

Lecture 14: Shadows & Interpolation Shading

Adam Hawley

December 22, 2018

1 Shadowing

There are two ways a point can be in a shadow: firstly, points can be in shadow because they face away from the light source or they can be in darkness caused by another point. The first is called a self/attached shadow while the second is a cast shadow.

1.1 Self-Shadowing

How can we test for self shadows? The check is based on whether the angle between the surface normal and the light source is greater than 90° :

$$\theta_i > 90^\circ \implies \text{Self shadow}$$

Equivalently, check whether the dot product is negative:

$$\mathbf{n} \cdot \mathbf{s} < 0 \implies \text{Self shadow}$$

This is a very efficient process as it depends on local information only. Normals are already computed for the reflectance calculation, simply check dot product.

Self shadowing can be incorporated into the Lambertian model since Lambertian shading uses the same calculation. (The dot product between the surface normal and the light source) To make use of the pre-calculated dot product, simply clamp negative values to zero:

$$I = \max(0, \mathbf{n} \cdot \mathbf{s})$$

Note: we did not consider this in either the shape-from-shading or photometric stereo algorithms in the previous lecture. So self shadows are not accounted for. “Add noise to the solution”

1.2 Cast Shadows

Case shadows are much more difficult to calculate. A point can be shadowed by a distant part of the surface. (They depend on global geometry) To check whether a point is in shadow, we must consider all other points on the surface.

If we set height values for every pixel (x, y) as $z(x, y)$. Then the cast shadow test can be written as:

$$\forall d \in \mathbb{R}^+, z(x, y) + ds_z > z(x + ds_x, y + ds_y) \implies \text{Not in cast shadow}$$

For general polygonal surfaces it is more complex: For every polygon or vertex, compute equation of line to light source. Then test whether the line intersects any other polygon in the scene e.g:

- Compute plane equation for every polygon
- Compute line/plane intersection
- Test whether intersection is inside or outside extent of polygon

There are efficient ways to calculate this.

2 Flat Shading

For each planar polygon we compute the surface normal. The surface normal is constant over the polygon and used to compute a single colour for the polygon. The whole polygon is then filled with this colour. For objects which consist of only planar surfaces, it can look fine. However it generally looks bad for polygonal approximations to curved surfaces.

Flat shading has a very bad effect with curved surfaces. **Mach banding** caused by **lateral inhibition** means that the human eye effectively has a built in edge detector. Perception of change in intensity exaggerates actual change.

To smoothly vary the colour of a polygon across the surface we compute a surface normal for each vertex. The meaning of the normal direction to a point is not well defined so there are two options:

- If the surface is described by a parametric function (e.g a sphere or cylinder), just **derive an analytical expression for the surface normal** from the function.
- If the surface is an arbitrary polygonal mesh, compute **one normal per polygon and combine** these in some way to find vertex normals.

3 Face Normals

Typically, a surface is represented by a list of vertices and a list of faces. Faces are specified such that vertex ordering is clockwise or counterclockwise when viewed from outward facing side. To compute a normal to a face, use the cross product of two edge vectors:

$$\mathbf{n} = \mathbf{e}_1 \times \mathbf{e}_2$$

Remember: The length of \mathbf{n} gives twice the area of the triangle. To find \mathbf{n} as a unit vector simply divide it by its magnitude.

4 Vertex Normals

A vertex normal is a combination of the surface normals of the facets adjacent to the vertex.

To find a vertex normal, one can find the average of the surrounding facet normals. Often a weighted average is used — weights are areas of facets. Normals of larger facets contribute more to the vertex normal. If we use the unnormalised cross products of facet edges, we get this weighting for free. This method is fine for most purposes but there are more complex methods justified by differential geometry.