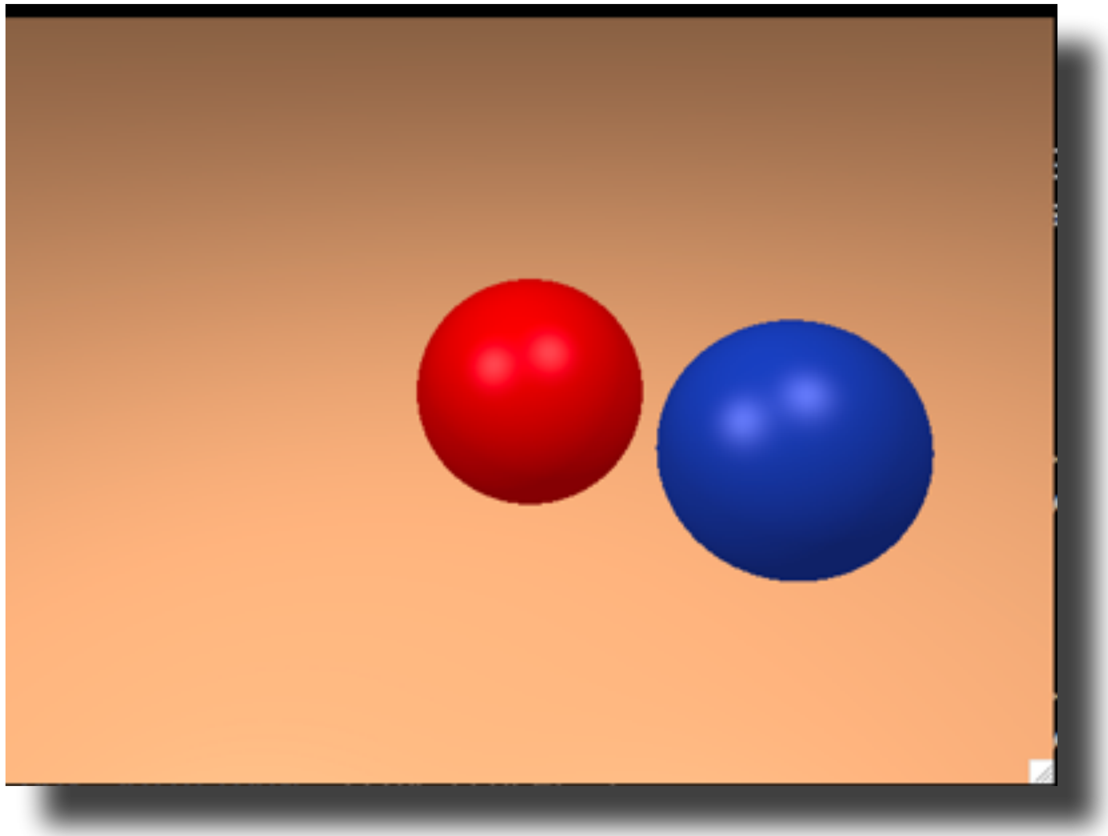
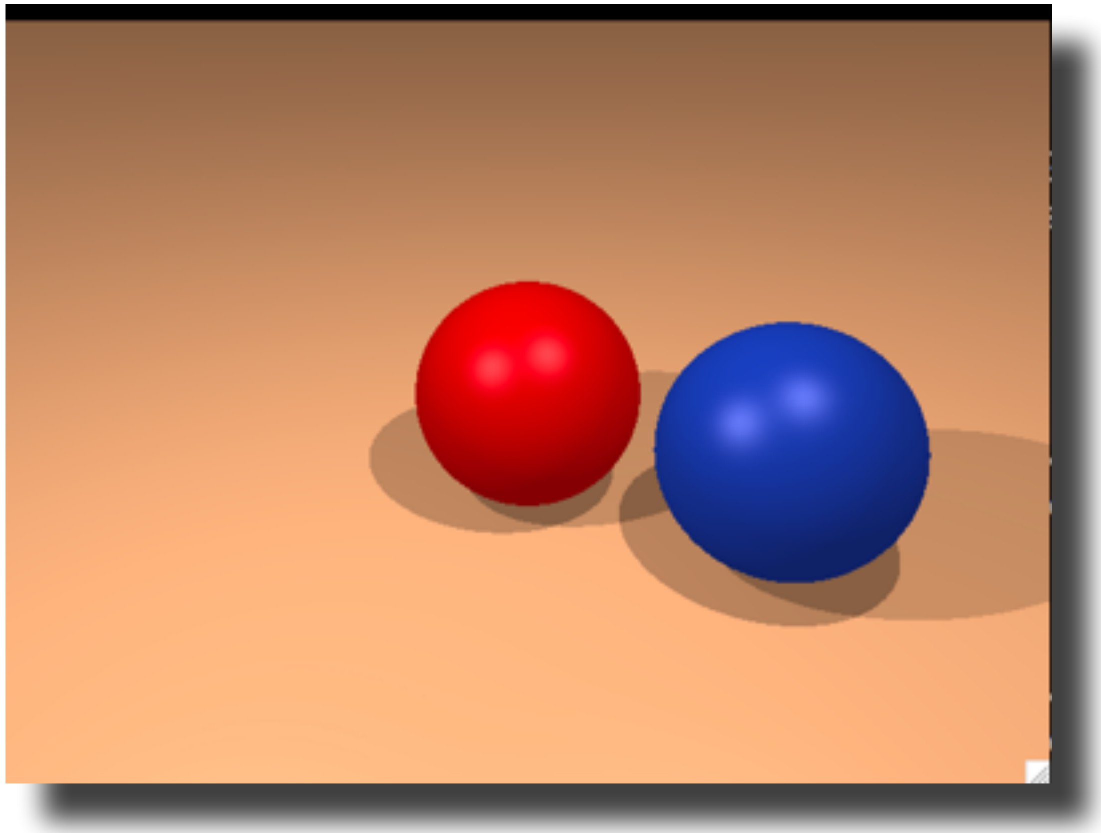
Shadows





Shadows

```
for each pixel do
    compute viewing ray
    if ( ray hits an object with t in [0, inf] ) then
        compute n
        evaluate shading model and set pixel to that color
    else
        set pixel color to the background color
```

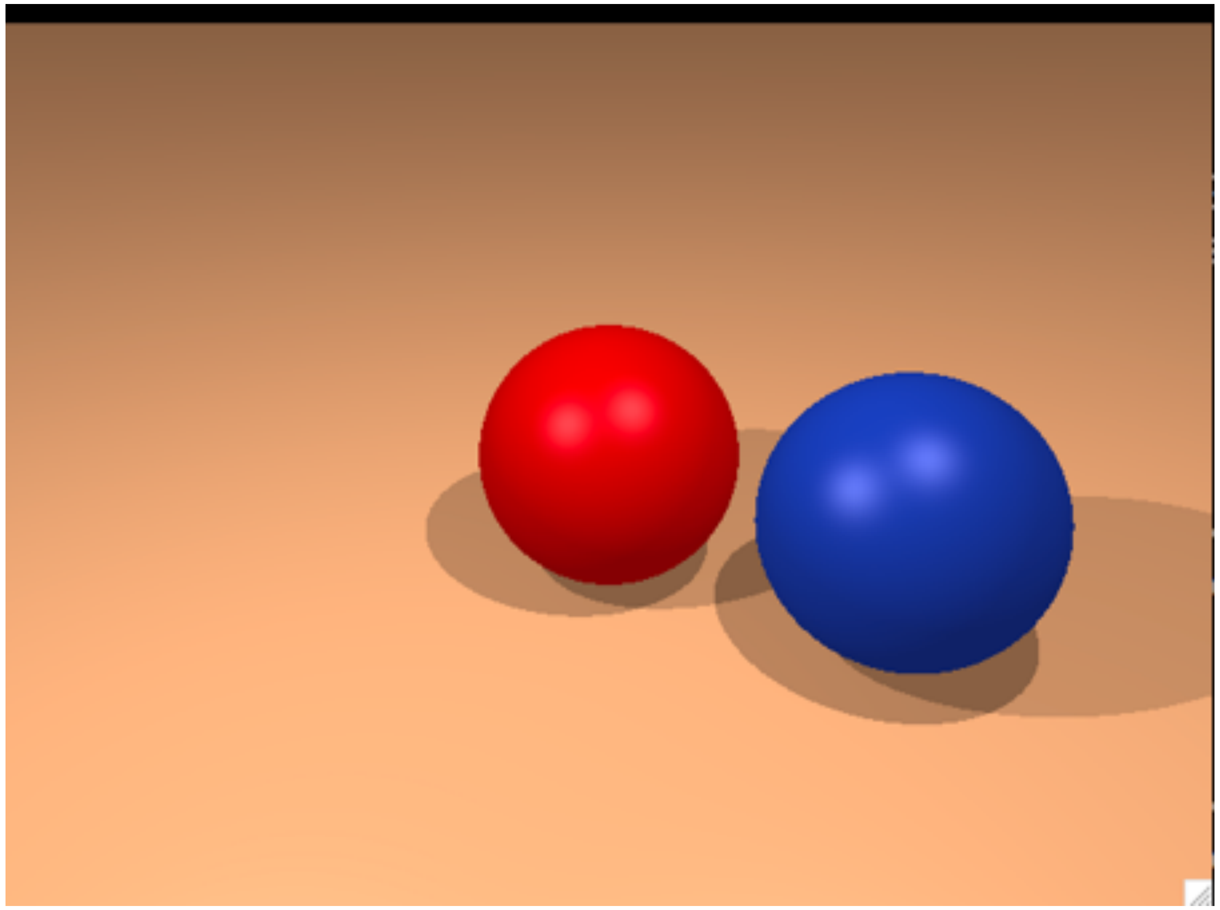
Shadows

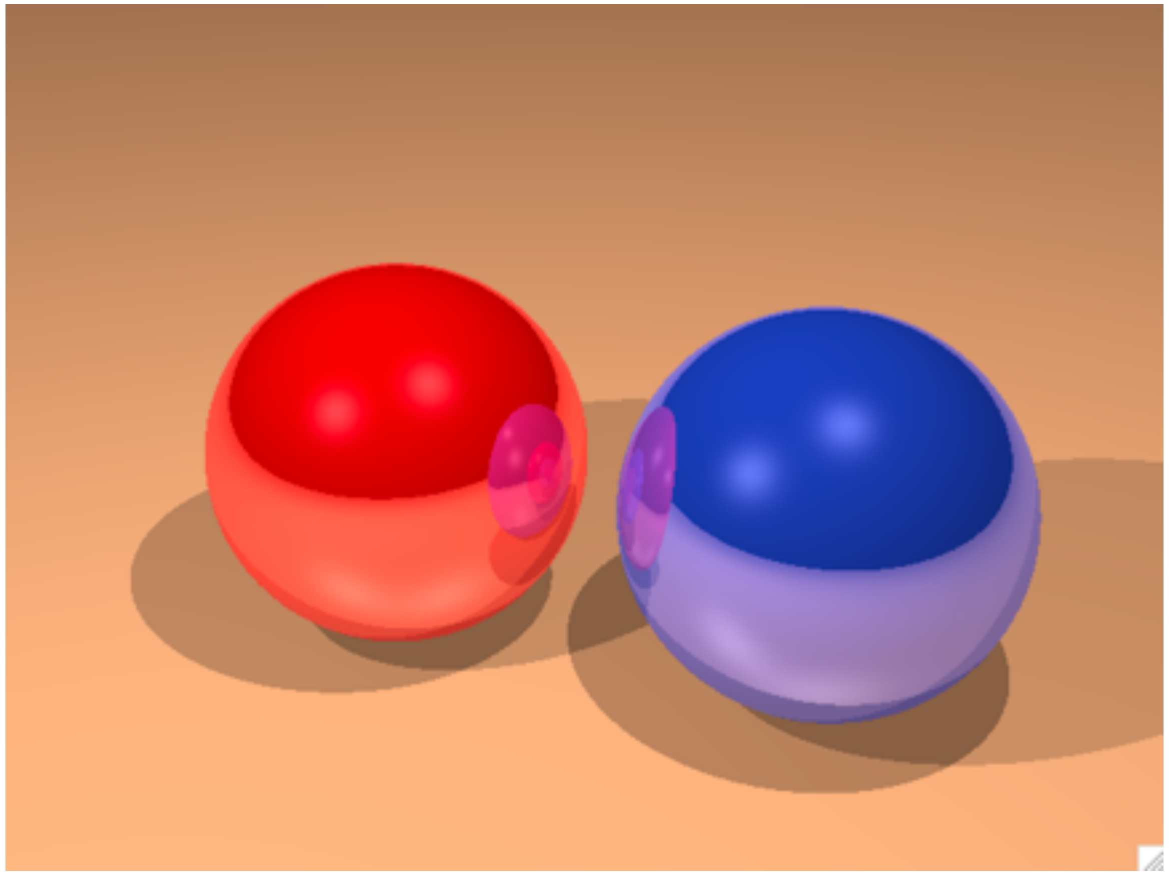
```
for each pixel do
    compute viewing ray
    if ( ray hits an object with t in [0, inf] ) then
        compute n
        evaluate shading model and set pixel to that color
    else
        set pixel color to the background color
```

Shadows

```
for each pixel do
    compute viewing ray
    if ( ray hits an object with t in [0, inf] ) then
        compute n
        // e.g., phong shading
        for each light
          add light's ambient component
          compute shadow ray
          if ( ! shadow ray hits an object )
            add light's diffuse and specular components
    else
        set pixel color to the background color
```

Reflections





Reflections

```
for each pixel do
    compute viewing ray
    if ( ray hits an object with t in [0, inf] ) then
        compute n
        evaluate shading model and set pixel to that color
    else
        set pixel color to the background color
```

Reflections

```
for each pixel do
    compute viewing ray
    if ( ray hits an object with t in [0, inf] ) then
        compute n
        evaluate shading model and set pixel to that color
    else
        set pixel color to the background color
```

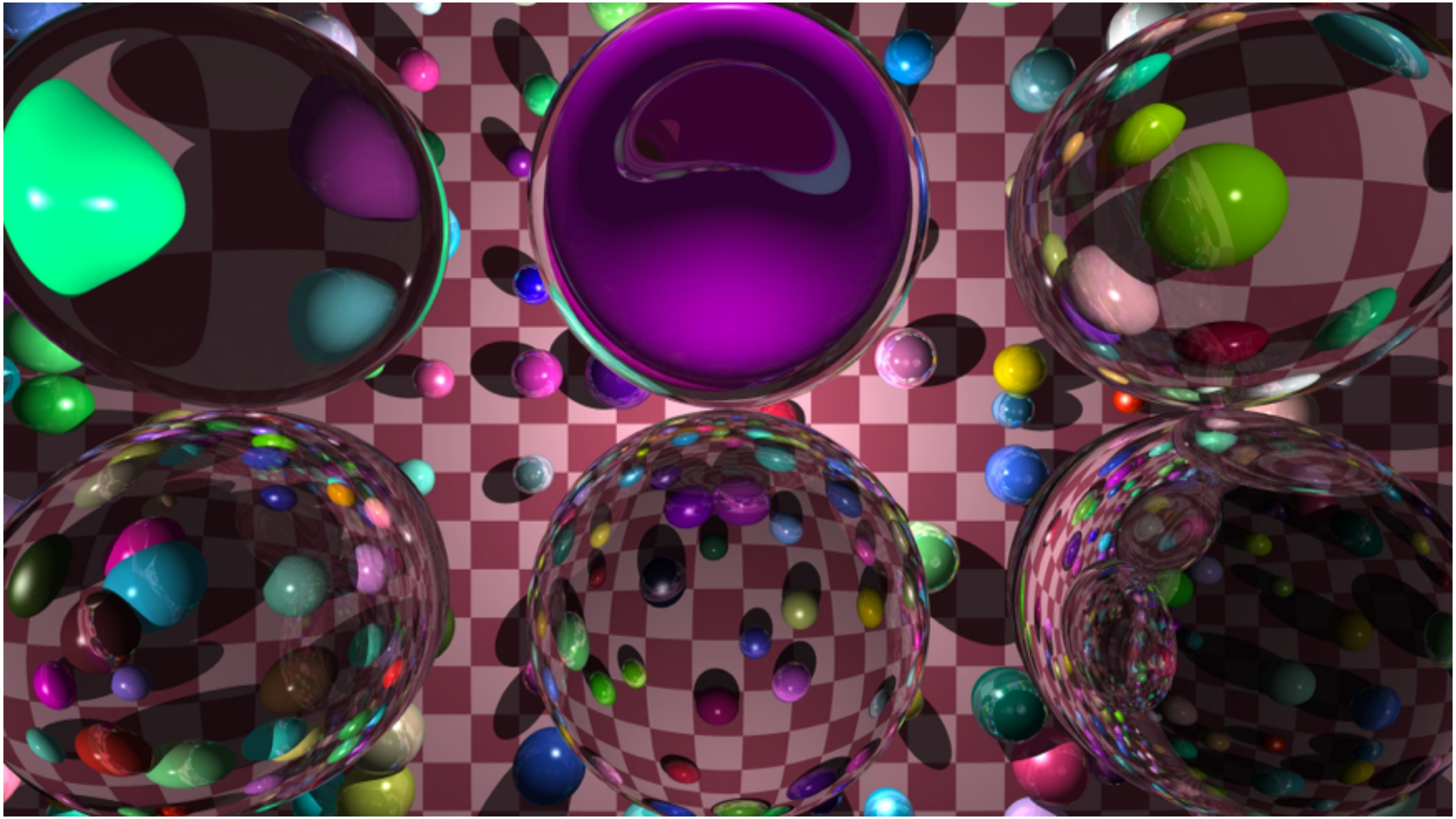
Reflections

```
for each pixel do
    compute viewing ray
    pixel color = cast_ray(viewing ray)

cast_ray:
    if ( ray hits an object with t in [0, inf] ) then
        compute n
        return color = shade_surface
    else
        return color = to the background color

shade_surface:
    color = ...
    compute reflected ray
    return color = color + k * cast_ray(reflected ray)
```

ray tracer extensions

- refraction
- more complex geometry
  - instancing
  - CSG
- distribution ray tracing (Cook et al., 1984)
  - antialiasing
  - soft shadows
  - depth of field
  - fuzzy reflections
  - motion blur

Transparency and Refraction

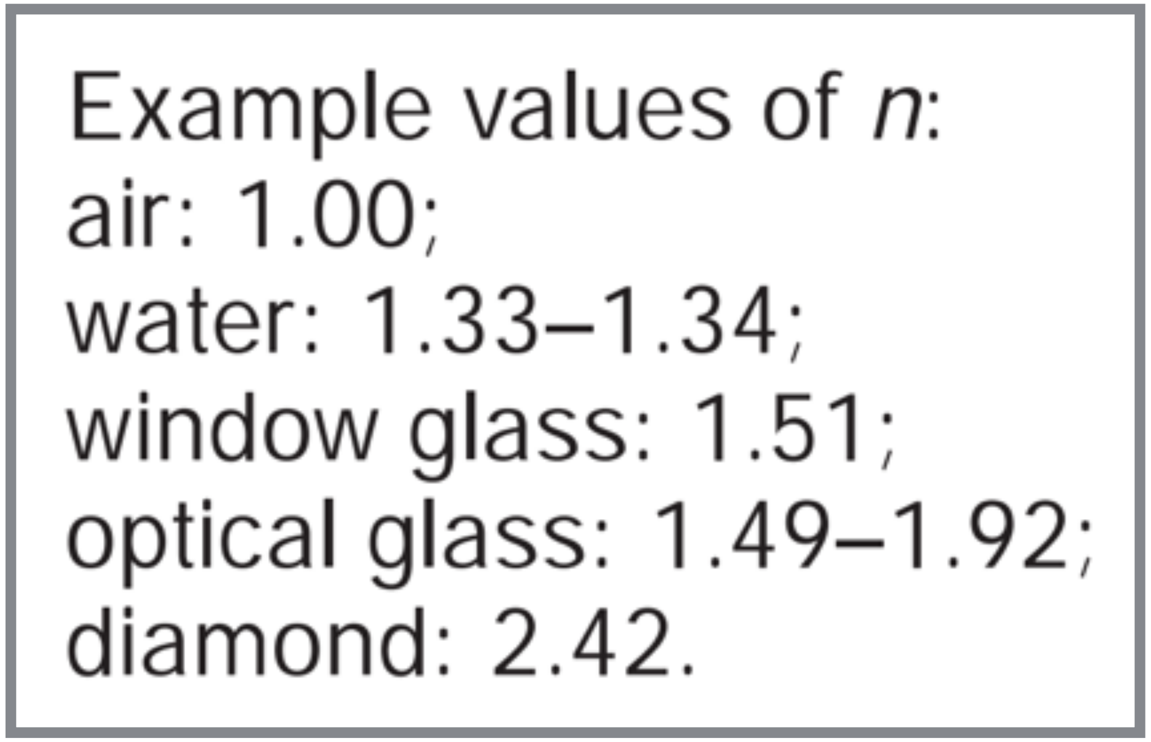


[marczych/github]

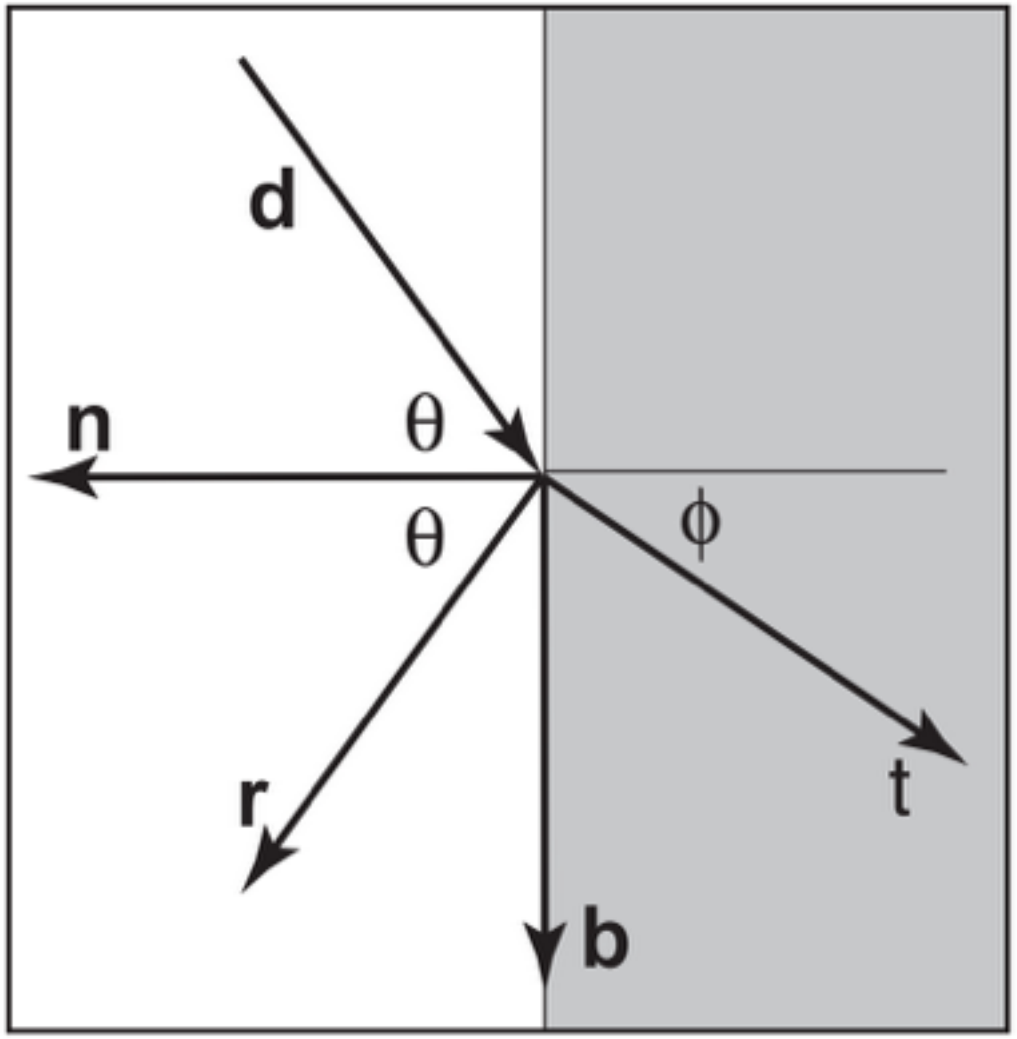
Transparency and Refraction

Snell's Law

$$n1 \sin\theta = n2 \sin\phi$$



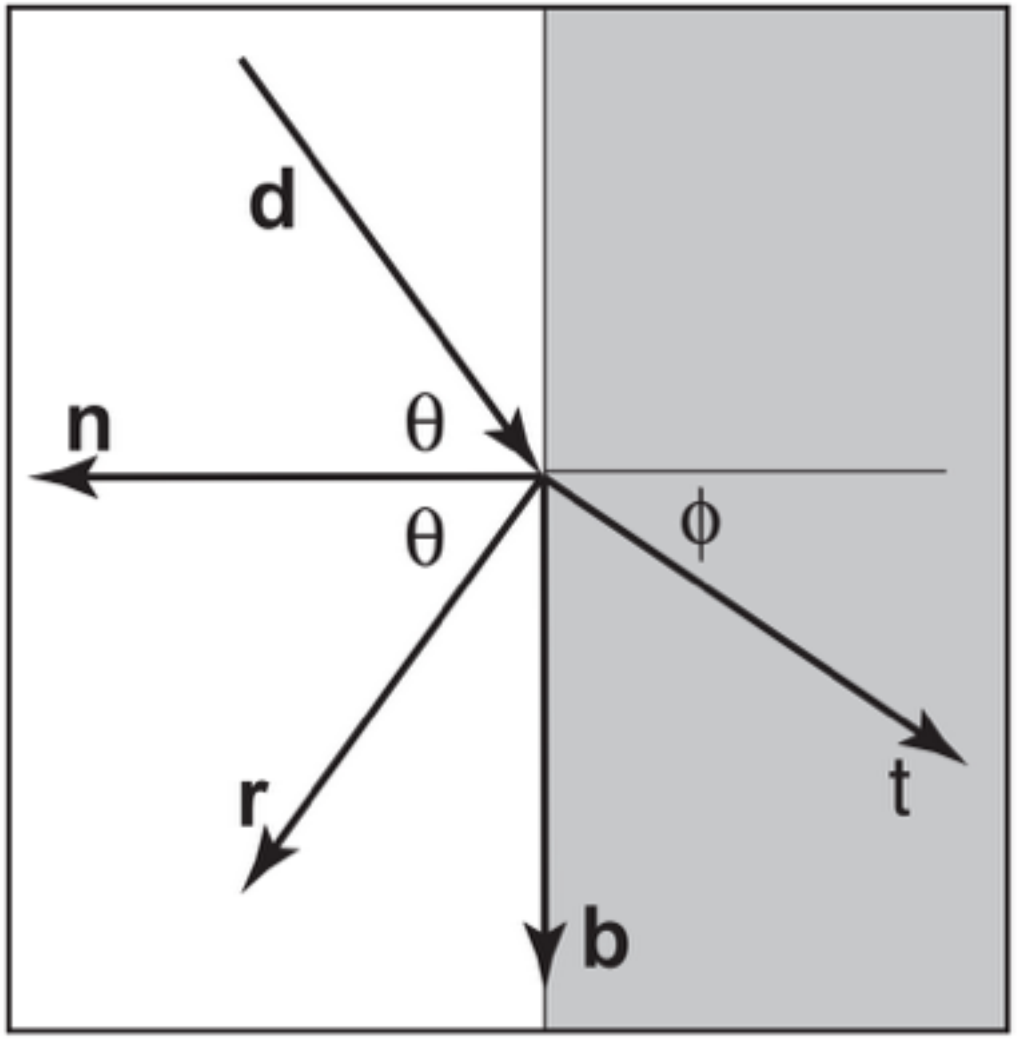
Example values of *n*:
air: 1.00;
water: 1.33–1.34;
window glass: 1.51;
optical glass: 1.49–1.92;
diamond: 2.42.



<whiteboard>

Transparency and Refraction

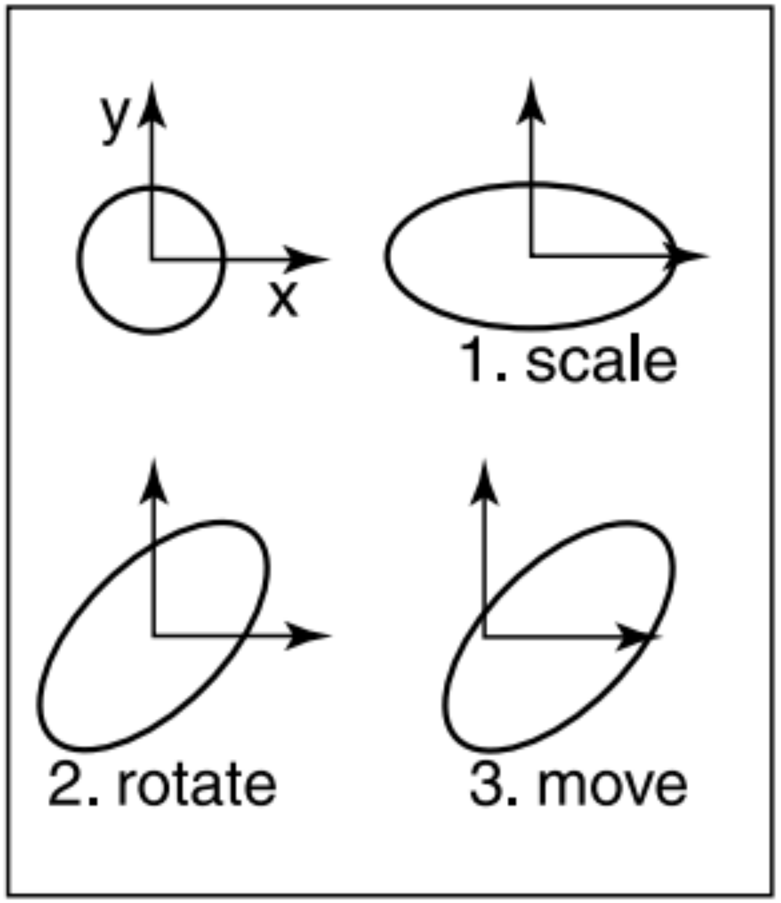
Snell's Law



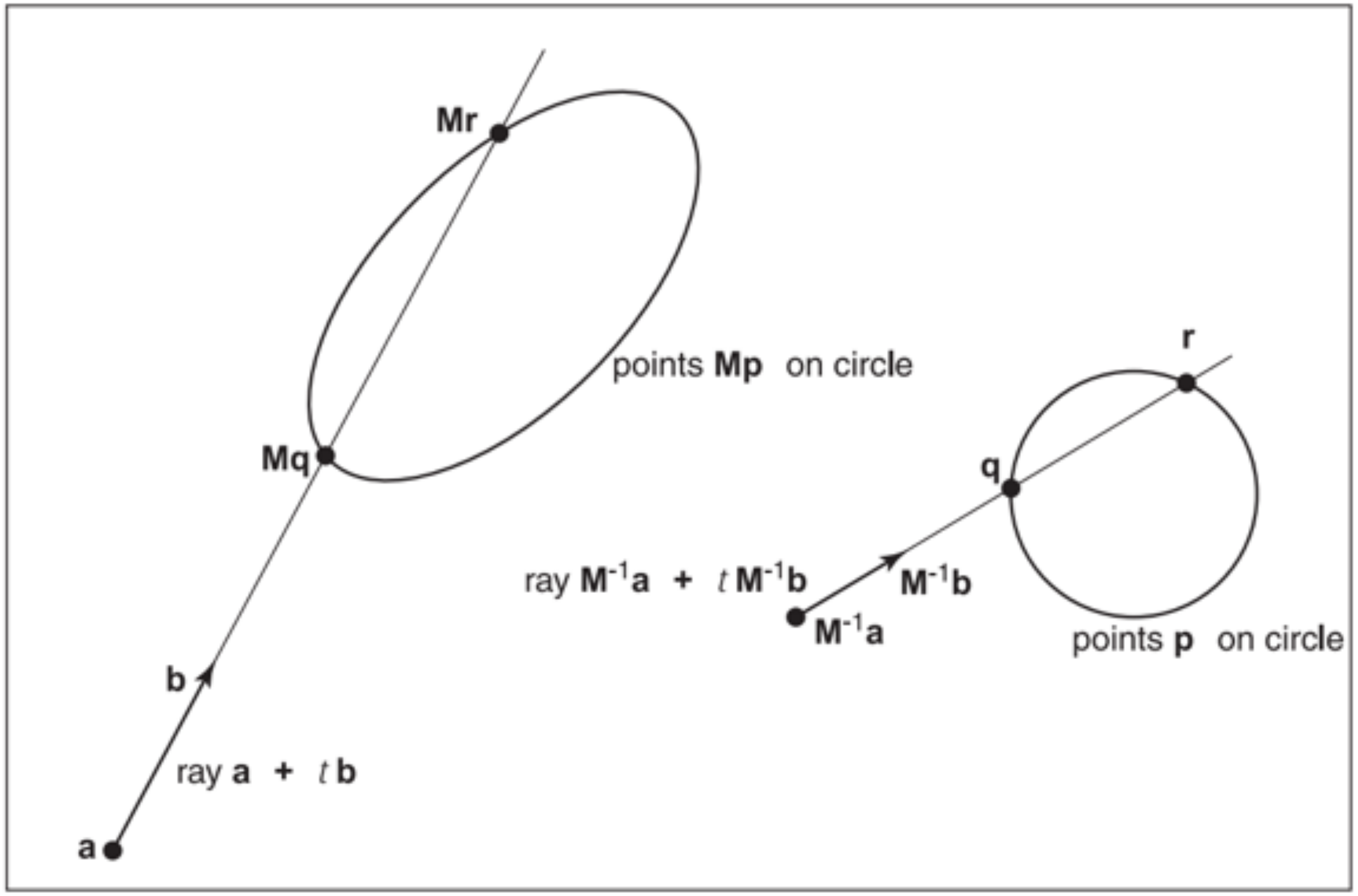
Additional effects

- varying reflectivity
  Fresnel equations
- attenuation of light intensity
  Beer's Law

Object Instancing

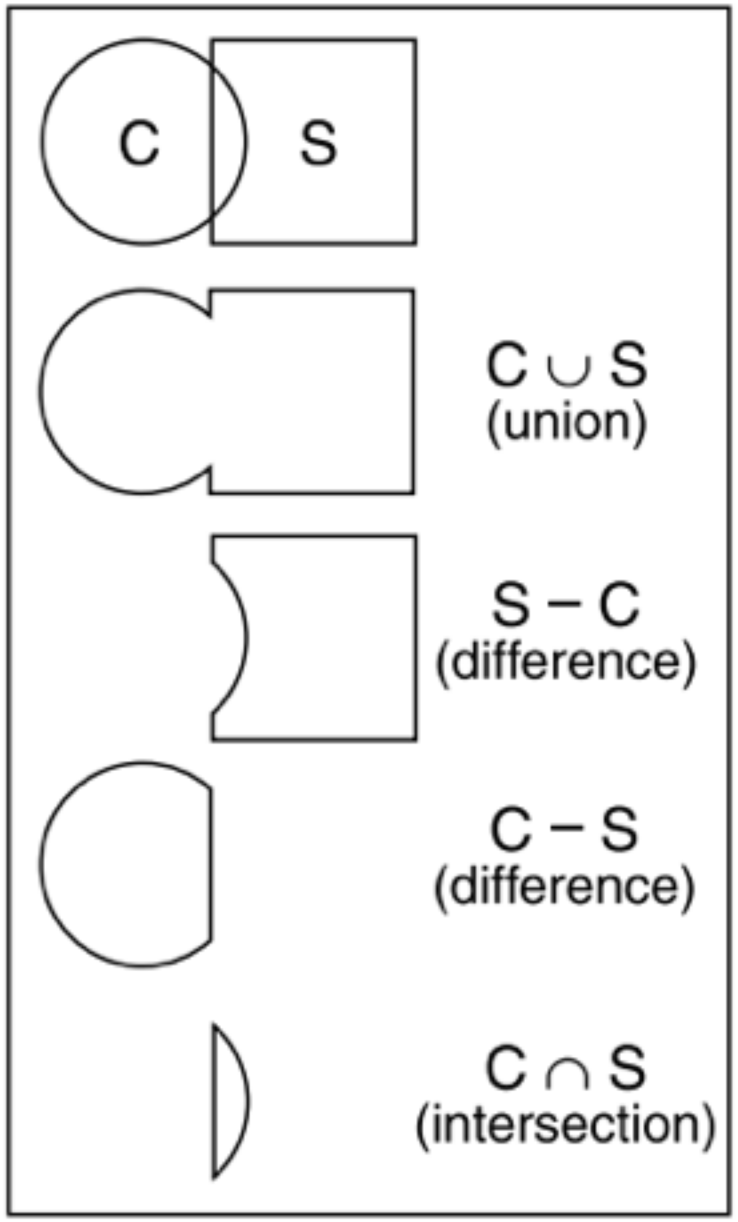


instance of circle with 3 transformations applied

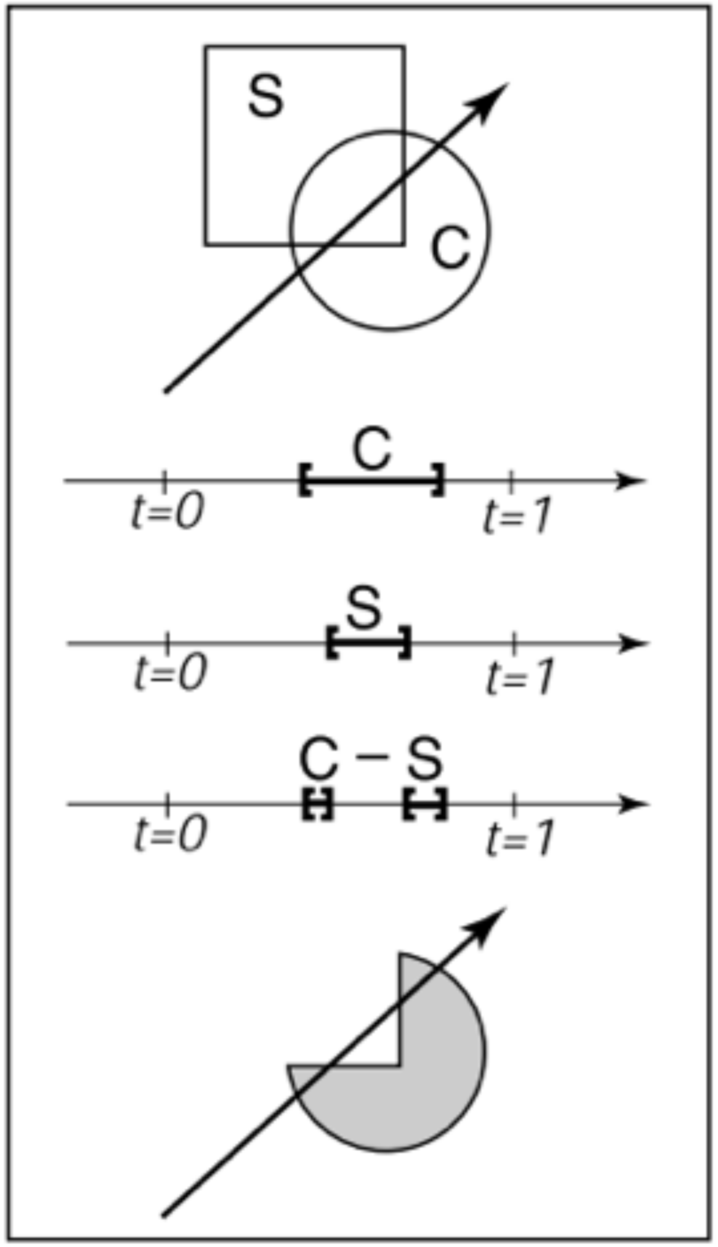


ray intersection problem in the two spaces are simple transforms of each other

Constructive Solid Geometry (CSG)



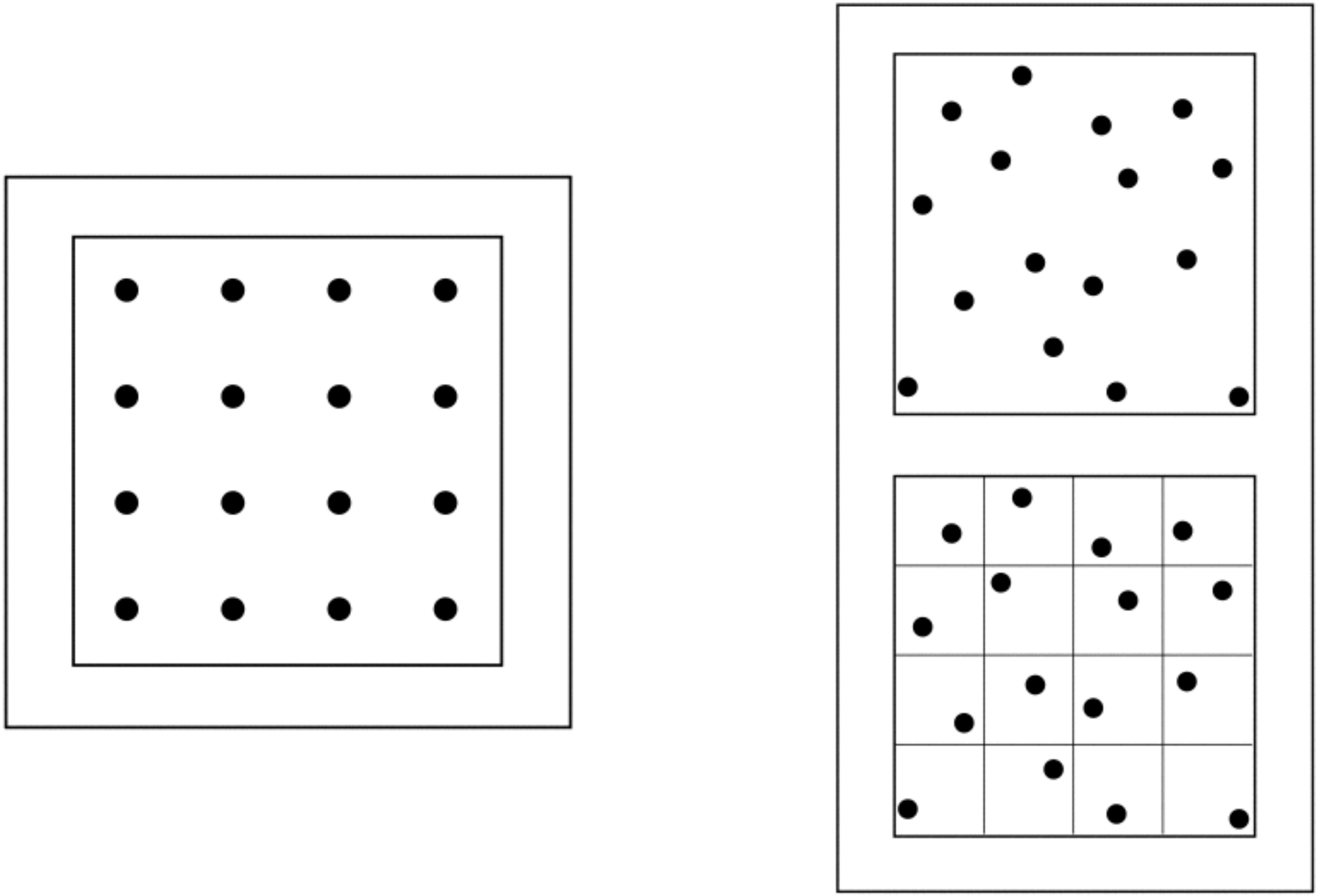
use set operations to combine solid shapes



intersection with composite object

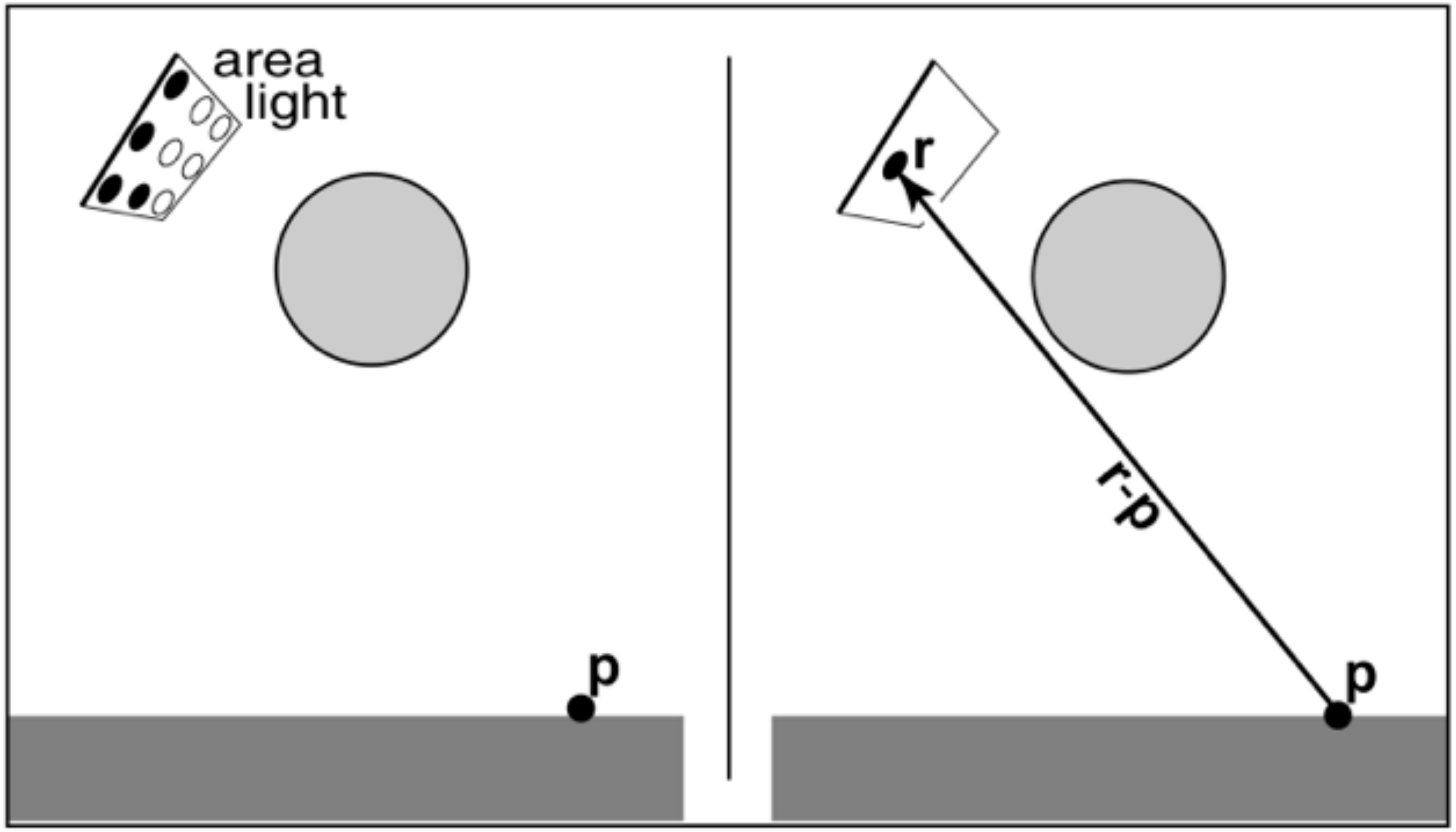
Distribution Ray Tracing

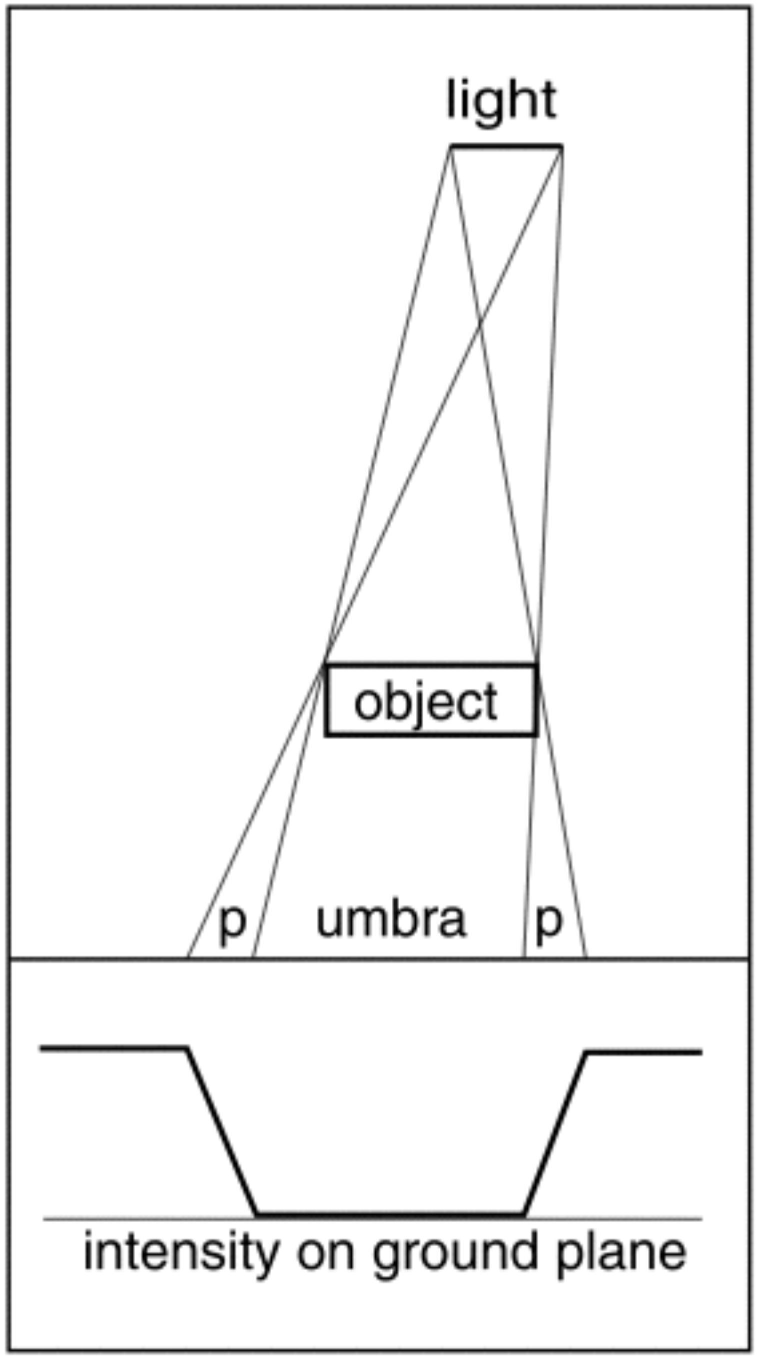
Anti-aliasing



[Shirley and Marschner]

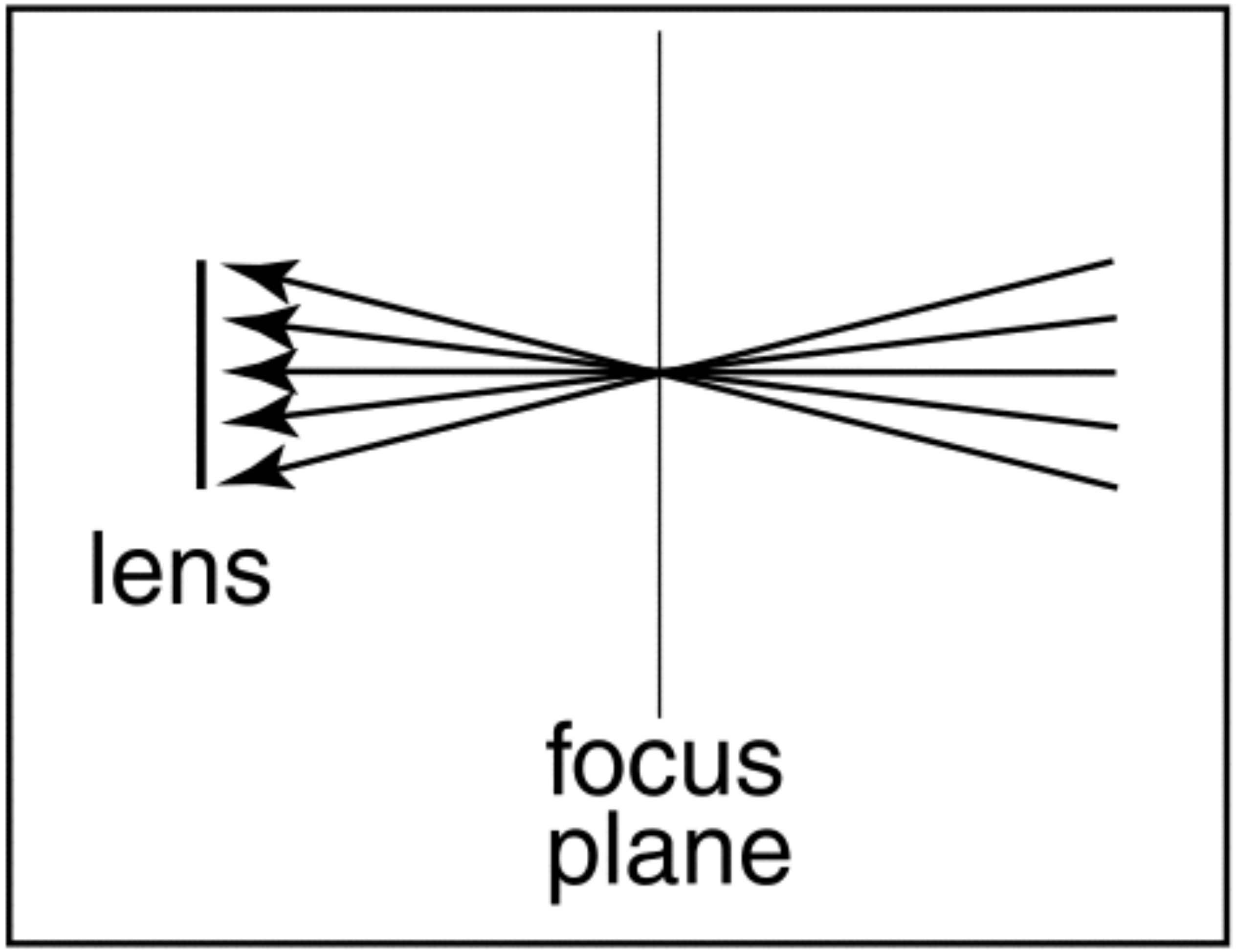
Soft Shadows

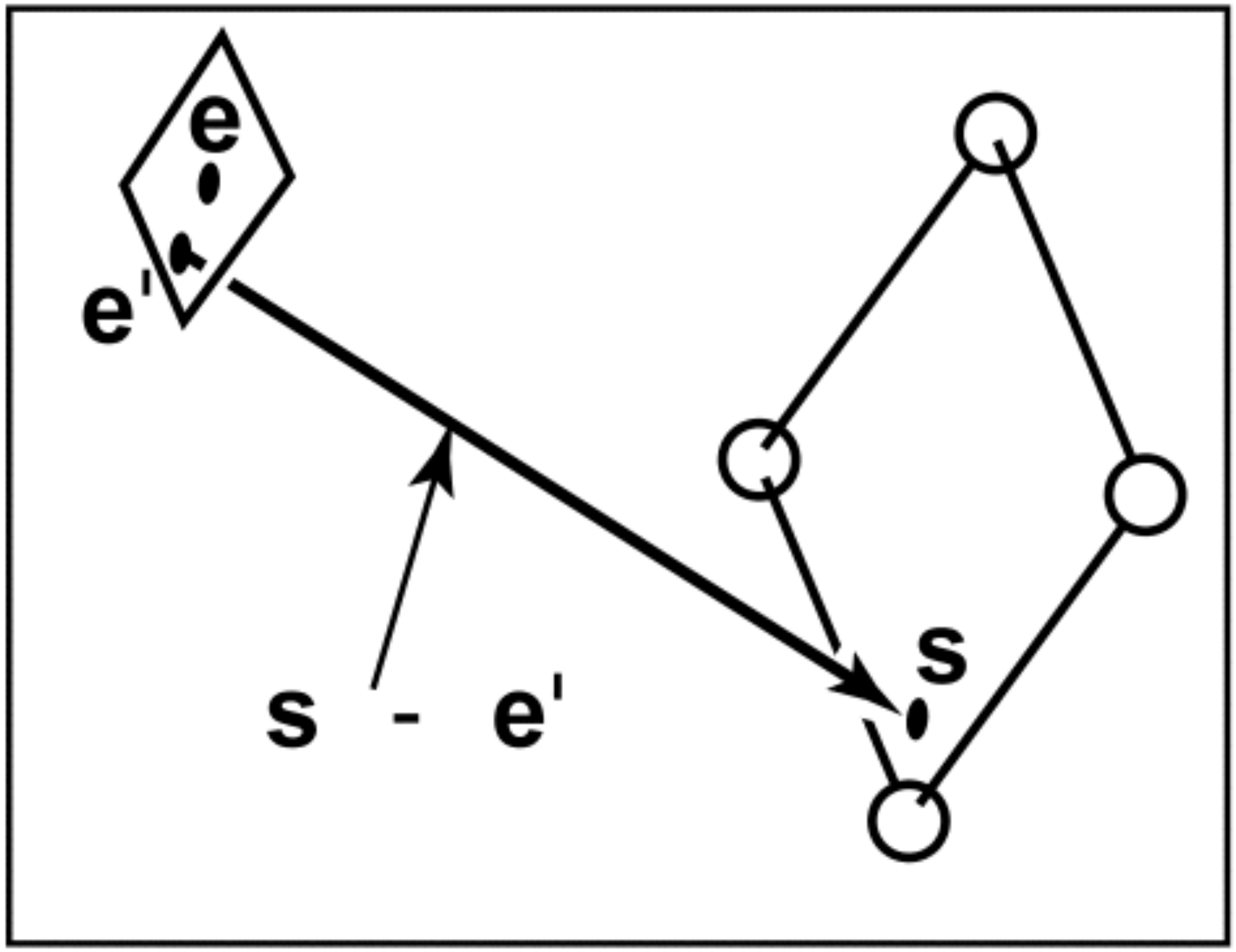




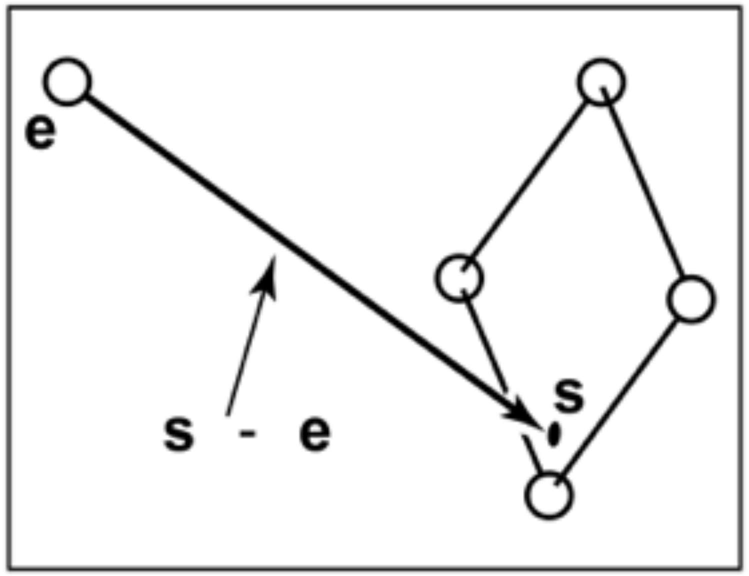
[Shirley and Marschner]

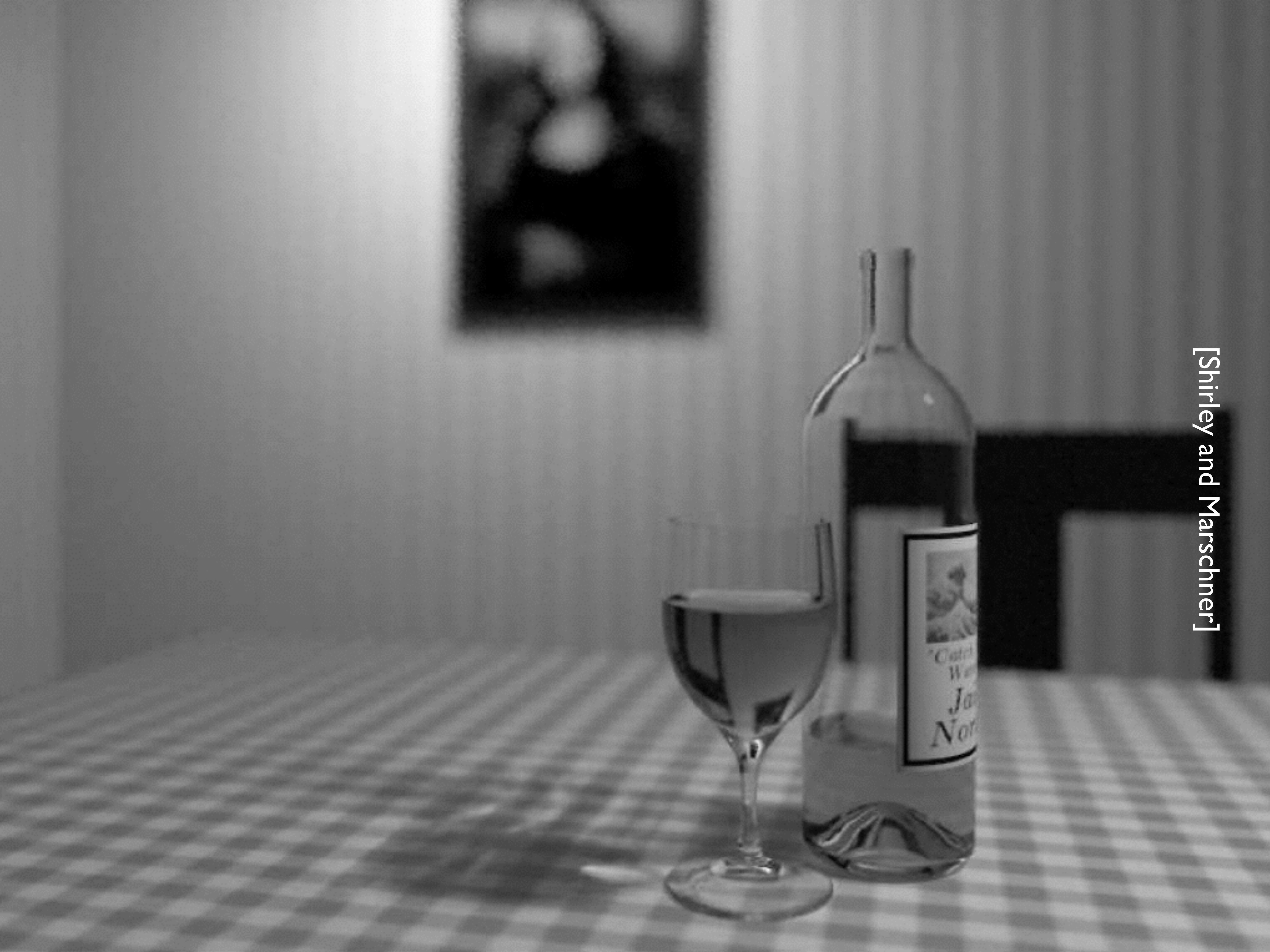
Soft Focus





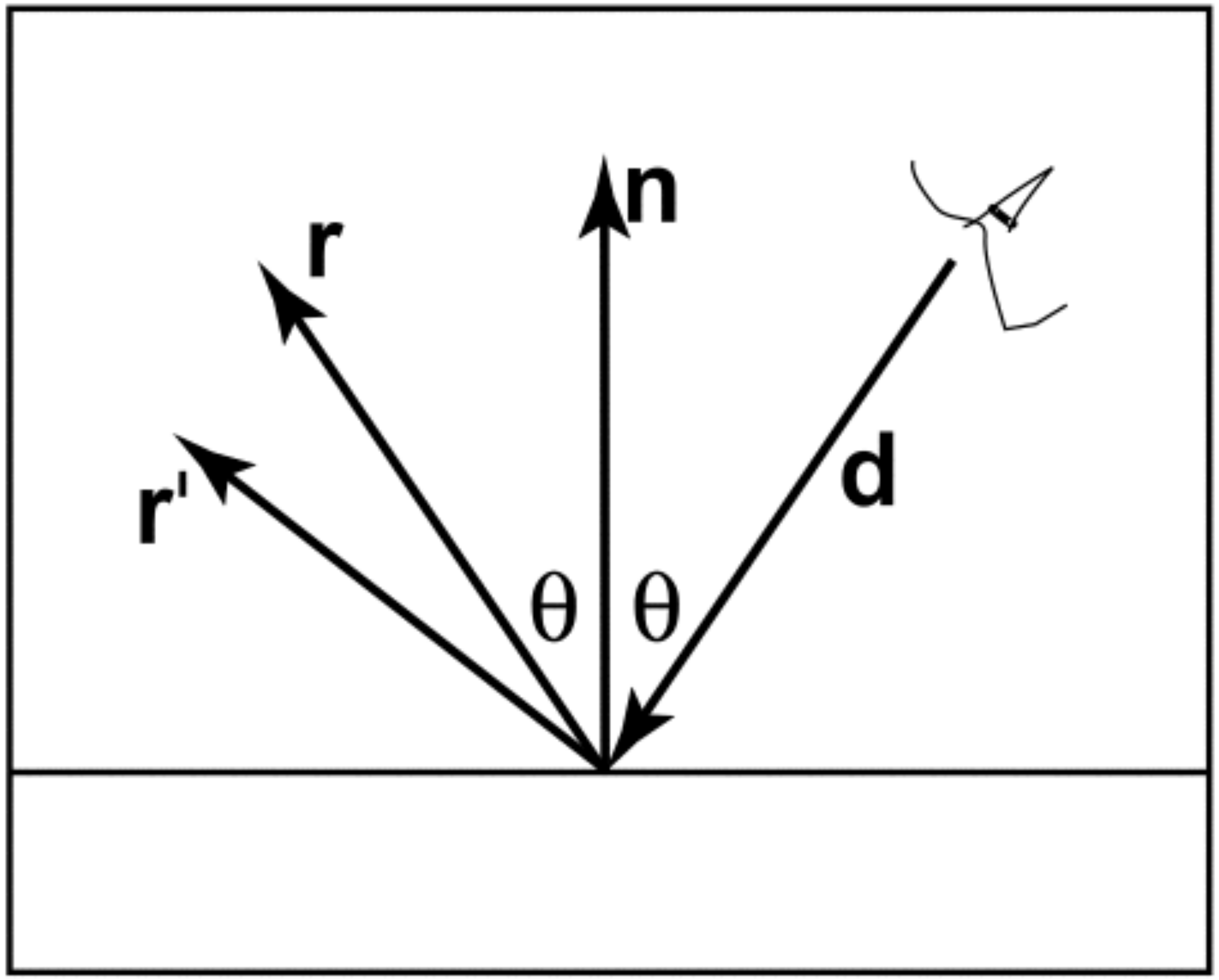
[Shirley and Marschner]





[Shirley and Marschner]

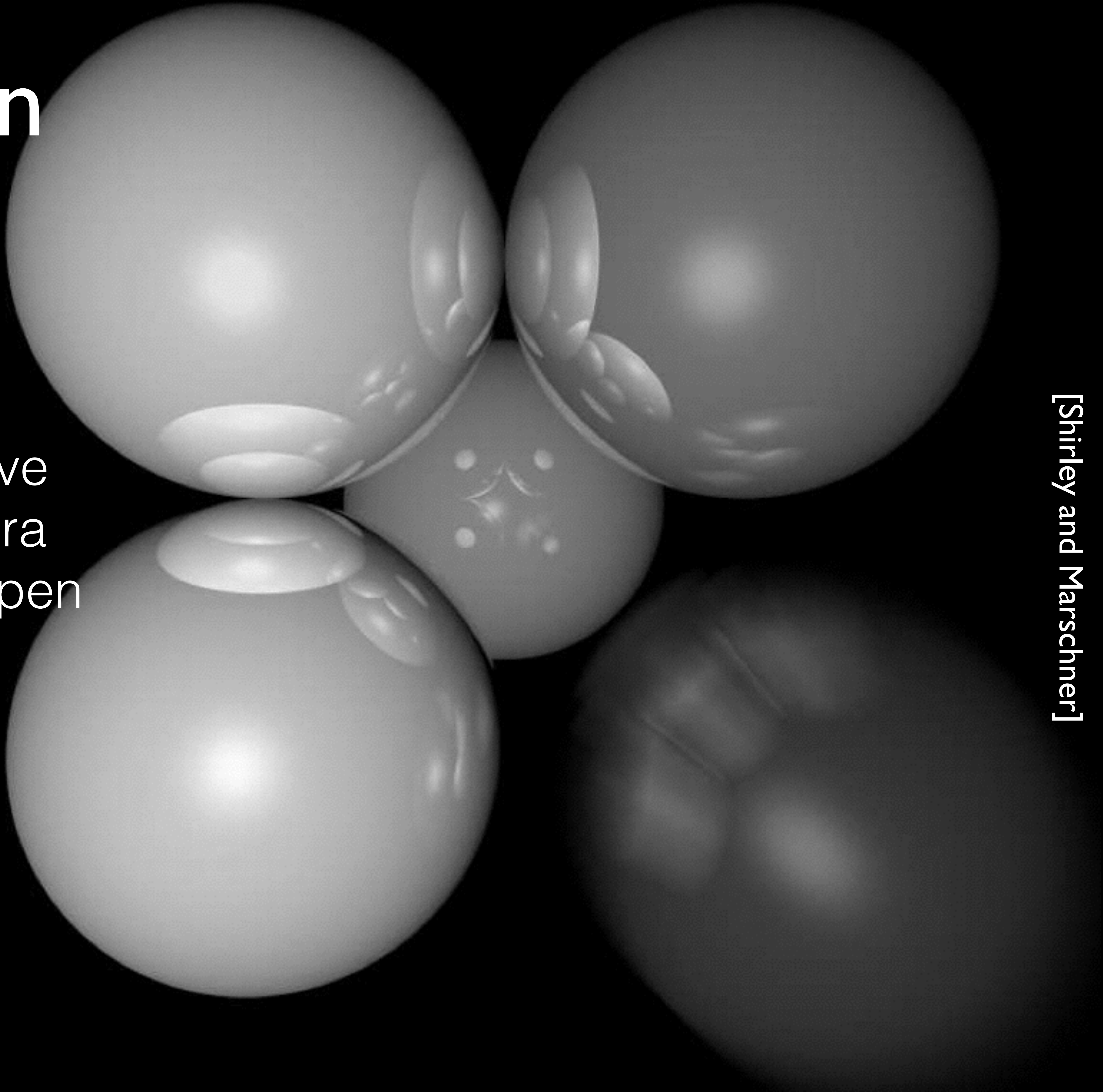
Fuzzy Reflections



[Shirley and Marschner]

Motion Blur

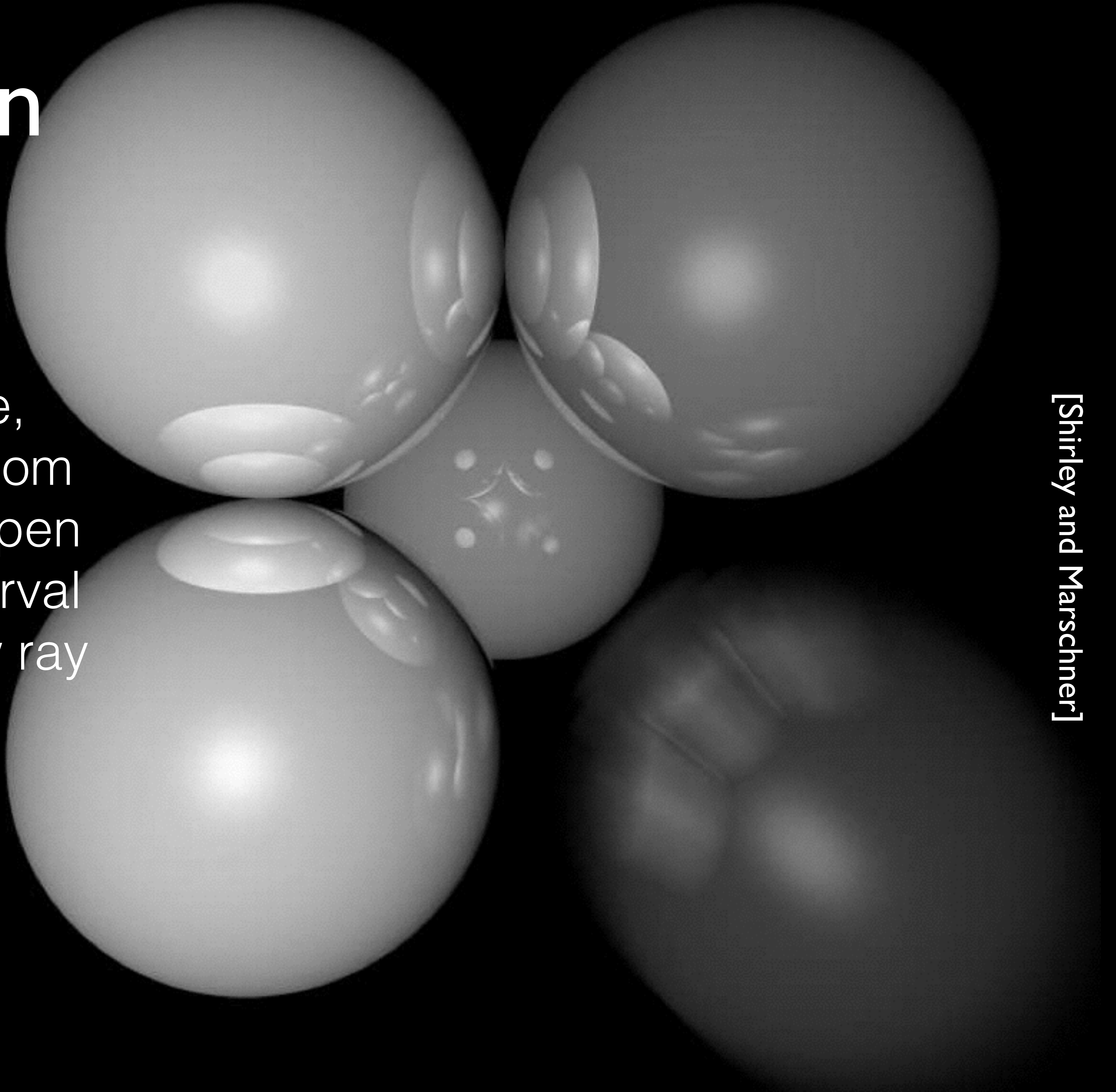
objects move while camera aperture is open



[Shirley and Marschner]

Motion Blur

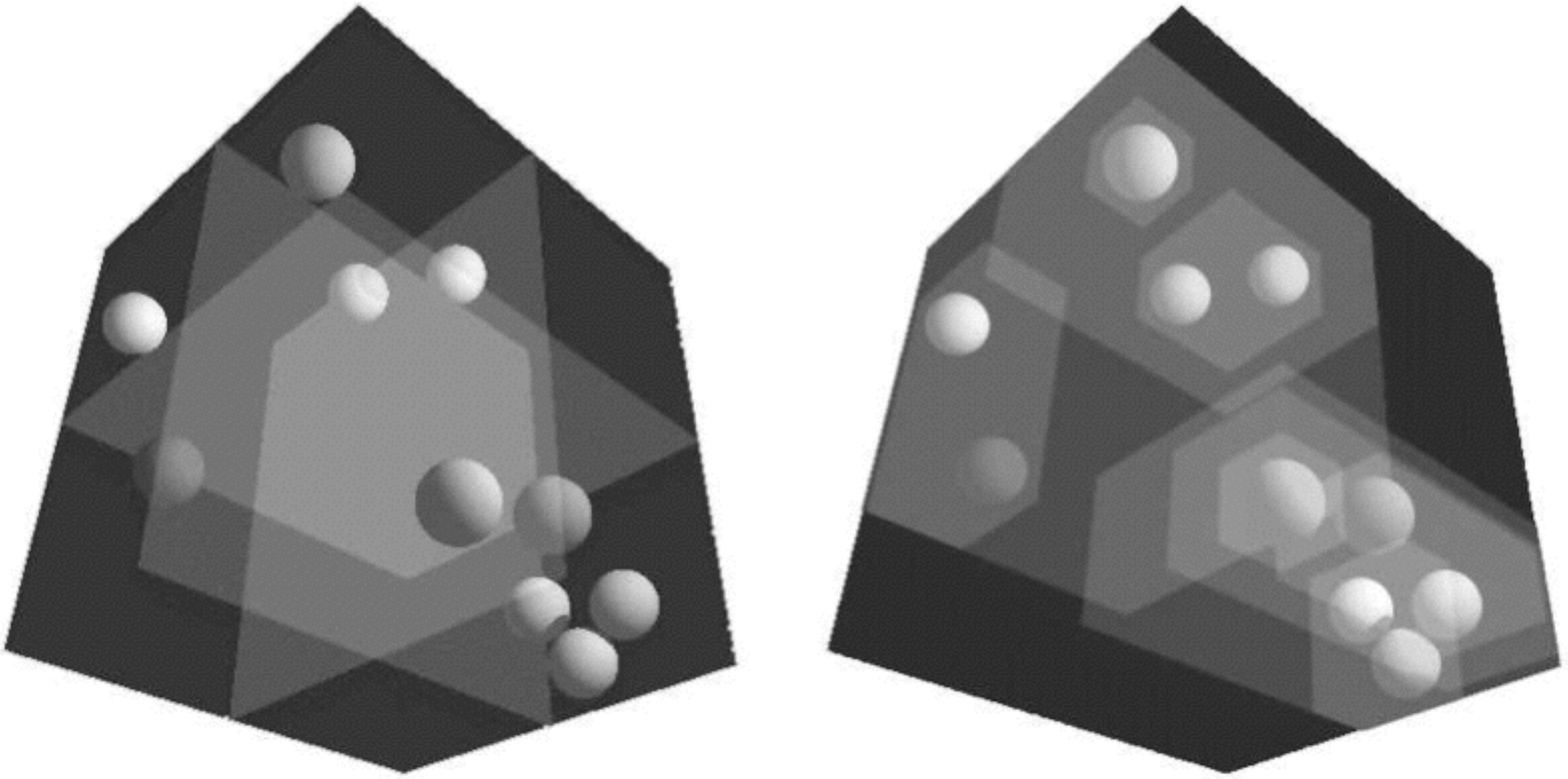
to simulate, choose random time within open aperture interval for each view ray



[Shirley and Marschner]

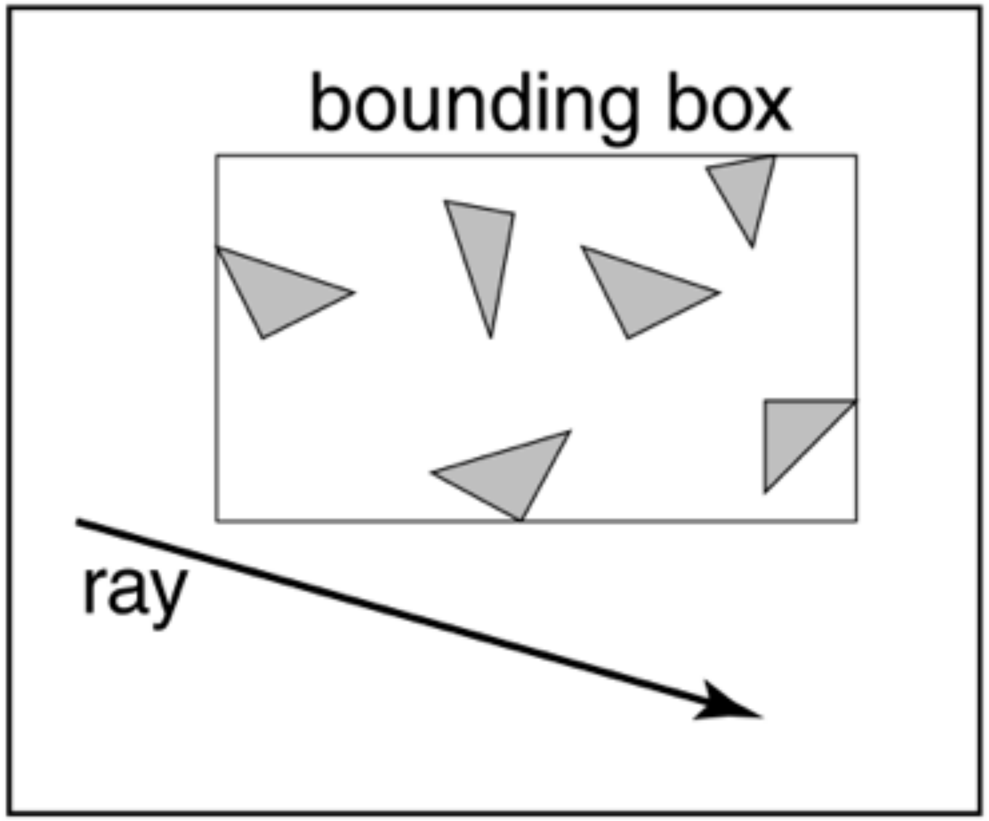
Acceleration Structures

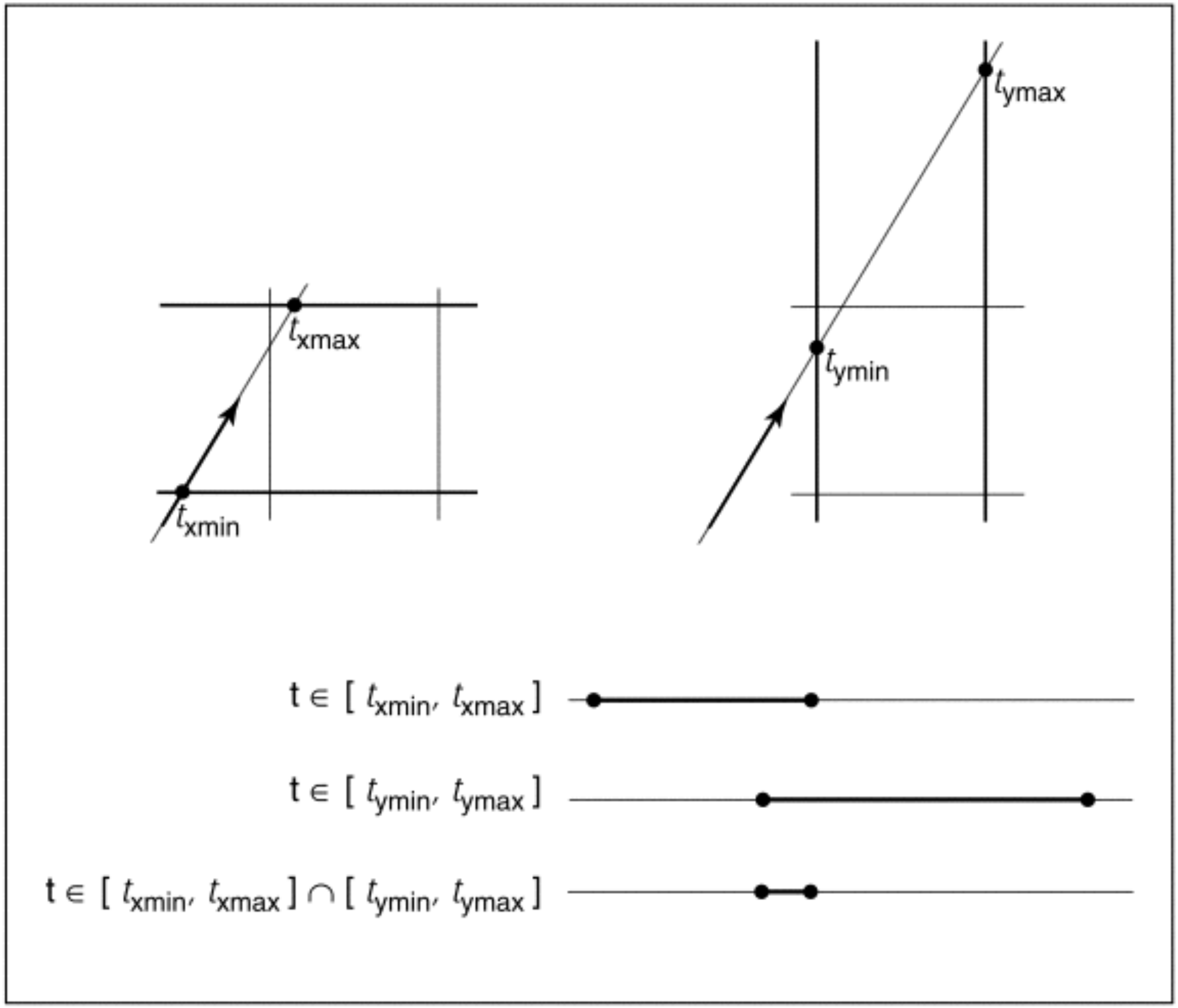
Acceleration Structures



[Shirley and Marschner]

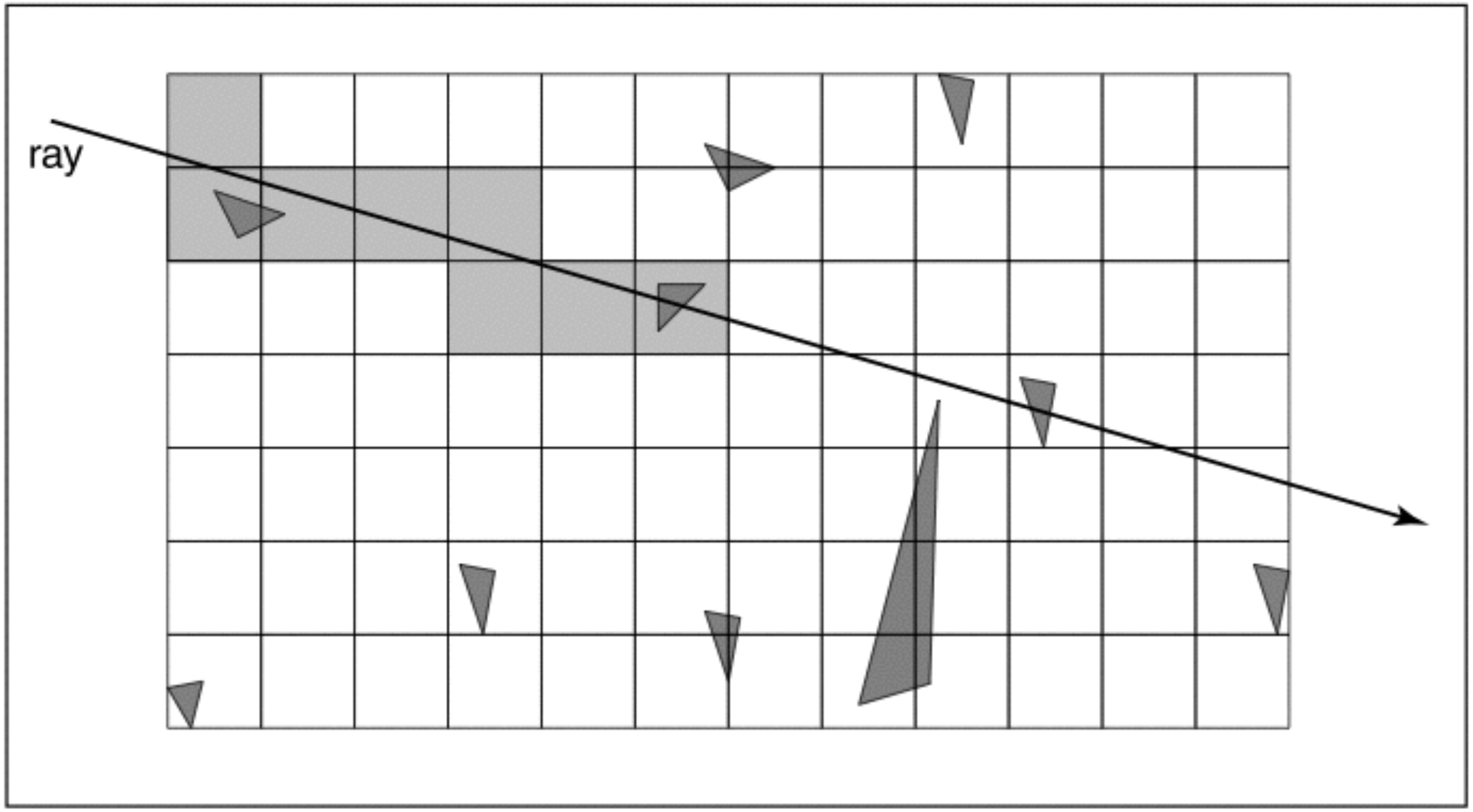
Bounding boxes




t_{xmax}
t_{xmin}
t_{ymax}
t_{ymin}
$t \in [\, t_{xmin}, t_{xmax} \,]$
$t \in [\, t_{ymin}, t_{ymax} \,]$
$t \in [\, t_{xmin}, t_{xmax} \,] \cap [\, t_{ymin}, t_{ymax} \,]$

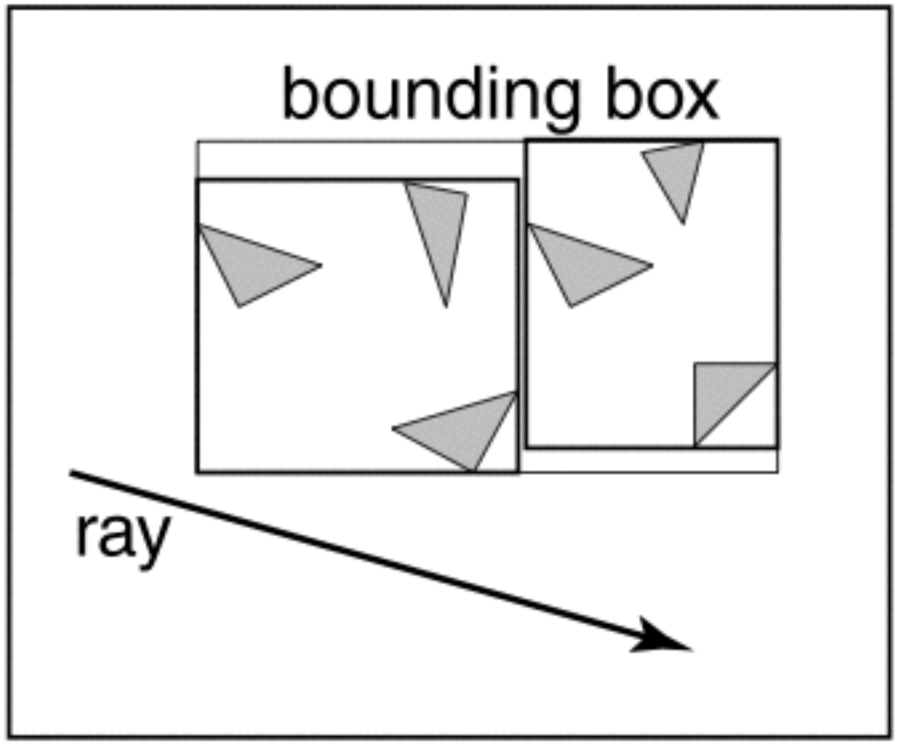
[Shirley and Marschner]

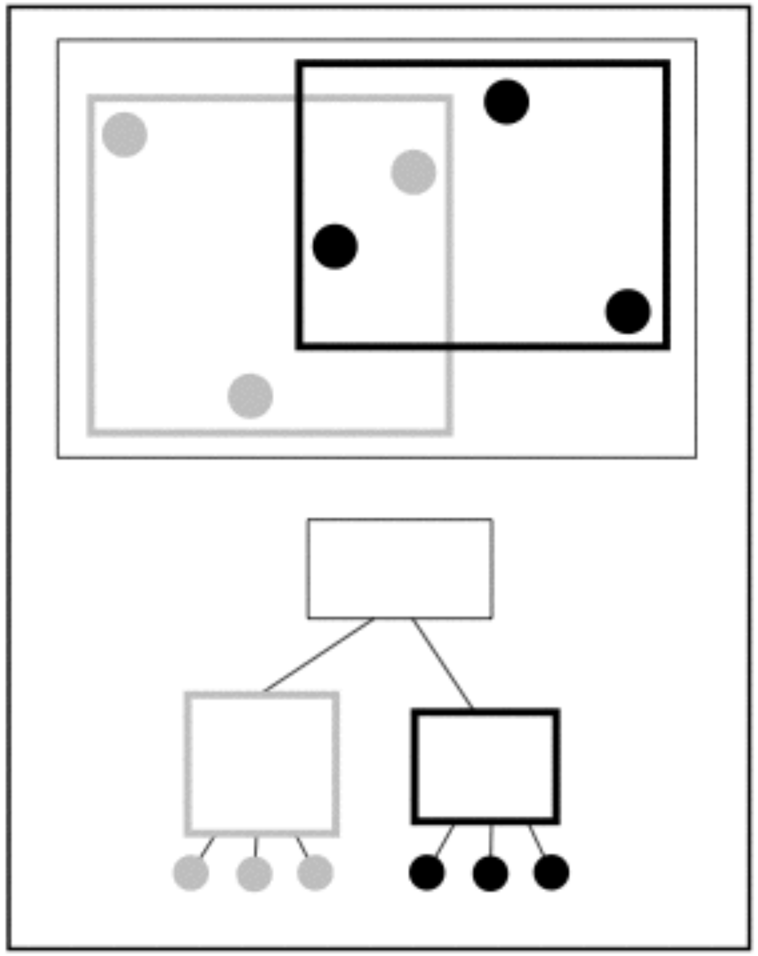
Uniform Spatial Partitioning



[Shirley and Marschner]

Bounding Volume Hierarchy





[Shirley and Marschner]