OpenGL Transparency

Oregon State University

Mike Bailey

mjb@cs.oregonstate.edu

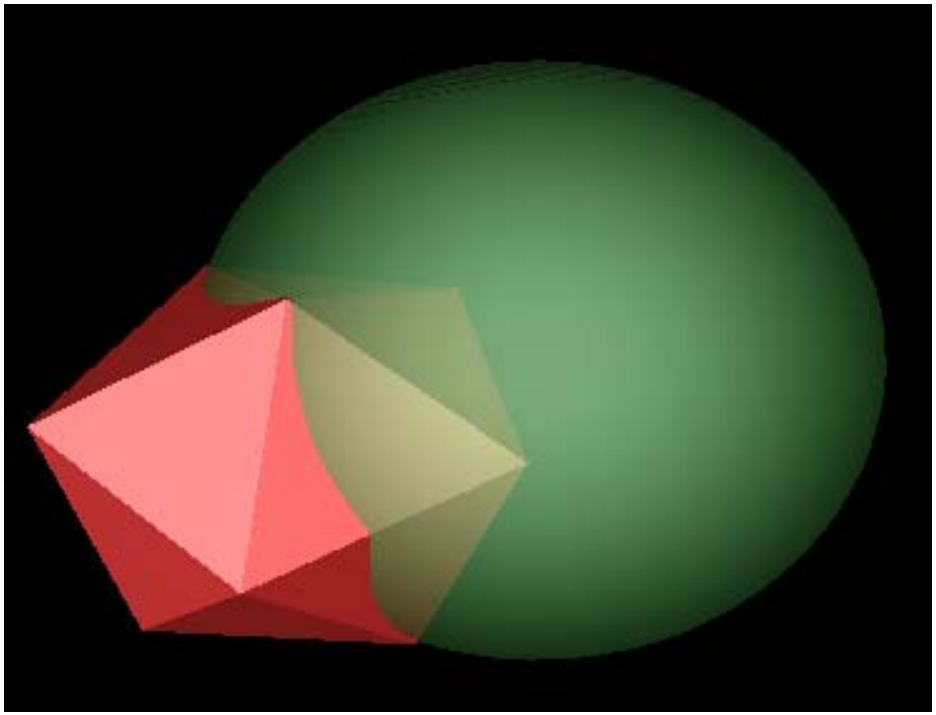
Oregon State University
Computer Graphics

OpenGL Transparency

OpenGL has a nice feature that lets you display see-through objects. This is useful in visualization when wanting to see some objects through other objects.

OpenGL calls it **"transparency",** but in fact it is really **"blending"**. Be sure to remember this. Real transparency is a subtractive-color process. For example, looking at a pure red object through a pure green piece of glass will give you **black**.

OpenGL "transparency" would instead blend the RGB of the red object with the RGB of the green glass, giving a shade of yellow.



When Defining Your Object

Instead of using **glColor3f( )** to specify the red, green, and blue of an object, use **glColor4f( )** to specify red, green, blue, and *alpha*. Alpha is the transparency factor.

Or, if you are using lighting,

glMaterialfv(GL_FRONT, GL_AMBIENT, rgba);
glMaterialfv(GL_FRONT, GL_DIFFUSE, rgba);

- An alpha value of **0.0** means that this object is **completely transparent** (i.e., invisible – not too useful).
- An alpha value of **1.0** means that this object is completely **opaque** (also not useful as a transparency).

$$C' = \alpha C_{new} + (1. - \alpha) C_{old}$$

In the Display( ) Callback

1. Draw the solid things first

2. Enable color blending:

```
glEnable( GL_BLEND );
```

3. Make the Z-buffer read-only:

```
glDepthMask( GL_FALSE );
```

This is important because you don't want the presence of a transparent object close to your eye to prevent the writing of its blend with an object a little farther away.

In the Display( ) Callback

4. Define how much of the about-to-be-written pixel color (the “source”) and how much of the already-existing pixel color (the “destination”) will end up being used:

glBlendFunc(src, dst);

The value for src multiplies the about-to-be-written source pixel color (S) and the value of dst multiplies the already-existing destination pixel color (D). While there are several options for src and dst, the most useful combination is:

Src (C_{new})	Dst (C_{old})	Result (C’)
GL_SRC_ALPHA	GL_ONE_MINUS_SRC_ALPHA	C’ = αS + (1-α)D

5. Draw the transparent things.

6. After drawing all the transparent elements of the scene, set the depth mask back to read-write and disable blending:

glDepthMask(GL_TRUE);
glDisable(GL_BLEND);

Tips

- Remember that the way OpenGL implements this is not really transparency, it is *blending*. True transparency is a color-subtractive process. Blending is a color-additive process. If the goal is to make a yellow window look and behave like a real yellow window, this won't work correctly. If the goal is to just see inside something, this works very well.

- Recognize that OpenGL picking will know nothing about your transparency. As far as it is concerned, you drew all solid, opaque polygons.

- An example from the world of medical visualization: