

CS 130 : Computer Graphics

Ray Tracing

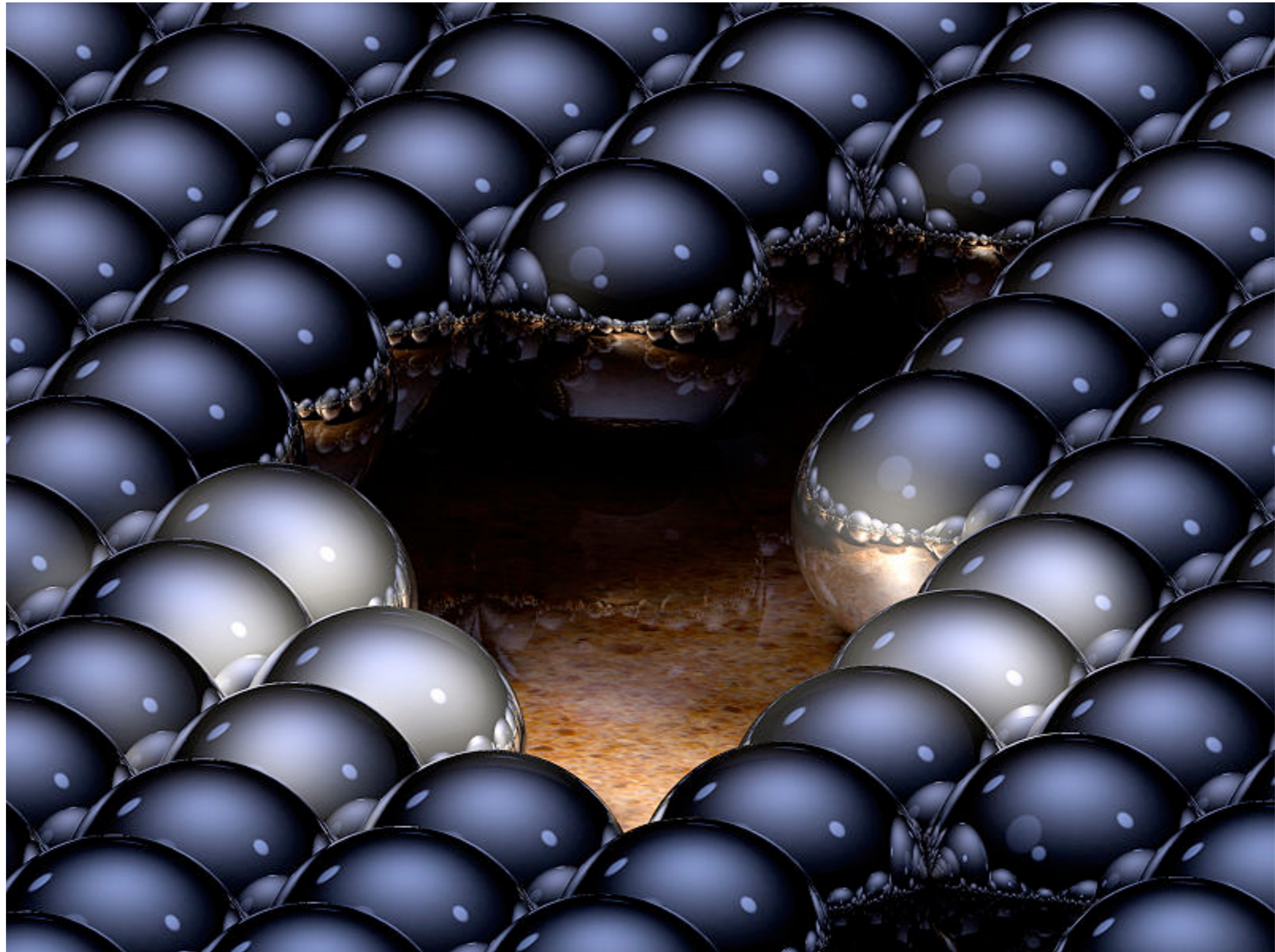
Tamar Shinar

Computer Science & Engineering

UC Riverside

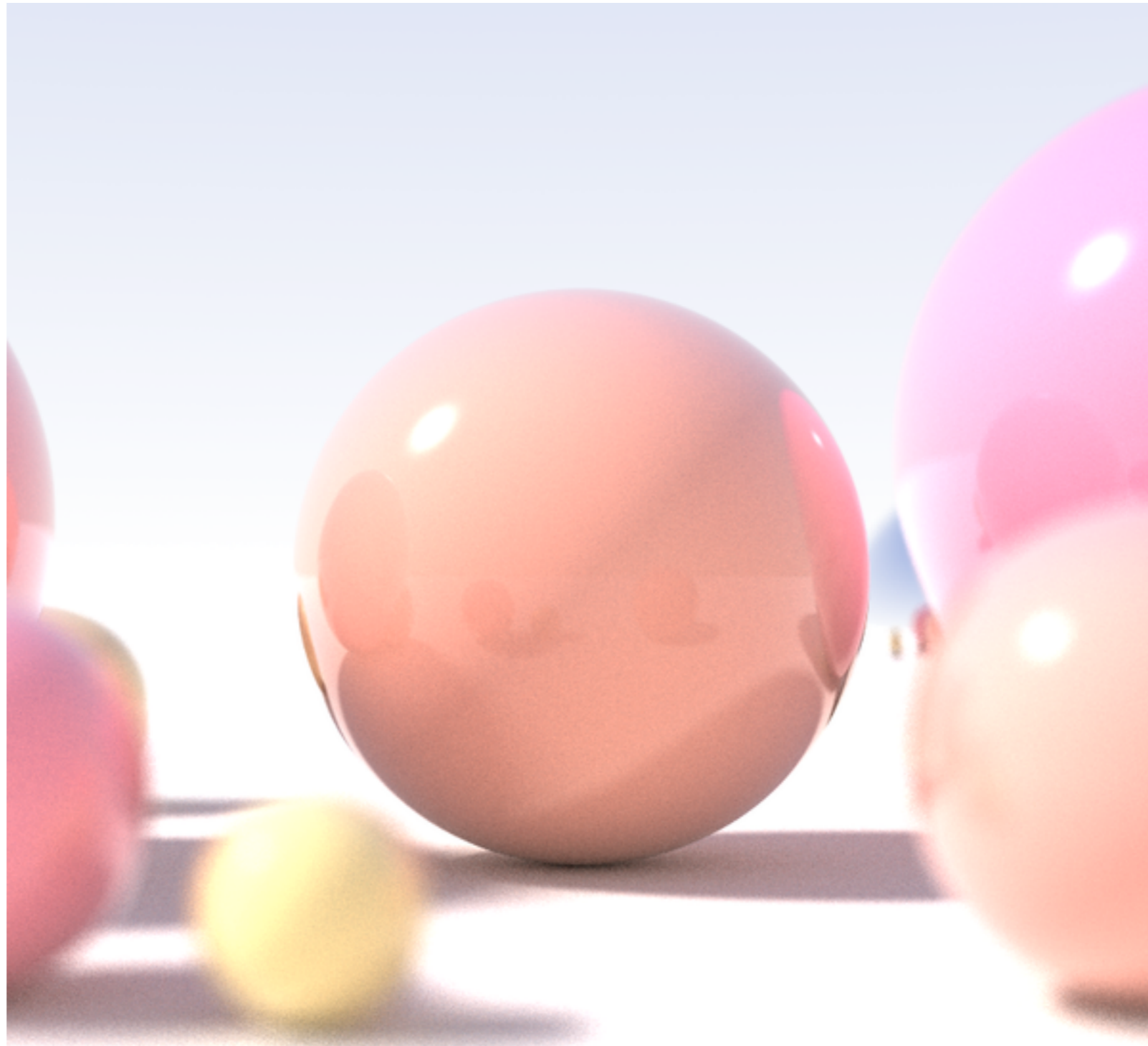


Wikimedia Commons



up to 16 reflections per ray

Greg L., Wikimedia Commons



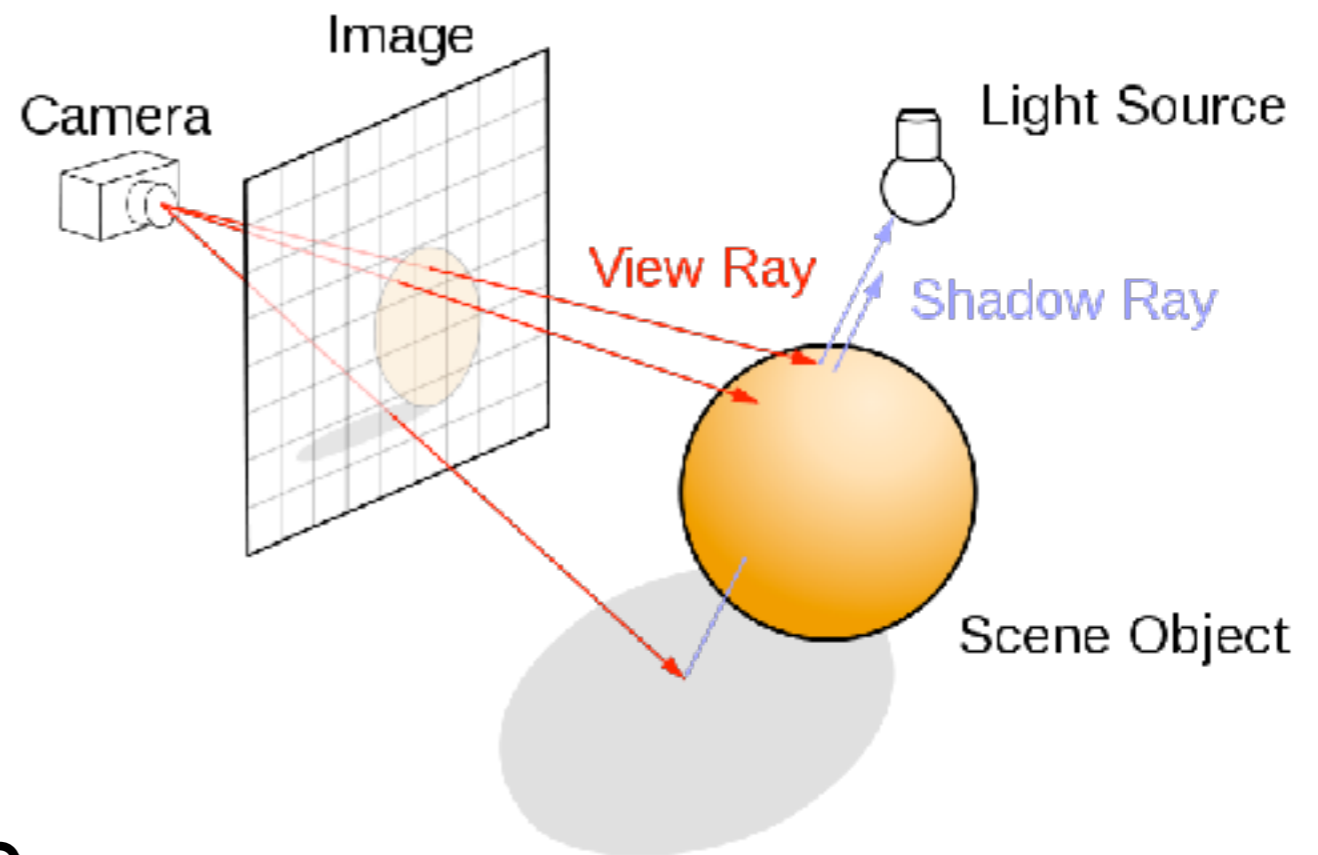
Wikimedia Commons

shallow depth of field, area light sources,
diffuse inter-reflection

Basic Algorithm

for each pixel

1. **cast view ray:** compute view ray from camera through pixel into scene
2. **intersect:** find intersection of ray with closest object
3. **shade:** compute the color of the intersection point

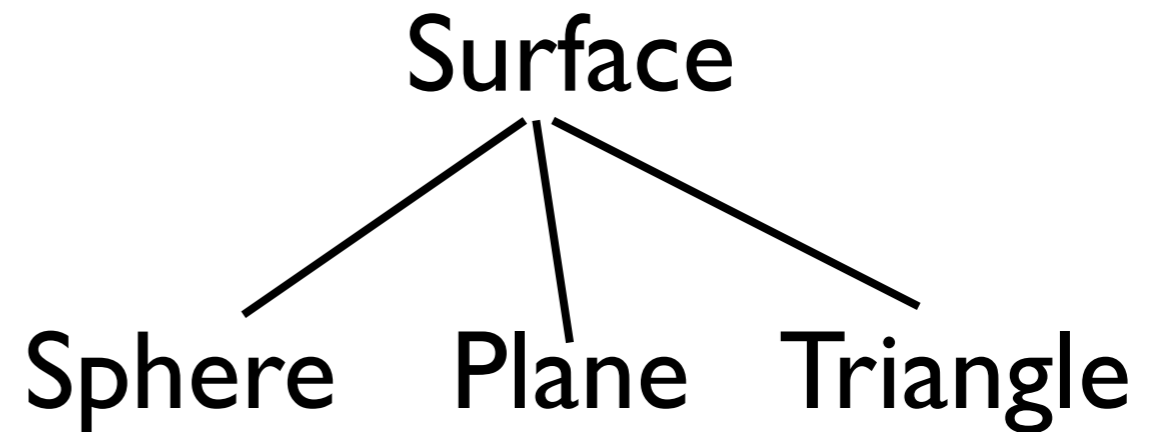


Ray Tracing Program

```
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
```

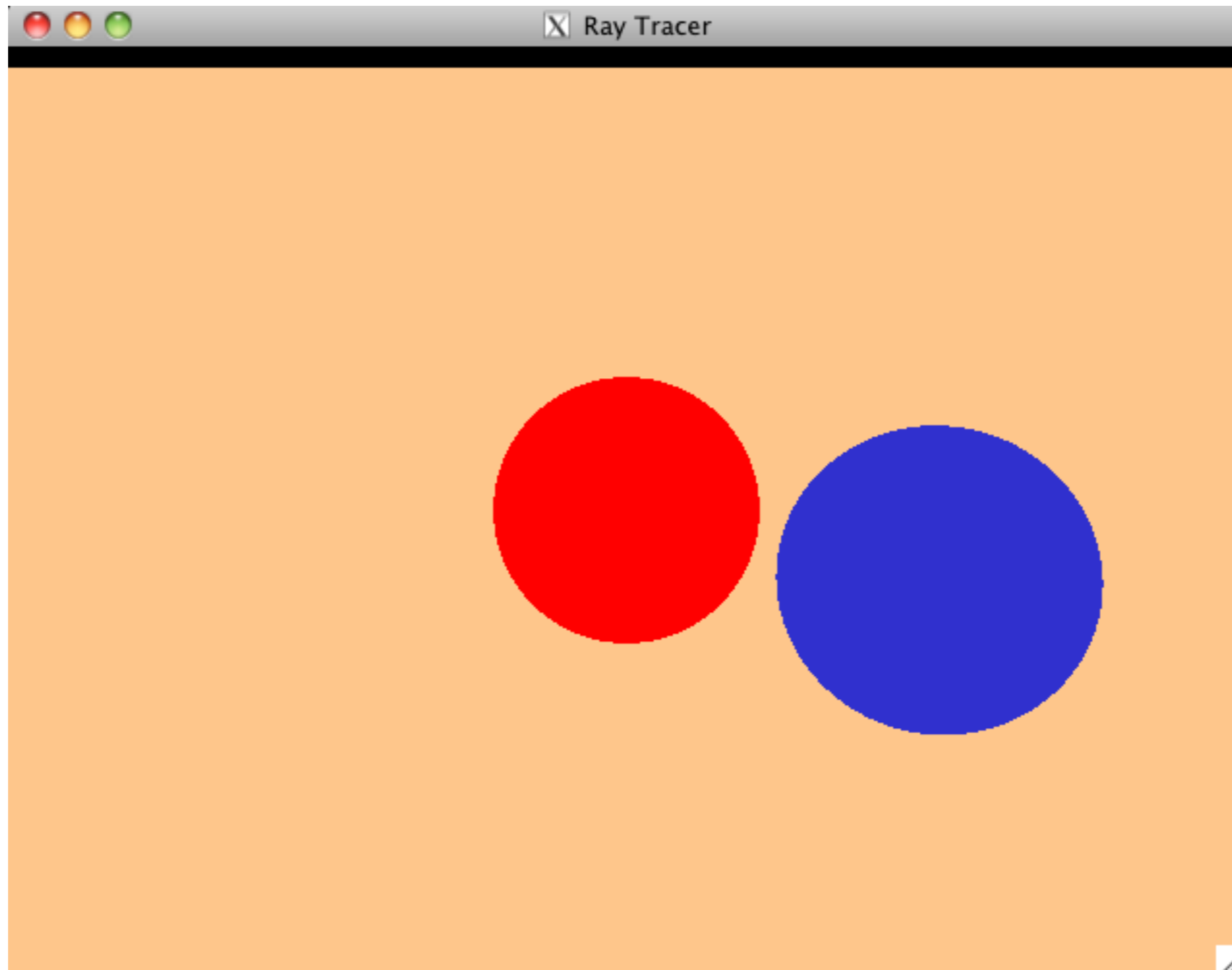
Object-oriented design

```
class Surface
{
    public:
        bool Intersection(RAY& ray)=0;
        Box Bounding_Box()=0;
}
```

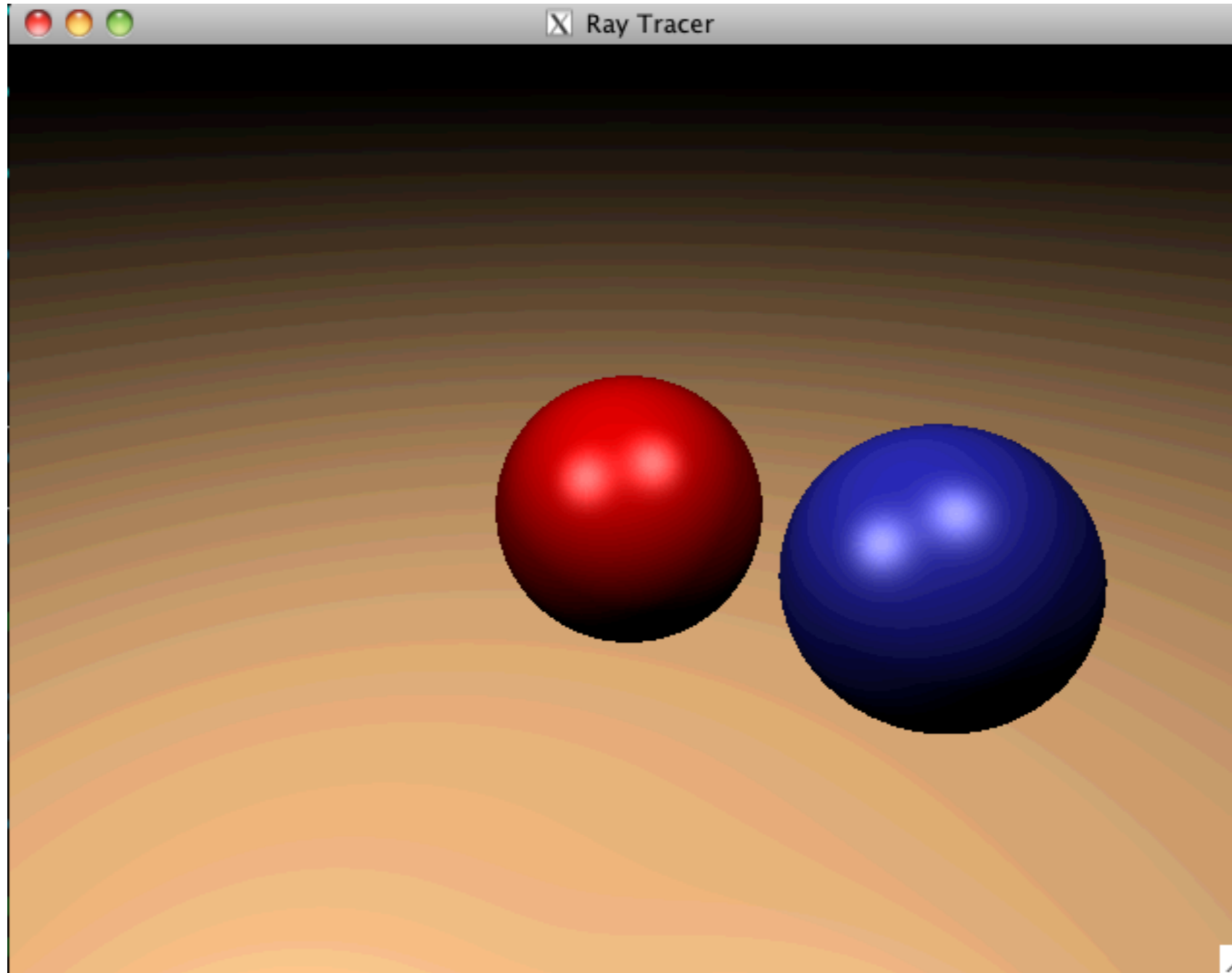


**Other objects: Ray, Light,
Material, Camera, Film, World**

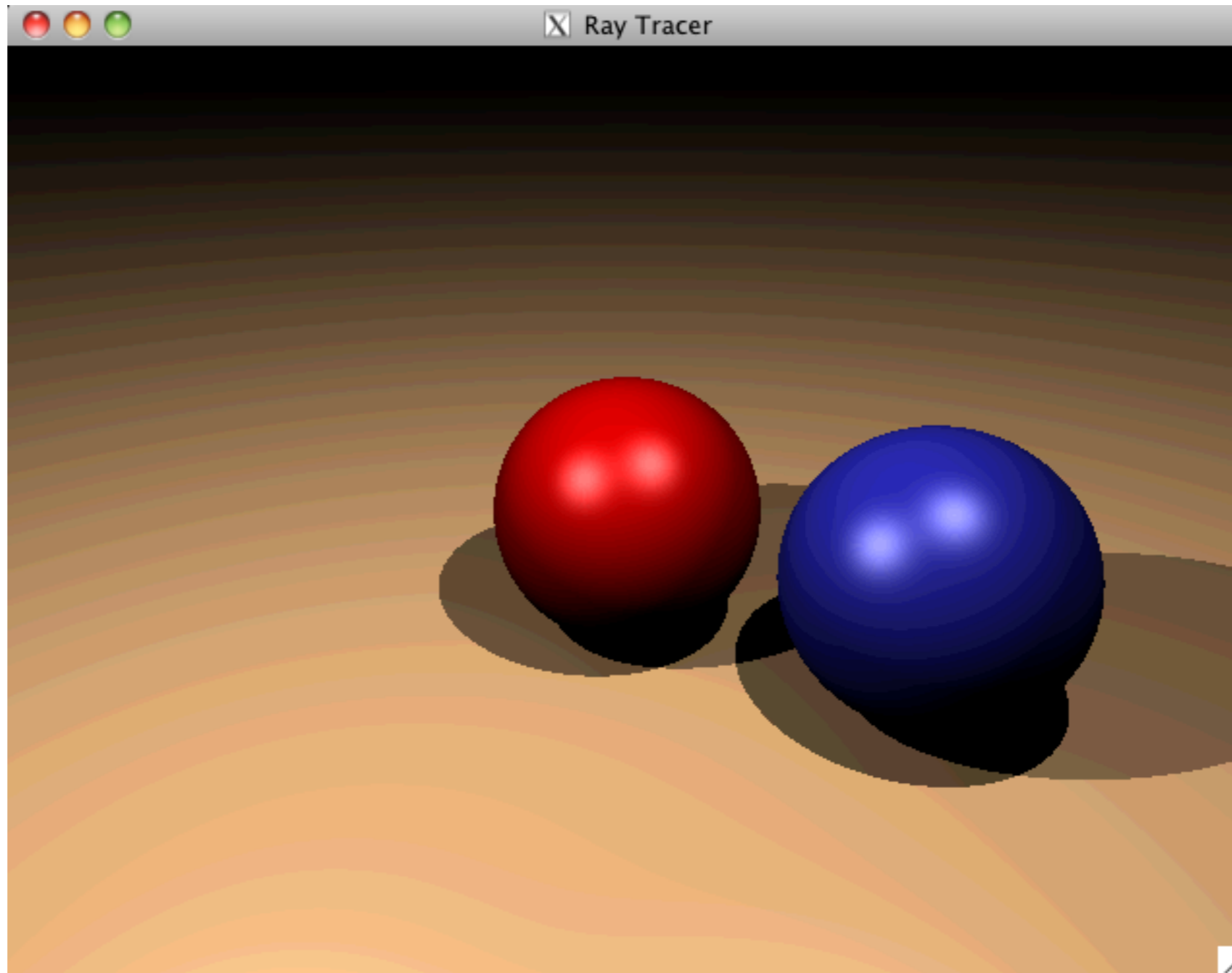
Simple Ray Tracer



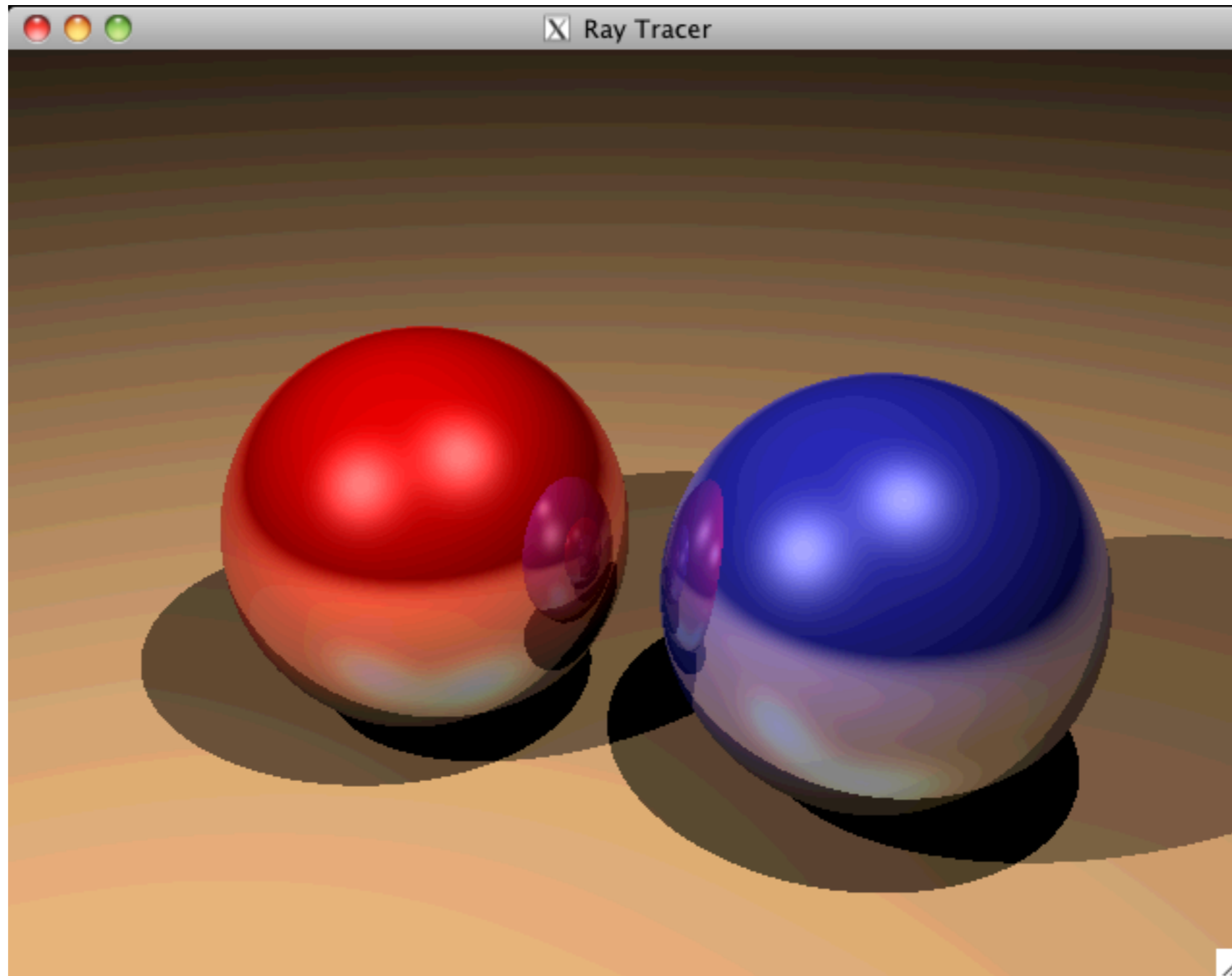
Add Phong Shading



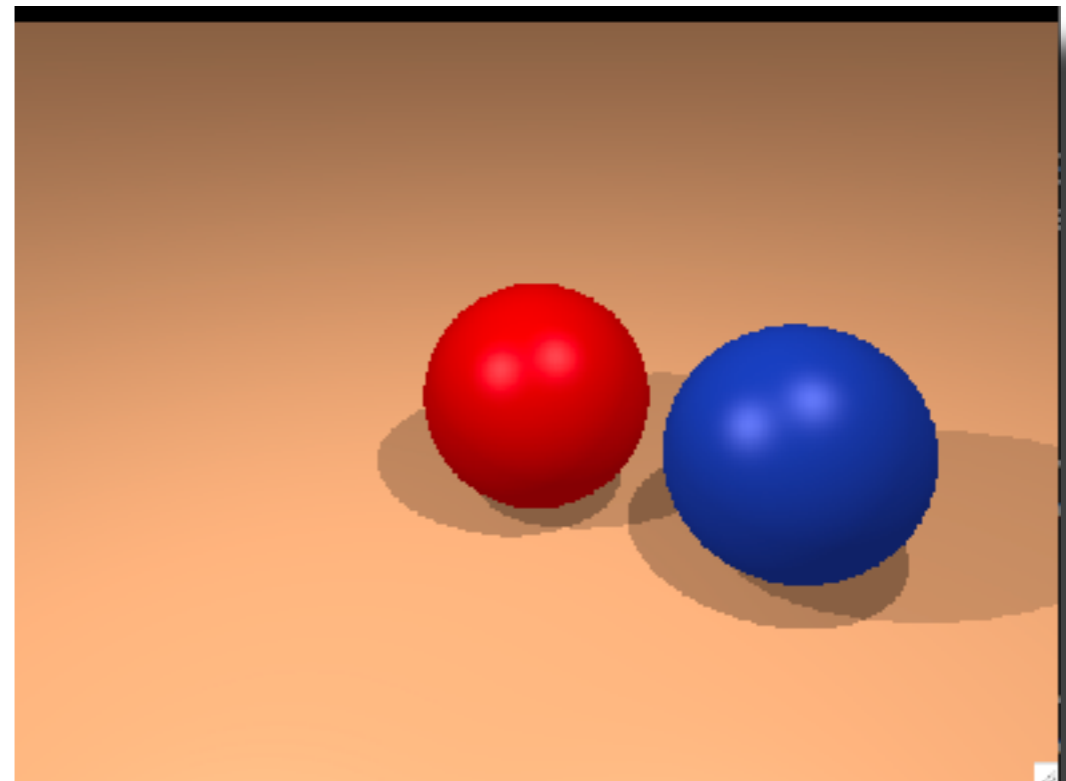
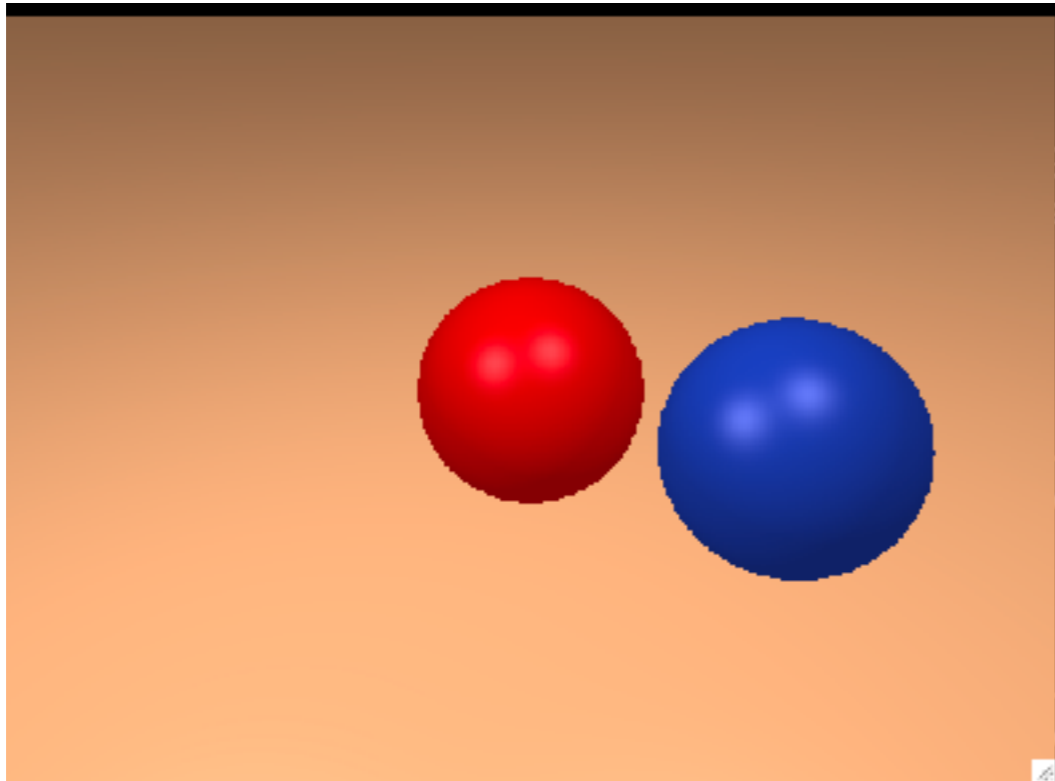
Add Shadows



Add Reflections



Shadows



Shadows

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

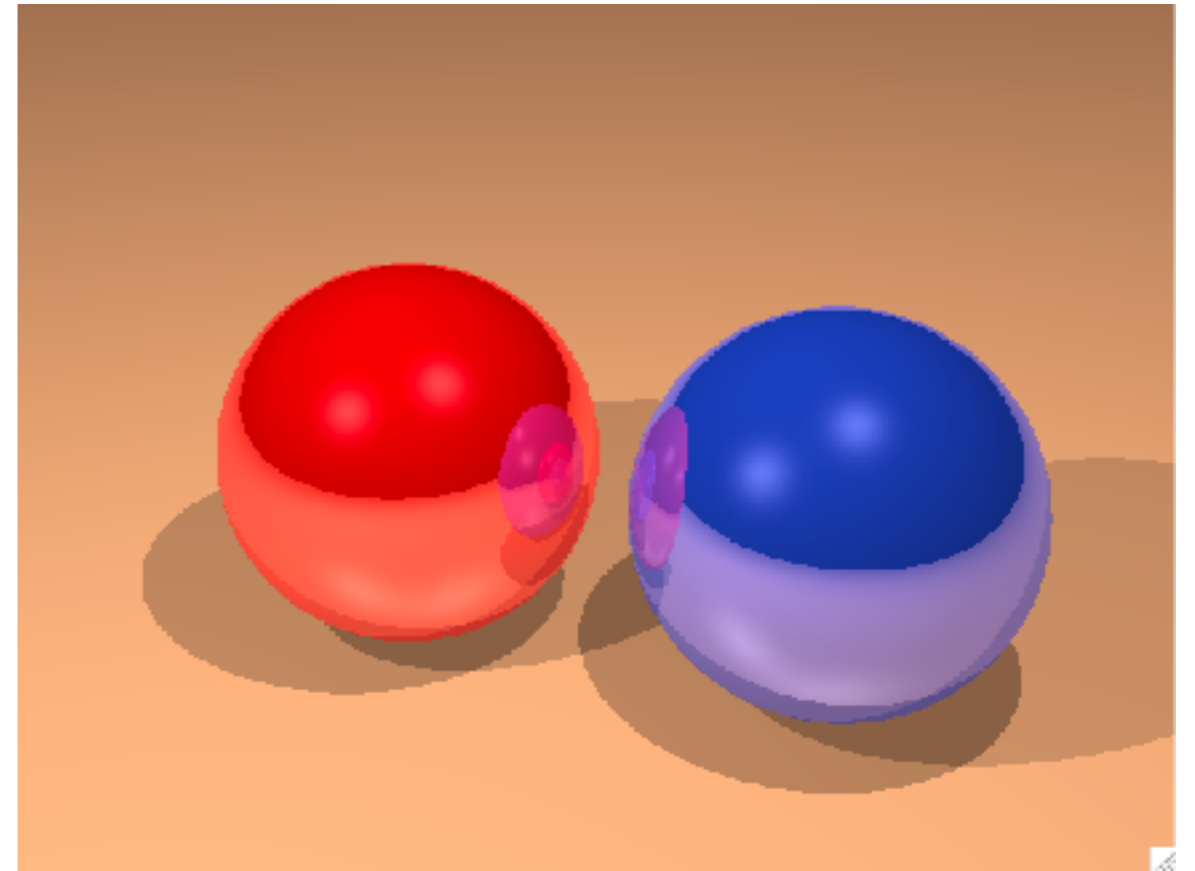
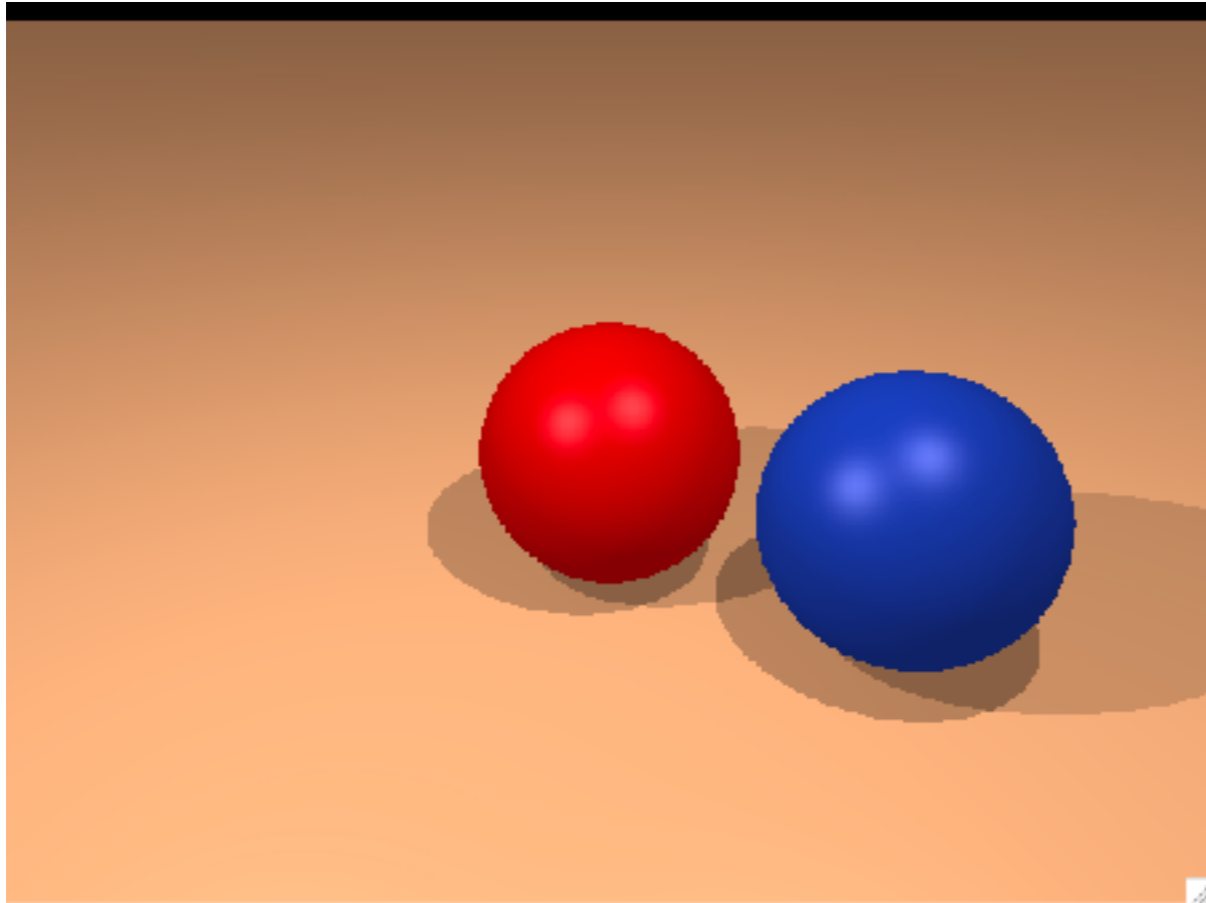
Shadows

```
for each pixel do  
  compute viewing ray  
  if ( ray hits an object with  $t$  in  $[0, \infty]$  ) 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  $[0, \infty]$  ) 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 = (1-k) * color + k * cast_ray(reflected ray)
```