# OpenGL "Transparency" Made Simple

OpenGL has a nice capability that lets you display see-through objects. This is useful in visualization when wanting to see some objects through other objects. It is called "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.
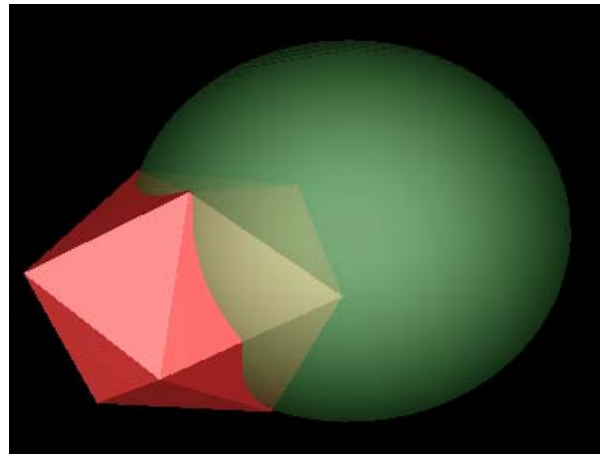
Here's how to do OpenGL blending:

<div style="border:2px solid black">

When Defining Objects

</div>

1. 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.

    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).

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

<div style="border:2px solid black">

In `Display():`

</div>

2. Draw the solid things first.

3. Enable color blending:

    ```
    glEnable( GL_BLEND );
    ```

4. 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.

5. 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	$C' = \alpha S + (1-\alpha)D$

6. Draw the transparent things.

7. 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: