CS130: Computer Graphics
Ray Tracing

Tamar Shinar
Computer Science & Engineering
UC Riverside
up to 16 reflections per ray
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;
}

Sphere   Plane   Triangle

Other objects: Ray, Light,
Material, Camera, Film, World
Simple Ray Tracer
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
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
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

- Reflective_Shader subclass of Phong shader
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

```plaintext
for each pixel do
    compute viewing ray
    pixel color = \text{cast\_ray}(viewing\ ray)

\text{cast\_ray}:
    if (ray hits an object with \( t \) in [0, \( \infty \)]) then
        compute \( n \)
        return color = \text{shade\_surface}
    else
        return color = to the background color

\text{shade\_surface}:
    color = ...
    compute reflected ray
    return color = color + k \times \text{cast\_ray}(reflected\ ray)
```