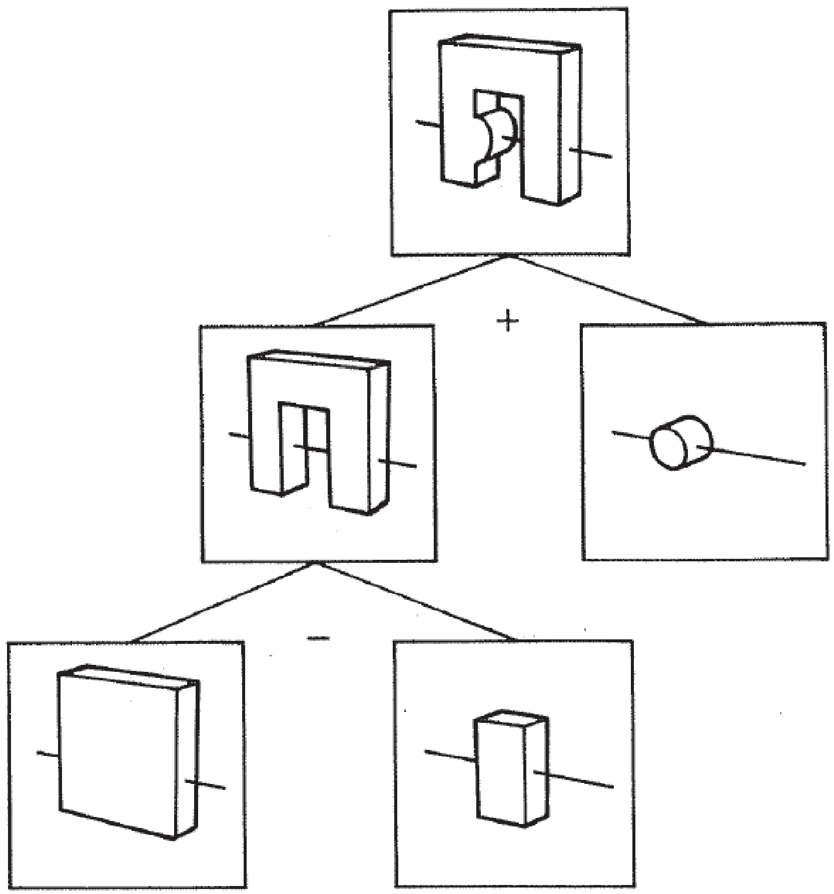
Constructive solid geometry (CSG)

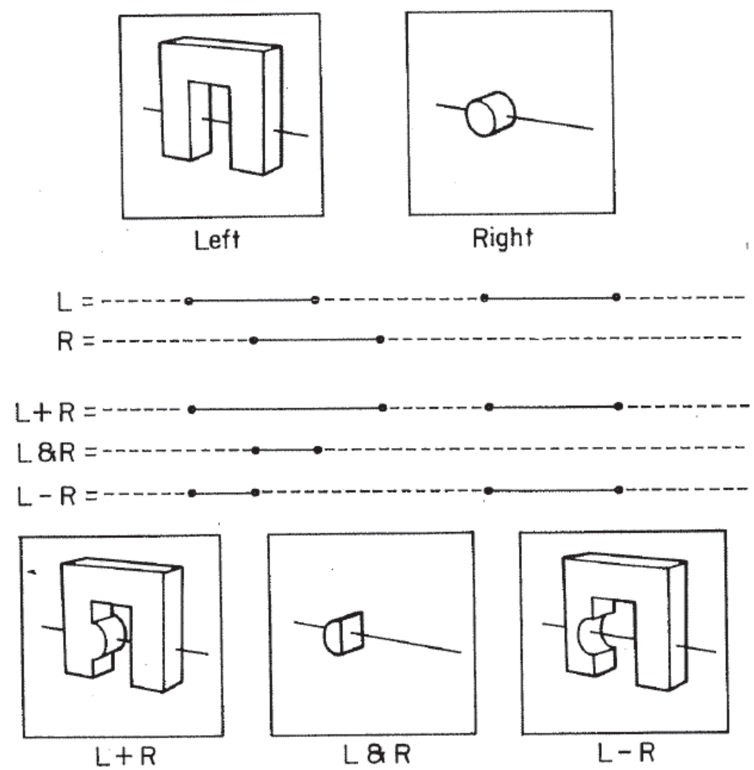
- Boolean operations on solids
  - Union, intersection and difference





Constructive solid geometry (CSG)





Constructive solid geometry (CSG)