Chapter objectives:

- Understand OpenGL blending to achieve transparency, antialiasing, and fog
- Use images for rendering directly or for texture mapping
- Understand OpenGL texture mapping programs

4.1 Blending

Given two color components $I_{\lambda,1}$ and $I_{\lambda,2}$, the blending of the two values is an interpolation between the two:

$$I_{\lambda} = \alpha I_{\lambda,1} + (1 - \alpha)I_{\lambda,2} \tag{EQ 97}$$

where $\alpha$ is called the alpha blending factor, and $\lambda$ is R, G, B, or A. Transparency is achieved by blending. Given two transparent polygons, every pixel color is a blending of the corresponding points on the two polygons along the projection line.

In OpenGL, without blending, each pixel will overwrite the corresponding value in the frame buffer during scan-conversion. In contrast, when blending is enabled, the current pixel color component (namely the source $I_{\lambda,1}$) is blended with the corresponding pixel color component already in the frame buffer (namely the destination $I_{\lambda,2}$). The blending function is an extension of Equation 97:

$$I_{\lambda} = B_1 I_{\lambda,1} + B_2 I_{\lambda,2} \tag{EQ 98}$$
where $B_1$ and $B_2$ are the source and destination blending factors, respectively.

The blending factors are decided by the function: $\text{glBlendFunc}(B_1, B_2)$ where $B_1$ and $B_2$ are predefined constants to indicate how to compute $B_1$ and $B_2$, respectively. As in Example 4.1 (Fig. 4.1), $B_1 = \text{GL\_SRC\_ALPHA}$ indicates that the source blending factor is the source color’s alpha value, which is the $A$ in the source pixel’s RGBA. That is, $B_1 = A$, and $B_2 = \text{GL\_ONE\_MINUS\_SRC\_ALPHA}$ indicates that $B_2 = 1-A$. When we specify a color directly, or specify a material property in lighting, we now specify and use the alpha value as well. In Example 4.1, when we specify the material properties, we choose $A=0.3$ to represent the material’s transparency property. Here, if we choose $A=0.0$, the material is completely transparent. If $A=1.0$, the material is opaque.

/* Example 4.1.blending.c: transparent spheres */

```c
void drawSolar(float E, float e, float M, float m)
{
    float red[] = {1., 0., 0., .3};

    drawSphere(); // opaque sphere
    glEnable (GL_BLEND);
    glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    glMaterialfv(GL_FRONT, GL_AMBIENT, red);
    drawSphere(); // transparent sphere
    glDisable (GL_BLEND);
}
```

4.1.1 OpenGL Blending Factors

Example 4.1 chooses the alpha blending factor as in Equation 97, which is a special case. OpenGL provides more constants to indicate how to compute the source or destination blending factors through $\text{glBlendFunc}()$. 