Will It Blend?

Yes! It blends!

—Tom Dickson, owner of the Blendtec blender company

In 2006, Tom Dickson posted a goofy video to YouTube illustrating how tough his company’s blenders were by blending some marbles into powder. Since then his frequent videos have been viewed more than 100 million times and have featured blendings of everything from a Tiki torch and a laser pointer to a Justin Bieber doll and a new camcorder. Tom’s kind of blending has nothing to do with our kind of blending, though, unless the sadistic and unmerciful pulverization of a couple of Android touchpad and phones count. After all, they are OpenGL ES devices: devices that have their own form of blending, albeit not nearly as destructive. (Yes, it’s a stretch.)

Blending plays an important role in OpenGL ES applications. It is the process used to create translucent objects that can be used for something as simple as a window to something as complicated as a pond. Other uses include the addition of atmospherics such as fog or smoke, the smoothing out of aliased lines, and the simulation of various sophisticated lighting effects. OpenGL ES 2.0 has a complex mechanism that uses small modules called shaders to do specialized blending effects, among other things. But before shaders, there were blending functions, which are not nearly as versatile but considerably easier to use.

In this chapter, you’ll learn the basics of blending functions and how to apply them for both color and alpha blending. After that, you’ll use a different kind of blending involving multiple textures, used for far more sophisticated effects such as shadowing. Finally, I’ll figure out how we can apply these effects in the solar-system project.
Alpha Blending

You have no doubt noticed the color quadruplet of “RGBA.” As mentioned earlier, the A part is the alpha channel, and it is traditionally used for specifying translucency in an image. In a bitmap used for texturing, the alpha layer forms an 8-bit image of sorts, which can be translucent in one section, transparent in another, and completely opaque in a third. If an object isn’t using texturing but instead has its color specified via its vertices, lighting, or overall global coloring, alpha will let the entire object or scene have translucent properties. A value of 1.0 means the object or pixel is completely opaque, while 0 means it is completely invisible.

For alpha to work as with any blending model, you work with both a source and a destination image. Because this topic is best understood through examples, we’re going to start with the first one now.

Grab your Chapter 1 exercise, and then use Listing 6–1 in place of the original methods. Solid squares of colors are used here first instead of textured ones, because it makes for a simpler example.

**Listing 6–1. The New and Improved **onDrawFrame()** Method

```java
public void onDrawFrame(GL10 gl) {
    gl.glClearColor(0.0f, 0.0f, 0.0f, 1.0f);        //1
    gl.glClear(GL11.GL_COLOR_BUFFER_BIT | GL11.GL_DEPTH_BUFFER_BIT);

    gl.glMatrixMode(GL11.GL_MODELVIEW);
    gl.glEnableClientState(GL11.GL_VERTEX_ARRAY);

    //SQUARE 1
    gl.glLoadIdentity();
    gl.glTranslatef(0.0f, (float)Math.sin(mTransY), -3.0f);   //2
    gl.glColor4f(0.0f, 0.0f, 1.0f, 1.0f);
    mSquare.draw(gl);

    //SQUARE 2
    gl.glLoadIdentity();       //3
    gl.glTranslatef((float)(Math.sin(mTransY)/2.0f), 0.0f, -2.9f);
    gl.glColor4f(1.0f, 0.0f, 0.0f, 1.0f);
    mSquare.draw(gl);

    mTransY += .075f;
}
```