1 Phong model

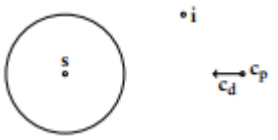
Explain the four components used in Phong model and how they differ between black rubber and glossy plastic.

Black rubber No emission Ce
Ambient Ca = (0,0,0)
Diffuse Cd = (0, 0, 0)
Specular Cs = (0, 0, 0)
Specular exponent f = 1 or at least < 10

Glossy plastic No emission Ce
Ambient Ca = (0,0,0)
Diffuse Cd = (r, g, b) depending on color of plastic
Specular Cs = (1, 1, 1) and ks $<= 1$
Specular exponent f between 10 and 100

Compute the colour of a spheres image's centre pixel using the Phong model given a sphere with material components The colour is computed from the Phong reflection model equation:
$C = C_e + C_a + C_d(w_i * n) + C_s(v * w_r)^f$
The given components are:

- $C_e = (0, 0, 0)$
- $C_a = (0, 0, 0)$
- $C_d = (0.5, 0.5, 0.5)$
- $C_s = (0.5, 0.5, 0.5)$
- $f = 1$
- $r = 1$ - radius
- $s = (0, 0, 0)$ - centre point
- $c_p = (3, 0, 0)$ - camera position
- $c_d = (-1, 0, 0)$ - camera direction
- $i = (2, 0, 1)$ - isotropic white point light position



We have white light $I = (1, 1, 1)$, so we omit the I colour terms from the equation. For a unit sphere, the normal vector is the same as the position vector of the surface point, so $n = (1, 0, 0)$.

The direction to the light is $i - h = (2, 0, 1) - (1, 0, 0) = (1, 0, 1)$.

Normalising the vector gives $w_i = \frac{(1,0,1)}{||(1,0,1)||} = (\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}})$.

The direction to the camera is $c_p - h = (3, 0, 0) - (1, 0, 0)$, so the normalised view vector is $v = (1, 0, 0)$.

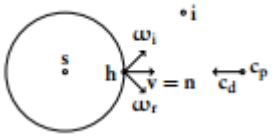
Since the normal is parallell to the x axis, is is easy to compute the mirror direction $w_r = (\frac{1}{\sqrt{2}}, 0, -\frac{1}{\sqrt{2}})$.

We plug these into the equation:

$C = C_e + C_a + C_d(\frac{1}{\sqrt{2}}) + C_s(\frac{1}{\sqrt{2}})^f$

Since the colours of the sphere's materials have equal rgb components, the colour of the pixel will be $C = (k, k, k)$ where

$k = 0 + 0 + \frac{1}{2\sqrt{2}} + \frac{1}{2\sqrt{2}} = \frac{1}{\sqrt{2}}$



2 Rendering equation

$$L(P, \omega_o) = \underbrace{L^e(P, \omega_o)}_{A} + \int_{\omega_i \in S^2(P)} \underbrace{L(P, -\omega_i)}_{B} \underbrace{f_s(P, \omega_i, \omega_o)}_{C} \underbrace{(\omega_i \cdot n_P)}_{D} d\omega_i.$$

Figure 1: Rendering equation

Describe what the terms A-D mean in the rendering equation

- A: Light emitted from P in direction w_o
- B: Incoming light to P from direction w_i
- C: How much of the light of the that arrives at P from w_i scatters in direction w_0
- D: The cosine of the angle between the incoming light and the normal at P (assuming normalised vectors)

Describe how the rendering equations is implemented in ray tracing and path tracing Path tracing computes the integral "depth first" by shooting a single secondary ray per recursion, where a ray hits a surface. The

rendering progresses by iterating over all pixels multiple times, each time sampling a new w_i direction in order to better estimate the integral. In that sense, it is more of an "any-time" algorithm than basic ray tracing. You get a noisy result, which gets better the longer you wait.

In basic ray tracing, a single ray is shot per pixel, and the integral is estimated by a fixed number of samples: one per light source with w_i = the direction to the light source, optionally a reflected recursive ray and optionally a transmitted recursive ray. This gives a low-variance result, but with a bias compared to the true solution. In "distribution ray tracing", at any recursion step, multiple rays may be emitted, but this can quickly lead to a combinatorial explosion of rays ("breadth-first").

Describe why the rendering equation is difficult to solve in general. List two reasons, and mention why those are problematic

- It is an integral equation, where the function $L(P,w_o)$ that we want to find is also on the right hand side of the equation, inside an integral. (We need to simplify the B part to be able to solve it)
- The integral cannot usually be solved analytically (algebraically). (Infinite amount of summarisations)
- It involves shadowing from other obstacles, so we need to evaluate it numerically instead, using discrete samples. Most modern methods use randomised samples: Monte Carlo integration.
- Choosing the samples wisely, to get low variance (little noise) but still get an image decently fast, is nontrivial.
- Not only reflections from surfaces contribute to the incoming light. There could also be light scattered in patricipating media.
- Subsurface scattering is not explicitly included in this version of the rendering equation. One would also need a second integral, integrating over output points, not just out direction, to cover that.

3 Matrix calculations

$$\mathbf{T}_i \mathbf{p} = \mathbf{T}_i \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \mathbf{p}'.$$

$$\mathbf{T}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{T}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{T}_3 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{T}_4 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

What does each of the following transformation matrices do, when left-multiplied with the position vector p of a point:

- T1: Does nothing (identity matrix)
- T2: Translates 1 point along the z-axis
- T3: Rotates -90 degrees counter-clockwise around z
- T4: Shears perpendicular to the z-axis

Describe how you would implement the transformation between and object's local frame and the world frame without using homogenous coordinates. Also motivate why it is convenient to use homogeneous coordinates. Transforming an object from its local coordinate frame tot he world frame is typically done by first rotation the object so that it has the desired orientation with respect to the world frame, and then translate it.

With homogeneous coordinates, it is possible to store all transformations as matrices and append transformations just by multiplying the matrices together. Without homogeneous coordinates, translations would need to be implemented with vector addition rather than matrix multiplication.

Using homogeneous coordinates allows all common operations - translation, rotation, scaling as well as perspective projection - to be implemented as matrix operations. This means that any sequence of several translations and rotations can still be represented using a single matrix. Another advantage is using homogeneous coordinates, infinity can be represented with real numbers, [x,0].

4 Shadow maps

Outline steps for a basic shadow mapping algorithm

1. Render scene from light source's position
2. Save the corresponding depth buffer
3. Render scene from camera view
4. Transform each pixel's coordinates to light space
5. If z (distance from light source) is greater than the corresponding value in the shadow map, then the pixel is in shadow.

List four advantages of shadow maps over shadow volumes, or vice versa, for real-time shadow rendering

Shadow maps

- can generate shadows for any rasterisable geometry
- no need to compute silhouette edges
- no need to generate additional geometry
- no need to care about whether camera is in shadow or not
- faster (just one extra rendering pass per directional light source)
- faster (doesnt affect fill rate)

Shadow volumes

- have no issue with biasing
- have no issue with aliasing
- works well with omnidirectional light sources

Augment your shadow mapping algorithm so that the algorithm also takes into account z fighting and shadow map aliasing.

- Add a depth bias, to prevent z-fighting
- Describe one of the percentage closest filtering, shadow map fitting, warping or partitioning