Material Expressions into the Custom Lighting channel, you can create your own simulation of how light should interact with the surface of your material. For this channel to be used, you must set the lighting model to MLM_Custom (see **FIGURE 6.21**).

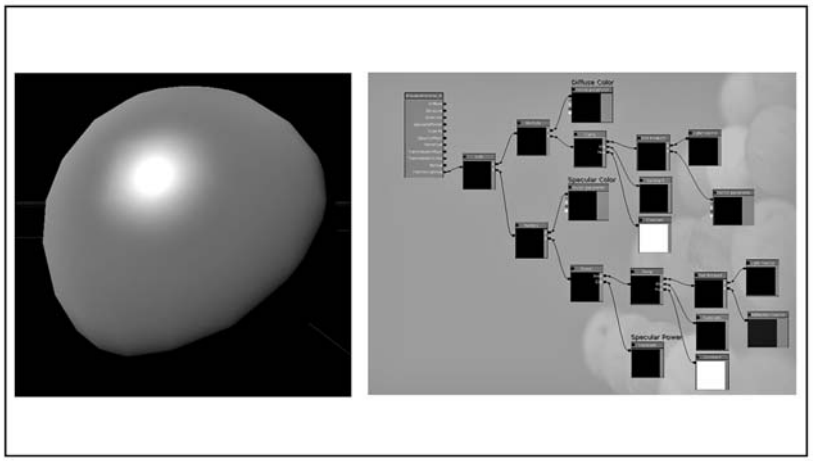

FIGURE 6.21 This example uses the intricate network on the right to approximate the Phong lighting model.

Now that you basically know what channels are available to you, you need to know what you can do with them. This is where Material Expressions come into play. In the next section, we look at Material Expressions and how they are used in your materials.

Material Expressions

Material Expressions are the fundamental building blocks used to create materials. Each expression contains a specific functionality such as providing an interface to a texture, adding two values or vectors, or modifying texture coordinates. By combining these expressions into elaborate networks, many interesting effects can be created, resulting in much more striking visuals for your level. Before we take a close look at each of the available Material Expressions in the Material Editor, we're going to add a few of the simpler expressions into the basic material we created earlier. In **TUTORIAL 6.2**, we demonstrate how to add and connect these material expressions nodes to change its look in some way.

Clamp

A Clamp expression takes in three inputs: Input, Min, and Max. The expression then limits (or "clamps") the value of the first input, Input, so that is always between the two values of the Min and Max inputs (see **FIGURE 6.34**).

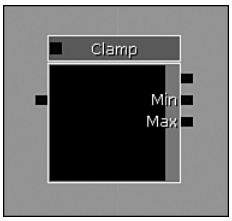

FIGURE 6.34 Clamp expression icon

ComponentMask

A ComponentMask expression takes in one value which can be a vector of any size (see **FIGURE 6.35**). This expression enables you to pick which components of a vector are allowed to pass through it. For example, you could input a Texture Sample, and using the properties of the ComponentMask, output only the R and A values of that texture. The information would then be treated just like a Constant2Vector, in that it would only have two components. This means it could be added, subtracted, multiplied, or divided by a Constant2Vector expression, or any other vector value containing only two components.

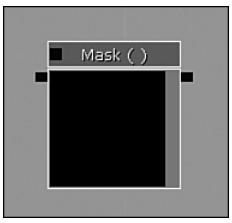

FIGURE 6.35 ComponentMask expression icon

Constant Expressions

The Constant expressions are, in essence, placeholders for static values in your material. These expressions have no inputs—only properties that hold user-inputted values. These values are then outputted to various parts of the expression. There are four different types of Constant expressions, differentiated by the number of components each expression contains, ranging from one component (R), to four components (R, G, B, and A).

Constant

This is the simplest Constant expression, as it can contain only a single value, which is read as R inside the Properties window (see **FIGURE 6.36**).

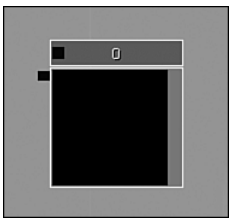

FIGURE 6.36 Constant expression icon

Time

The Time expression is used to add the passage of time to another expression that is time dependent, such as a Panner, Rotator, Sine, or Cosine, allowing such changes to take place constantly during gameplay. The bIgnorePause property allows time progression to continue even when the game is paused (see **FIGURE 6.83**).

NOTE

You don't necessarily need a time node for the Panner and Rotator expressions, but if you have a custom Time that has been affected by other expressions, you can plug it into a Panner or Rotator.

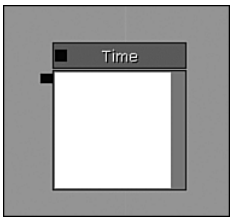

FIGURE 6.83 Time expression icon

Transform

The Transform expression enables you to transform any three-component vector in different coordinate systems, such as world space, view space, and local space (see **FIGURE 6.84**). The expression takes in one input, which must be a three-component vector, which is then transformed from tangent space into the space specified in the TransformType parameter. The options are TRANSFORM_World, TRANSFORM_View, and TRANSFORM_Local. This expression can be useful sampling cubemaps, as it enables you to transform a normal map to world space, rather than tangent space, which exists only across a surface. More information on cubemaps is presented later in this chapter.

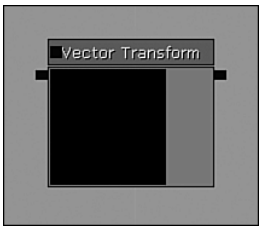

FIGURE 6.84 Transform expression icon

TransformPosition

This expression is deprecated and should not be used.

VectorParameter

See the Parameter expression descriptions at the end of this section.

VertexColor

A VertexColor expression functions like a MeshEmitterVertex Color, discussed previously. However, this expression is intended for use on sprite emitters instead of mesh emitters. The VertexColor expression outputs the vertex color for the current fragment being operated on (see **FIGURE 6.85**). This node also allows access to the vertex colors of a static mesh as exported out of a traditional 3d modeling package, for use in your material.

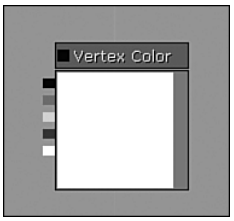

FIGURE 6.85 VertexColor expression icon