INFOGR – Computer Graphics
Jacco Bikker & Debabrata Panja - April-July 2018
Lecture 6: “Ray Tracing (2)”

Welcome!
Today’s Agenda:

- Recap
- Normals
- Assignment P1
- Reflections
- Recursion
- Shading models
- TODO
Recap
**Recap**

**Transport**

Energy arriving at an angle:

A small bundle of light arriving at a surface affects a larger area than the cross-sectional area of the bundle. 

Per m², the surface thus receives less energy. The remaining energy is proportional to:

\[ \cos \alpha \quad \text{or} \quad \mathbf{N} \cdot \mathbf{L} \]

**Reflective light:**

\[ E_{\text{to\_eye}} = \sum_{i=1}^{\text{#lights}} \frac{(N \cdot L)}{\text{dist}^2} \circ E_{\text{light}} \circ C_{\text{material}} \]
Today’s Agenda:

- Recap
- Normals
- Assignment P1
- Reflections
- Recursion
- Shading models
- TODO
We Need a Normal

For a plane, we already have the normal.

\[ Ax + By + Cz + D = 0 \]  \text{or}  \quad (P \cdot \mathbf{N}) + D = 0

Fun fact: \( \begin{pmatrix} A \\ B \\ C \end{pmatrix} \) is the normal.

A plane is the set of points that are at distance 0 from the plane.
Distance increases when we move away from the plane.
We move away from the plane by moving in the direction of the normal.

Distance attenuation: \( \frac{1}{r^2} \)

Angle of incidence: \( \mathbf{N} \cdot \mathbf{L} \)
We Need a Normal

Question:

How does light intensity relate to scene size?
i.e.: if I scale my scene by a factor 2, what should I do to my lights?

➔ Distance attenuation requires scaling light intensity by $2^2$

➔ Scene scale does not affect $N \cdot L$. 

```c
// Code snippet
```
We Need a Normal

Question:

What happens when a light is near the horizon?

- Angle approaches 90°; \( \cos \alpha \) approaches 0 (and so does \( \vec{N} \cdot \vec{L} \))

- Light is distributed over an infinitely large surface (so, per unit area it becomes 0)

Note: below the horizon, \( \cos \alpha \) becomes negative.

\( \Rightarrow \) Clamp \( \vec{N} \cdot \vec{L} \) to zero.
Normals

We Need a Normal

Normals are also used to *prevent* shadow rays.

Situation:

A light source is behind the surface we hit with the primary ray:

\[ \mathbf{N} \cdot \mathbf{L} < 0 \]

In this case, visibility is 0, and we do not cast the shadow ray.
We Need a Normal

Normals for spheres:

The normal for a sphere at a point $P$ on the sphere is parallel to the vector from the center of the sphere to $P$.

$$\vec{N}_P = \frac{P - C}{||P - C||} \quad \text{or} \quad \vec{N}_P = \frac{P - C}{r}$$
Normals

We Need a Normal

Normals for spheres:

When a sphere is hit from the inside, we need to reverse the normal.

\[ \vec{N}_P = \frac{C - P}{||P - C||} \]

How to detect this situation when it is not trivial:

1. Calculate the normal in the usual manner \((P - C)\);
2. If \(\vec{N}_P \cdot \vec{D}_{ray} > 0\) then \(\vec{N}_P = -\vec{N}_P\).
Normals

Normal Interpolation

Simulating smooth surfaces using normal interpolation:

1. Generate **vertex normals**.

A vertex normal is calculated by averaging the normals of the triangles connected to the vertex and normalizing the result.

2. Interpolate the normals over the triangle.

In a ray tracer, use barycentric coordinates to do this. Normalize the interpolated normal.
Normals

Normal Interpolation

Using the interpolated normal:

- Use the interpolated normal in the $\vec{N} \cdot \vec{L}$ calculation.
- Use the original face normal when checking if a light is visible.
Today’s Agenda:

- Recap
- Normals
- Assignment P1
- Reflections
- Recursion
- Shading models
- TODO
Assignment P1

1k-sw-raytrace'em all by Tristar & Red Sector Inc. (2004)
Assignment P1

Use That Debug Output!
Assignment P1

Get on Slack!
Today’s Agenda:

- Recap
- Normals
- Assignment P1
- Reflections
- Recursion
- Shading models
- TODO
Reflections
Reflections

Light Transport

We introduce a *pure specular* object in the scene.

Based on the normal at the primary intersection point, we calculate a new direction. We follow the path using a *secondary ray*.

At the primary intersection point, we ‘see’ what the secondary ray ‘sees’; i.e. the secondary ray behaves like a primary ray.

We still need a shadow ray at the new intersection point to establish light transport.
Reflections

Light Transport

For a pure specular reflection, the energy from a single direction is reflected into a single outgoing direction.

- We do not apply $\vec{N} \cdot \vec{L}$
- We do apply absorption

Since the reflection ray requires the same functionality as a primary ray, it helps to implement this recursively.

```cpp
vec3 Trace(Ray ray) {
    I, N, material = scene.GetIntersection(ray);
    if (material.isMirror)
        return material.color * Trace(...);
    return DirectIllumination() * material.color;
}
```
Reflections

Light Transport

For pure specular reflections we do not cast a shadow ray.

Reason: *Light arriving from that direction cannot leave in the direction of the camera.*
Reflections
Reflections

Partially Reflective Surfaces

We can combine pure specularity and diffuse properties.

Situation: our material is only 50% reflective.

In this case, we send out the reflected ray, and multiply its yield by 0.5. We also send out a shadow ray to get direct illumination, and multiply the received light by 0.5.
Reflections

Reflecting a HDR Sky

A dark object can be quite bright when reflecting something bright.

E.g., a bowling ball, pure specular, color = (0.01, 0.01, 0.01); reflecting a ‘sun’ stored in a HDR skydome, color = (100, 100, 100).

For a collection of HDR probes, visit Paul Debevec’s page:

http://www.pauldebevec.com/Probes
Today’s Agenda:

- Recap
- Normals
- Assignment P1
- Reflections
- Recursion
- Shading models
- TODO
Recursion

Whitted-style Ray Tracing, Pseudocode

```cpp
Color Trace( vec3 O, vec3 D )
{
  I, N, mat = IntersectScene( O, D );
  if (!I) return BLACK;
  return DirectIllumination( I, N ) * mat.diffuseColor;
}

Color DirectIllumination( vec3 I, vec3 N )
{
  vec3 L = lightPos - I;
  float dist = length( L );
  L *= (1.0f / dist);
  if (!IsVisible( I, L, dist )) return BLACK;
  float attenuation = 1 / (dist * dist);
  return lightColor * dot( N, L ) * attenuation;
}
```

Todo:
- Implement IntersectScene
- Implement IsVisible
Recursion

Whitted-style Ray Tracing, Pseudocode

```
Color Trace( vec3 O, vec3 D )
{
    I, N, mat = IntersectScene( O, D );
    if (!I) return BLACK;
    if (mat.isMirror())
    {
        return Trace( I, reflect( D, N ) ) * mat.diffuseColor;
    }
    else
    {
        return DirectIllumination( I, N ) * mat.diffuseColor;
    }
}
```

Todo:
Handle partially reflective surfaces.
Recursion

Whitted-style Ray Tracing, Pseudocode

```cpp
Color Trace( vec3 O, vec3 D )
{
   I, N, mat = IntersectScene( O, D );
   if (!I) return BLACK;
   if (mat.isMirror())
   {
      return Trace( I, reflect( D, N ) ) * mat.diffuseColor;
   }
   else if (mat.IsDielectric())
   {
      f = Fresnel( ... );
      return (f * Trace( I, reflect( D, N ) )
           + (1-f) * Trace( I, refract( D, N, ... ) ) ) * mat.DiffuseColor;
   }
   else
   {
      return DirectIllumination( I, N ) * mat.diffuseColor;
   }
}
```

Todo:
- Implement reflect
- Implement refract
- Implement Fresnel
- Cap recursion
Recursion

Spheres: pure specular
Recursion

Spheres: 50% specular
Recursion

Spheres: one 50% specular, one glass sphere
Recursion
Recursion

Ray Tree

Recursion, multiple light sampling and path splitting in a Whitted-style ray tracer leads to a structure that we refer to as the *ray tree*.

All energy is ultimately transported by a single primary ray.

Since the energy does not increase deeper in the tree (on the contrary), the average amount of energy transported by rays decreases with depth.
Today's Agenda:

- Recap
- Normals
- Assignment P1
- Reflections
- Recursion
- Shading models
- TODO
Shading

Diffuse Material

A diffuse material scatters incoming light in all directions.

Incoming:

\[ E_{\text{light}} \times \frac{1}{\text{dist}^2} \times \vec{N} \cdot \vec{L} \]

Absorption:

\[ C_{\text{material}} \]

Reflection:

\[ (\vec{V} \cdot \vec{N}) \]

Eye sees:

\[ \frac{1}{(\vec{V} \cdot \vec{N})} \]

terms cancel out.

A diffuse material appears the same regardless of eye position.
Shading

Specular Material

A specular material reflects light from a particular direction in a single outgoing direction.
Shading

```cpp
float diffuse(const vector3f& n, const vector3f& r, const vector3f& s, float pdf)
{
    float N = (dot(n, r) > 0.0f) ? 1.0f : -1.0f;
    float x = (dot(n, s) > 0.0f) ? 1.0f : -1.0f;
    float t = dot(r, s) / dot(n, r);
    float s = sqrt(1.0f - x * x - t * t);
    vector3f P = n * x + s * t;
    return 1.0f - (N * x * x + x * t * s + t * t) / 2.0f;
}
```
Shading

Glossy Material

A glossy material reflects *most* light along the reflected vector.

\[ \vec{R} = \vec{L} - 2(\vec{L} \cdot \vec{N})\vec{N} \]

For other directions, the amount of energy is:

\[ (\vec{V} \cdot \vec{R})^\alpha \], where exponent \( \alpha \) determines the specularity of the surface.
Shading

Phong Shading

Complex materials can be obtained by blending diffuse and glossy.

\[ I = C_{\text{material}} \cdot (1 - f_{\text{spec}})(\vec{N} \cdot \vec{L}) + f_{\text{spec}}(\vec{V} \cdot \vec{R})^\alpha \]

where

\[ \alpha \] is the specularity of the glossy reflection;
\[ f_{\text{spec}} \] is the glossy part of the reflection;
\[ 1 - f_{\text{spec}} \] is the diffuse part of the reflection.

Note that the glossy reflection only reflects light sources.
Today’s Agenda:

- Recap
- Normals
- Assignment P1
- Reflections
- Recursion
- Shading models
- TODO
TODO

Limitations of Whitted-style
Limitations of Whitted-style
TODO

Limitations of Whitted-style
TODO

Limitations of Whitted-style
TODO

Limitations of Whitted-style
INFOGR – Computer Graphics
Jacco Bikker & Debabrata Panja - April-July 2018

END OF lecture 6: “Ray Tracing (2)”

Next lecture: “Accelerate”