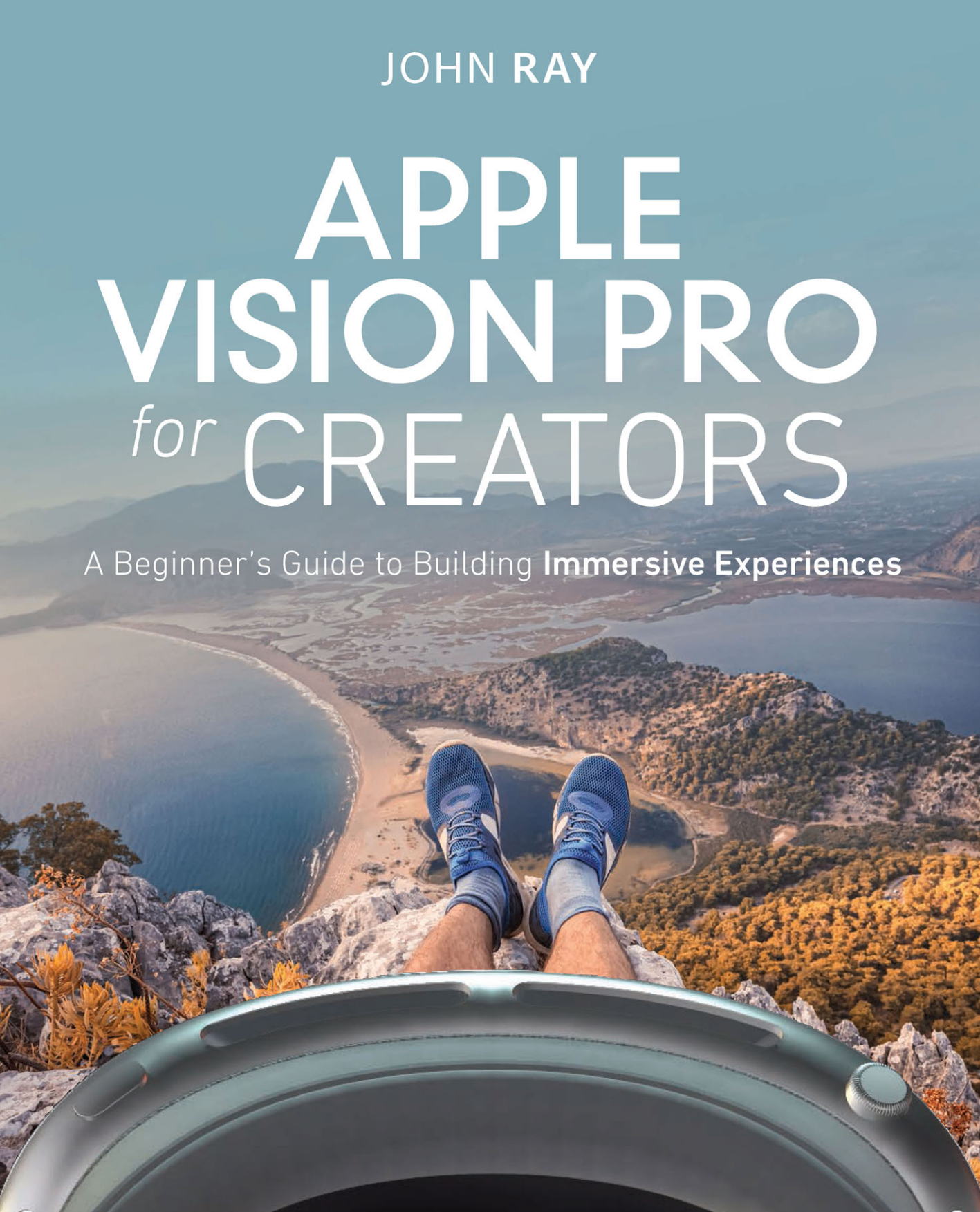
JOHN **RAY**

APPLE VISION PRO *for* CREATORS

A Beginner's Guide to Building **Immersive Experiences**



JOHN RAY

APPLE VISION PRO *for* CREATORS

A Beginner's Guide to Building **Immersive Experiences**

New Riders VOICES THAT MATTER™

connecting an "image" node that references a wood grain texture image to the surface color attribute of a material node. I give you an example shortly.

If this all sounds like technobabble to you, just know that you're going to be creating materials (surface shaders) by connecting different components that help describe the surface (a node graph). Making it all work behind the scenes is MaterialX.

I SEE THERE ARE SOME NICE HIGH-QUALITY MATERIALS THAT I CAN DOWNLOAD FOR FREE. CAN I USE THE TEXTURES I DOWNLOAD?

Open-source formats frequently result in a wealth of public content being made available, and MaterialX is no different. Unfortunately, in its current state there is no way to import MaterialX (.mtlx) materials into Reality Composer Pro. Apple supports importing materials that are part of USD files, but standalone textures are a no-show.

That being said, MaterialX *is* an open-source and text-based XML format. If you download a MaterialX material, it comes as a zipped folder structure. Within the folder are any resources required to build the material, along with a .mtlx file that describes each node and value that goes into the shader. After you've had a bit of experience creating your own custom shaders, you'll find it pretty easy to recreate the shader just by reading the text file and configuring the building blocks (nodes) within Reality Composer Pro.

I've included a car paint material ("Car_Paint_2k_16b") that you can use as a practice project at the end of this chapter. You can find other examples at the free online library here: https://matlib.gpuopen.com/main/materials/all

Hopefully Apple adds support for directly importing these materials soon!

Node Graphs

It isn't very easy to describe a node graph while showing you how to build one, so let's start with a graph that already exists, and I'll talk you through how it was created. Within the Chapter 4 download, open the Wooden Material folder and then double-click the Package.realitycomposerpro file to launch Reality Composer Pro.

When the workspace opens, you probably won't see much of anything except a wooden sphere, a bit like **FIGURE 4.15**.

By default, Scene.usda *should* be open, if not, switch to the Project Browser to open it (see Chapter 3, "Getting Started with Reality Composer Pro," for more details on working in the Project Browser). Make sure that the scene hierarchy is expanded so you can see Sphere and Wood Grain.

Now click Wood_Grain in the scene hierarchy and click the Shader Graph tab directly above the editor. You are now looking at the material definition for the wood appearance of the sphere, as shown in **FIGURE 4.16**.

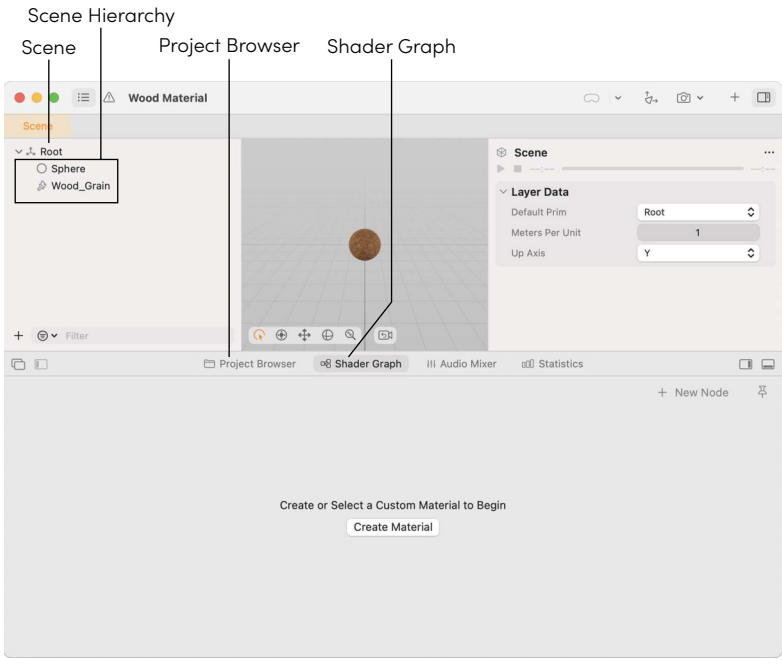


FIGURE 4.15 Open the project and expand the scene hierarchy.

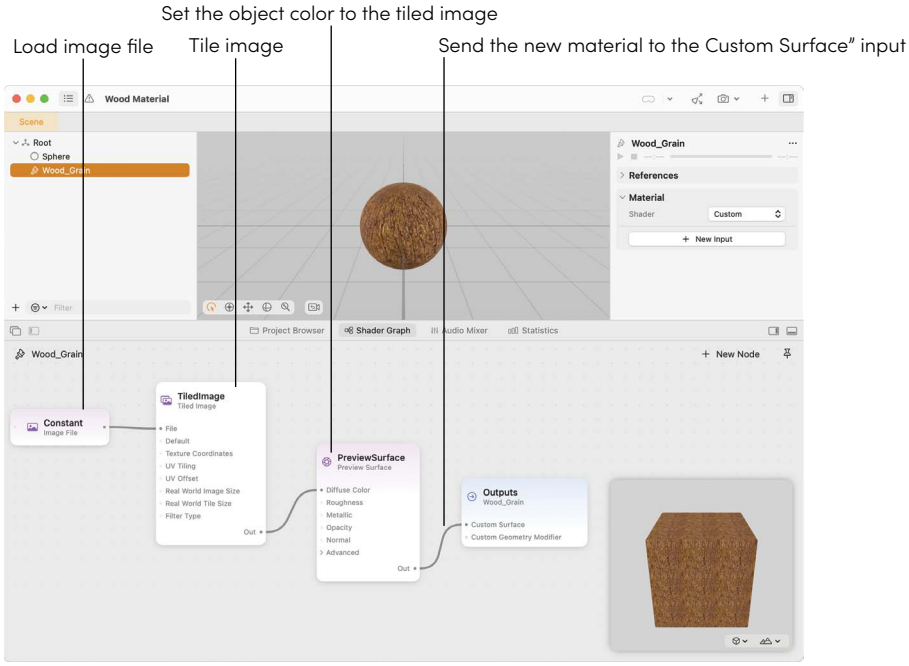


FIGURE 4.16 The scene contains a woodgrain sphere. Zoom in if you don't believe me.

Dissecting the Shader Graph

There are four nodes that make up the node graph (also known as the custom surface shader), and each performs a simple function along the way.

On the far left, the initial node opens an image file. What image file? Let's find out! Select the Constant Image File node and look at the Inputs panel in the inspector. Any values that can be configured manually are displayed here. As shown in **FIGURE 4.17**, the image file is set to woodgrain.jpg, an image file I've already added to the project. Clicking to the right of the file-name in the inspector allows me to pick from any image in the project.

NOTE To be clear, you can pick only images you've added via the Project Browser in Reality Composer Pro, not images located elsewhere on your Mac.

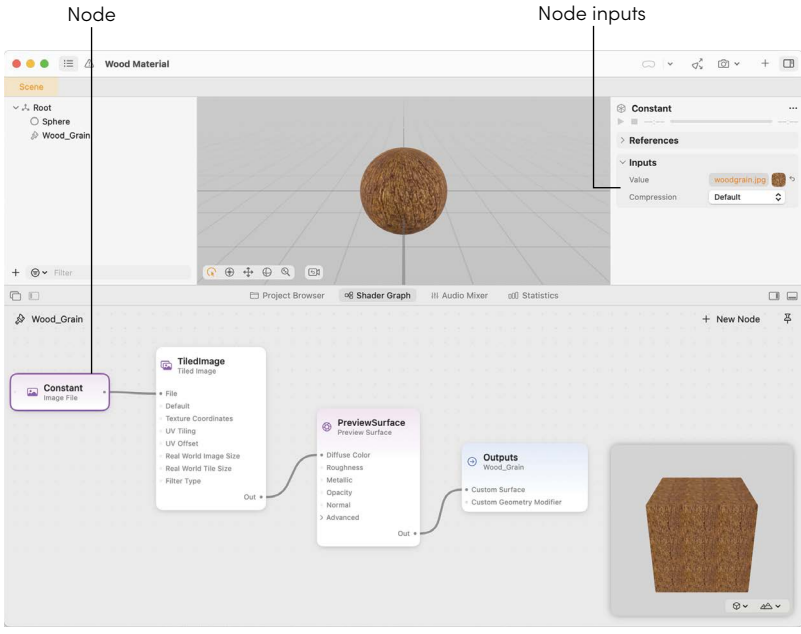


FIGURE 4.17 Use the inspector to set any node inputs you want to enter manually.

In the shader graph, I've added a Tiled Image node. This node can take an image and tile it so that a single image isn't stretched over huge objects. If you remember when you used to have to apply repeating desktop patterns on your Mac versus having a giant background, this is the same concept.

I create a tiled image from the single woodgrain image—three tiles horizontally and three vertically. I do this by selecting the Tiled Image node and using the inspector to edit the inputs beside UV Tiling; I enter 3 in the first field (horizontal) and 3 in the second field (vertical), as shown in **FIGURE 4.18**.

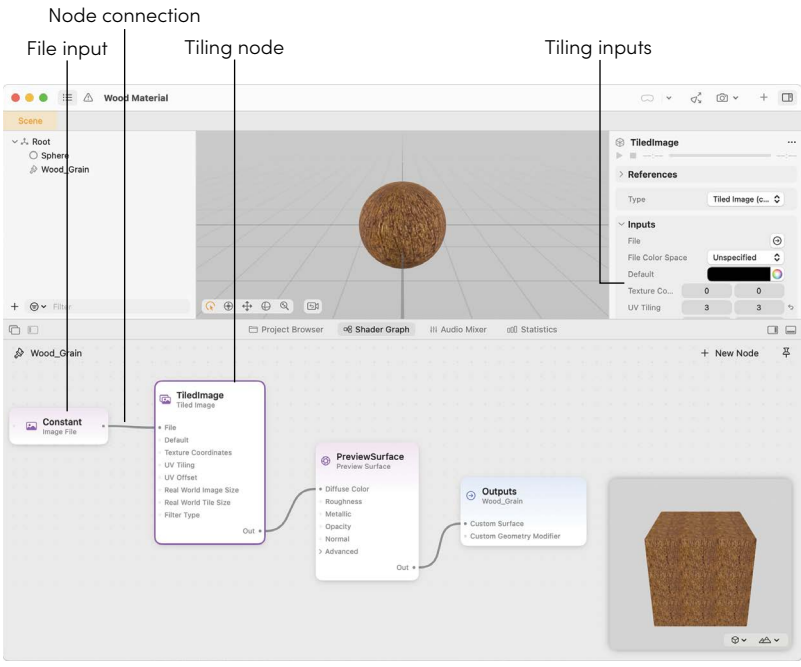


FIGURE 4.18 Tiling the image gives more texture to work with.

Initially, the image loaded in the first node isn't available to the tile node, so there's nothing for it to tile. To make the image file available, I drag from its output dot to the file input dot on the second image. This is visible in Figure 4.18 and is shown in a close-up in **FIGURE 4.19**.

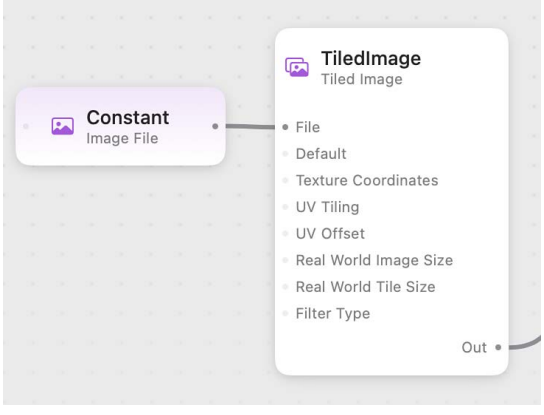


FIGURE 4.19 The output of the image node feeds the input of the tile node.

Now, I'm left with two nodes: Preview Surface and Outputs. These nodes are added automatically to shaders. The first, Preview Surface, is a configurable physical surface node identical

to the physical surface attributes that you used last chapter. Using the inputs of the Preview Surface node, you can manually configure any of the physical surface properties you'd like. You can feed others, such as the Diffuse Color (the base color), via the output of one of your nodes. For the tiled woodgrain texture, I choose to use the output of the Tiled Image to feed the Diffuse Color input, as shown in **FIGURE 4.20**. I've also set the Roughness input of the Preview Surface node to 1.0. This wood isn't intended to be shiny.

Finally, the Preview Surface node's output contains the finished surface. To make it apply to an object, it feeds the Custom Surface input of the Output node. This connection is added by default; breaking it causes your objects to show a dismal gray color as the surface.

That's it! That's all there is to this shader. It's now available to use on any object I add to my Reality Composer Pro project.

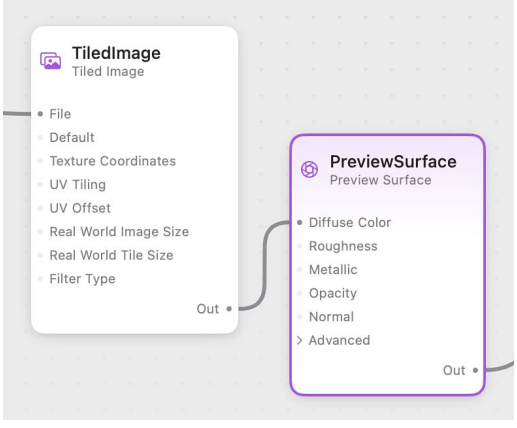


FIGURE 4.20 The Tiled Image node subsequently outputs an image that provides the Diffuse Color input for the Preview Surface node.

COMPLEXITY FOR LEARNING'S SAKE

I'm going to come clean. This surface shader is largely unnecessary and was just a means of providing an easy-to-understand example. If you were reading closely, you might have noticed that the node graph takes an image file and feeds it to the File input of Tile Image. Rather than bother with the node that loads the image, it could just be set in the inspector when viewing the Tile Image node, resulting in a shader graph with only one real node.

Furthermore, if I wasn't concerned about the tiling, I could have just set the image file directly as the diffuse color of a physically based material and never bothered with a graph at all.

Surface Shaders Versus Geometry Modifiers

The graph that we've just reviewed is an example of a surface shader. In looking at the nodes, you may have noticed that the Outputs node has another input: Custom Geometry Modifier.

As previously mentioned, a surface shader changes the outside appearance of an object. The shader graph is run once for every single pixel that is displayed for the object *per frame*. A geometry modifier (or geometry shader) is built in an identical fashion but alters the geometry of the object. In other words, a geometry modifier can change the attributes that define the shape, location, and size of an object.

Geometry modifiers work by altering the vertices of a model, once per frame. A **vertex** is a corner or edge where two faces of a 3D object come together. If you increase the X value of all the vertices in an object, the object moves to the right. Add to the Y value, and the object moves up, and so on.

Not all objects have vertices. Spheres, for example, do not have any edges that come together. However, you can still apply geometry modifiers to these objects, and MaterialX makes sure they do what you want.

TIP For a vertex-less sphere, you can imagine that the vertices are three lines that run perpendicularly to one another: side to side, front to back, and top to bottom. The same can be assumed for a cylinder—another shape with no vertices. In other words, just because a shape doesn't technically have vertices doesn't mean it can't be modified. These special cases will be taken care of for you.

Later in this chapter, you create a geometry modifier and see how a material has more power than just slapping a coat of paint on a shape.

CREATING A CUSTOM SHADER GRAPH

Now that you've seen what to expect from a node graph (and a shader graph, specifically), it's time to find out how to build one from scratch. To do this, start with a blank slate in Reality Composer Pro. Name the new project **Mottled Colors**. You're going to create a surface that looks a bit like an abstract watercolor painting or a tie-dye shirt. Although you could create a new material without an object, it's easier to work with a basic scene so that you can get a good look at your creation.

To that end, add a sphere to the default scene in the new project. Within the scene hierarchy, expand the Sphere object. Your scene should resemble **FIGURE 4.21**.

Now, select DefaultMaterial and press Delete. DefaultMaterial is the white/grayish default material that is attached to the sphere when you add it, and you don't need it. The sphere takes on a swirly candy-cane appearance (see **FIGURE 4.22**) to show it doesn't have a material attached.

You now have a scene with a sphere and zero materials. Time to add one!