Assignment 6: Rendering

15-462 Graphics I
Spring 2002
Frank Pfenning

Sample Solution

1 Intersections (30 pts)

1. Assume you are given two line segments in parametric form:

$$\begin{aligned}\mathbf{p}(\alpha) &= (1-\alpha)\mathbf{p}_1 + \alpha\mathbf{p}_2,\\ \mathbf{q}(\beta) &= (1-\beta)\mathbf{q}_1 + \beta\mathbf{q}_2\end{aligned}$$

   Find an algorithm for determining if the two lines intersect.

   We want to solve $\mathbf{p}(\alpha) = \mathbf{q}(\beta)$. *We set them equal and regroup:*

$$\begin{aligned}(1-\alpha)\mathbf{p}_1 + \alpha\mathbf{p}_2 &= (1-\beta)\mathbf{q}_1 + \beta\mathbf{q}_2\\ \mathbf{p}_1 - \alpha\mathbf{p}_1 + \alpha\mathbf{p}_2 &= (1-\beta)\mathbf{q}_1 + \beta\mathbf{q}_2\\ \alpha(-\mathbf{p}_1 + \mathbf{p}_2) + \mathbf{p}_1 &= \beta(-\mathbf{q}_1 + \mathbf{q}_2) - \mathbf{q}_1\end{aligned}$$

   We now expand this into 3 equations, one each for x, y, and z coordinate. This yields 3 equations with 2 unknowns (α and β) which can be solved with standard methods.

   If there are solutions for α and β we check if both of them lie between 0 *and* 1. *If so, the line segments intersect, otherwise not.*

   There is a special case if both line segments lie on the same line. In that case, for every α there is a corresponding β solving our equations. Then the two line segements intersect if any of the 2 endpoints of one line segment lie inside the other line segment.

2. Extend the algorithm from part (1) to determine if two flat, simple polygons intersect.

   We consider three cases.

   Case 1: The planes of the polygon are parallel.
   If the normals of the two polygons are the same, or oppositely signed - the polygons are coplanar or parallel. Find the Ax+By+Cz+D= 0 equation for both polygons. If the D value is different, the planes are parallel; the polygons do not intersect.

   Case 2: The polygons are coplanar.
   Test each line of one polygon against every line of the other polygon. If they intersect, there is an intersection. Test for one polygon inside the other by the odd-even rule.

Case 3: The polygons are not coplanar.
Find the intersection of the two planes; this will give a line. Check the line for intersection of every line in each polygon; if there is no intersection for both polygons, the polygons do not intersect. If the lines intersect both polygons, then use the odd-even rule to determine whether one polygon intersects the other.

3. What is the worst-case complexity of your algorithm?

   Worst case behavior is case 2, which is $O(mn)$, where m and n are the number of lines in both polygons, respectively.

4. Does your algorithm work correctly for flat, non-simple polygons under the odd-even rule to determine their interior? Explain why, or discuss possible adaptations of your method.

   Yes, it already uses the odd-even rule.

5. Suggest at least two efficiency improvements of your algorithm from part (2) and analyze whether they improve the worst-case complexity or just the practical behavior in common cases.

   Cover the polygons in bounding boxes. This does not reduce worst-case complexity. [Second optimization omitted—use your imagination.]

2 Compositing (20 pts)

In programming assignment 3 you were asked to generate a shadow by redrawing an object under a projection transformation (see also [Angel, Ch 5.9]).

1. Explain in detail how you might use blending to merely darken the shadow area instead of simply overwriting it with the shadow color. Give the critical calls to OpenGL and sketch where they would employed.

   In order to darken the areas of the shadow, it is necessary to blend the color of the shadow with the color of the object behind it. This can be done by setting a blending functions and adding an alpha value to the colors.

```
glEnable(GL_BLEND);

glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

DrawMobile();

glDisable(GL_BLEND);

DrawMobile() {
//Draw mobile normally
//set alpha value to 0.5 so half of color comes from background
//and half from the shadow color;
```

```
glColor4f(0.0, 0.0, 0.0, 0.5);

//Draw projected mobile
}
```

2. Describe two different methods for obtaining soft shadows. Which one would you prefer for the mobile animation and why?

One way to do soft shadows is by ray-tracing a scene and assume the light source has an area so that a pixel may hit the light source multiple times.

Another way to do a soft shadow is to jitter the object multiple times and draw alpha blended shadows. This will cause multiple blended shadows to overlap giving darker areas and around the borders where fewer shadows overlap it will make a lighter shadow.

The second way is preferable for the mobile because in OpenGL you cannot assign an area to a light.