Chapter 12. Creating the Materials in Cycles

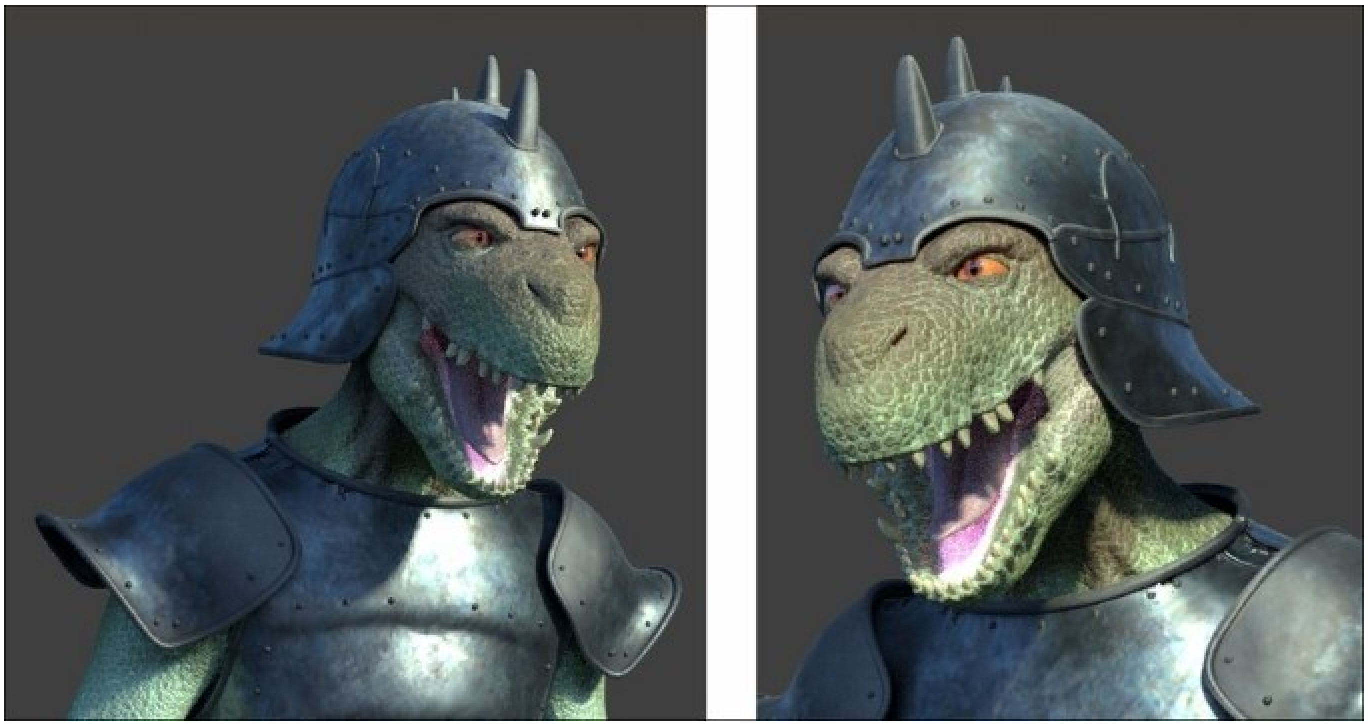
In this chapter, we will cover the following recipes:

- Building the reptile skin shaders in Cycles
- Making a node group of the skin shader to reuse it
- Building the eyes' shaders in Cycles
- Building the armor shaders in Cycles

Introduction

In Chapter 10, *Creating the Textures*, and in Chapter 11, *Refining the Textures*, we have prepared all the necessary texture images for the **Gidiosaurus** skin and for the iron **Armor** (the creation process for some textures, specifically the two textures for the character's **eyes**, hasn't been described, but basically it's a process similar to what we have already seen).

In this chapter, we'll see how to use these textures and how to set up the materials for the **Gidiosaurus** and the **Armor** in the **Cycles Render** engine.



A rendered example of the Cycles' shader final result

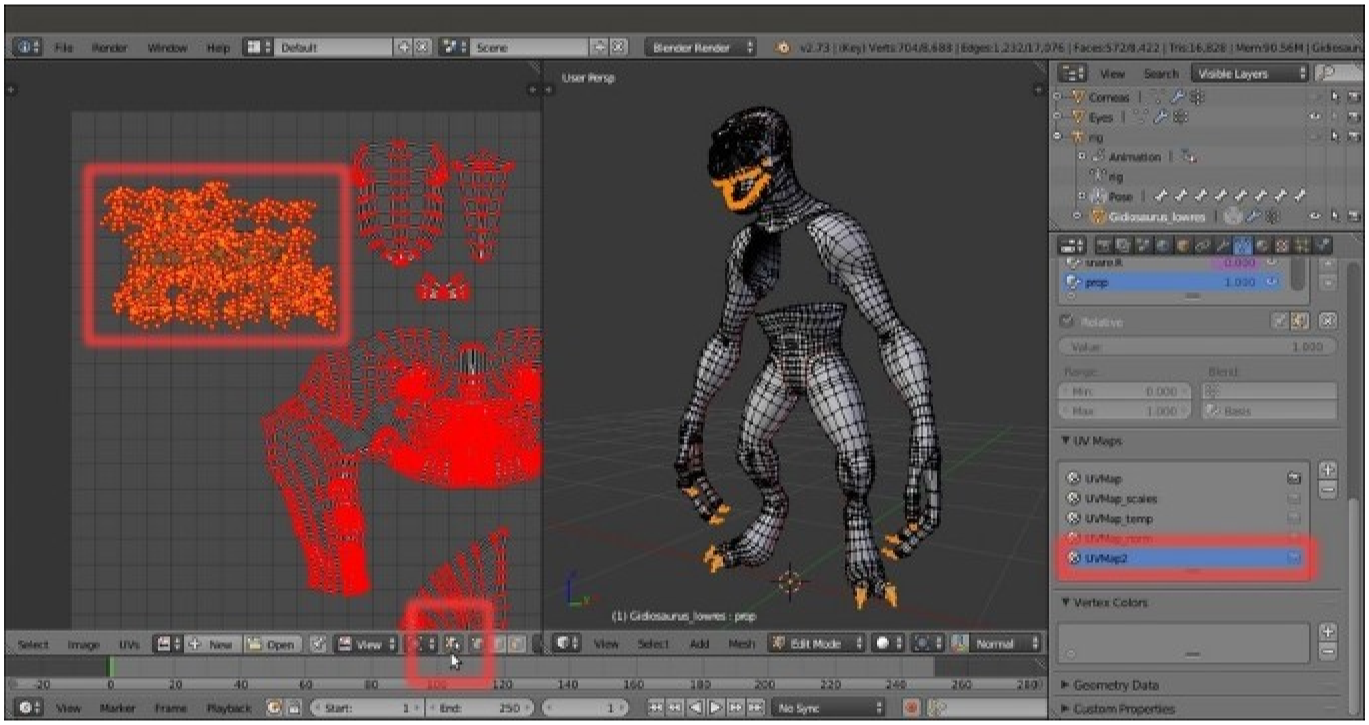
Building the reptile skin shaders in Cycles

So, let's start with the **Gidiosaurus** skin.

Getting ready

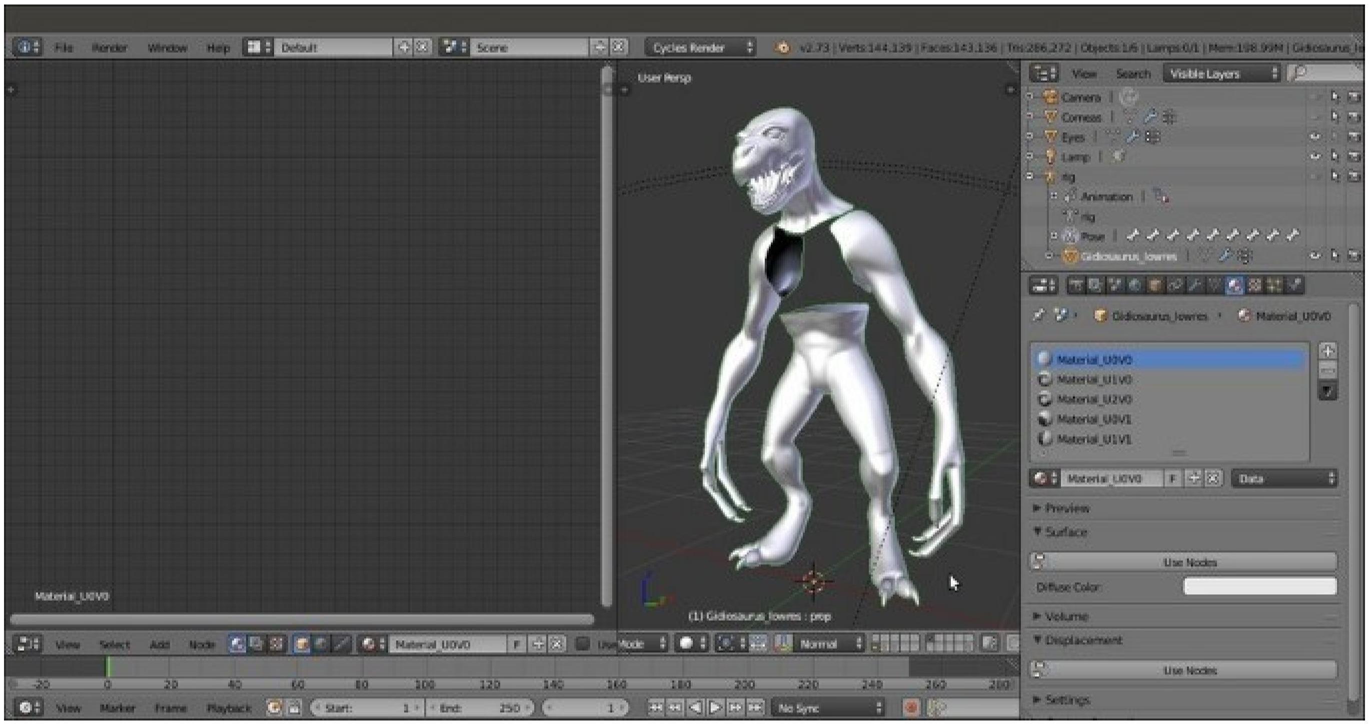
But first, as usual, we must prepare the file:

1. As the very first step, go to the `texture_making` folder and move the textures `vcol.png`, `vcol2.png`, `norm.png`, and `norm2.png` to the `textures` folder.
2. Then start Blender and open the `Gidiosaurus_baking_normals.blend` file we saved in Chapter 11, *Refining the Textures*.
3. Switch the left **UV/Image Editor** window with a **Node Editor** window and press the *N* key to get rid of the **Properties** sidebar. Put the mouse pointer in the 3D viewport to the right and press the *T* key to get rid of the **Tool Shelf** panel, then press the *Z* key twice to go in **Solid** viewport shading mode.
4. Enable the **3rd** scene layer, select and delete the **Armor_detailing** object (press the *X* key, then left-click to confirm).
5. Enable the **4th** scene layer and select and delete the **Gidiosaurus_for_baking** object as well. Enable the **14th** scene layer, and also select and delete the **Gidiosaurus_detailing** and the **enamels** objects.
6. Enable the **11th** scene layer and right-click on the **Gidiosaurus_lowres** object to select it.
7. It's not mandatory but, in case it is not already disabled, it is best to go to the **Object Modifiers** window and disable the **Armature** modifier visibility in the viewport by clicking on the eye icon button.
8. Go to the **UV Maps** subpanel under the **Object Data** window and select the **UVMap** coordinates layer (the first one); press *Tab* to enter **Edit Mode**, then click on the **+** icon button to the right side of the **UV Maps** subpanel to add a new coordinates layer; rename it **UVMap2**.
9. Go to the left window and change it into a **UV/Image Editor**; one by one, select the UV islands of the **talons** (at the moment these are placed inside the other **UDIM** tile spaces), and move them to the default **U0V0** tile, to overlap the location of the **teeth** islands. The reason for this will be clear later, when we will reuse the same color map both for the teeth and for all the talons.
10. If necessary, remember to disable the *Keep UV and edit mode mesh selection in sync* button on the **UV/Image Editor** toolbar:



Moving the talon islands to overlap the teeth islands inside the default U0V0 tile space

11. Go out of **Edit Mode** and switch the **UV/Image Editor** window back to a **Node Editor** window.
12. Click on the *Engine to use for rendering* slot on the main top header to switch to the **Cycles Render** engine.
13. Also, enable the **6th** scene layer to show the **Lamps**. Go to the **Outliner**, unhide and delete the **Lamp.001** object and select the **Lamp** object; in the **Object Data** window, change the type to **Spot** and then click on the **Use Nodes** button. Set the **Strength** to **10.000** and the **Color** to **R 1.000**, **G 1.000**, **B 0.650**, then set the **Size** to **0.500** and enable the **Multiple Importance** item (the **Multiple Importance Sampling** helps in reducing noise for big lamps and sharp glossy reflections, at the cost of the samples rendering a bit slower).
14. Put the mouse pointer in the 3D viewport and press *N* to call the **Properties** sidepanel; in the top **Transform** subpanel, type: **Location X** = **6.059204**, **Y** = **-9.912249** and **Z** = **7.546275**, and **Rotation X** = **55.788944°**, **Y** = **0°** and **Z** = **30.561825°**.
15. Go to the **Render** window and, under the **Sampling** subpanel, set the **Samples** to **400** for **Render** and **300** for **Preview**.
16. Go down to the **Light Paths** subpanel and disable the **Reflective Caustics** item, then set **Filter Glossy** to **1.000**.
17. Re-select the **Gidiosaurus_lowres** object and go to the **Material** window:

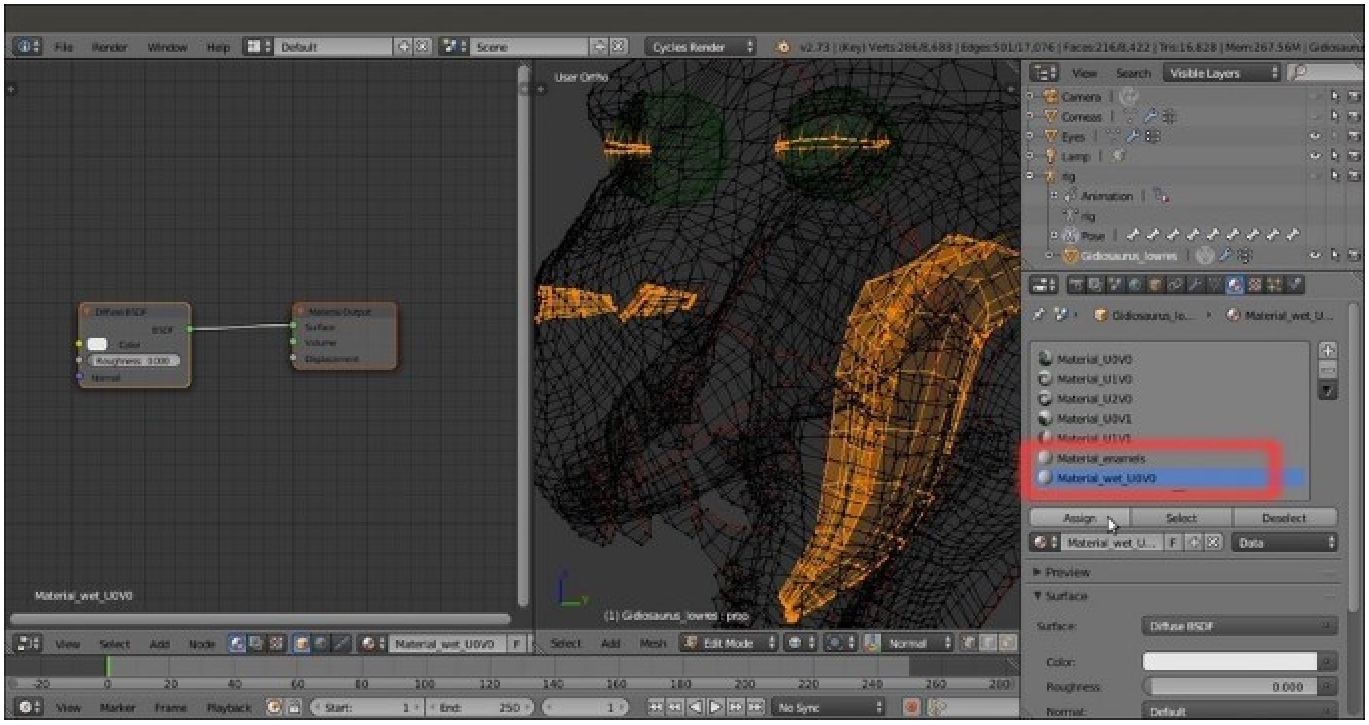


The 5 materials under the Cycles render engine

As you can see in the previous screenshot, the **Gidiosaurus_lowres** object has already assigned the **5** materials corresponding to the **5 UDIM** tile spaces (see Chapter 5, *Unwrapping the Low Resolution Mesh*, and Chapter 10, *Creating the Textures*).

The materials have been created under **Blender Internal** so, switching to **Cycles**, they show but aren't *initialized* as **node materials** yet; besides this, just before starting the creation of the first Cycles material, we must add **two** more materials.

18. Click the + icon button **twice** (*Add a new material slot*) to the right side of the **Material** window to add **two** new material slots.
19. Select the penultimate slot and click on the **New** button; rename the new material as `Material_enamels`. Select the last slot, click on the **New** button and rename it as `Material_wet_U0V0`.
20. Press *Tab* to enter **Edit Mode** and select all the vertices of the **teeth** and the **talons**; assign them to the `Material_enamels` slot.
21. Zoom on the **Gidiosaurus** head; select the vertices of the inner **nostrils**, of the inner edges of the **eyelids** and of the **tongue**, as shown in the following screenshot, and assign them to the `Material_wet_U0V0` slot:



Selecting the vertices of the "wet" areas of the character's head to assign them to the "Material_wet_U0V0" slot

22. Save the file as `Gidiosaurus_shaders_start.blend` file.

How to do it…

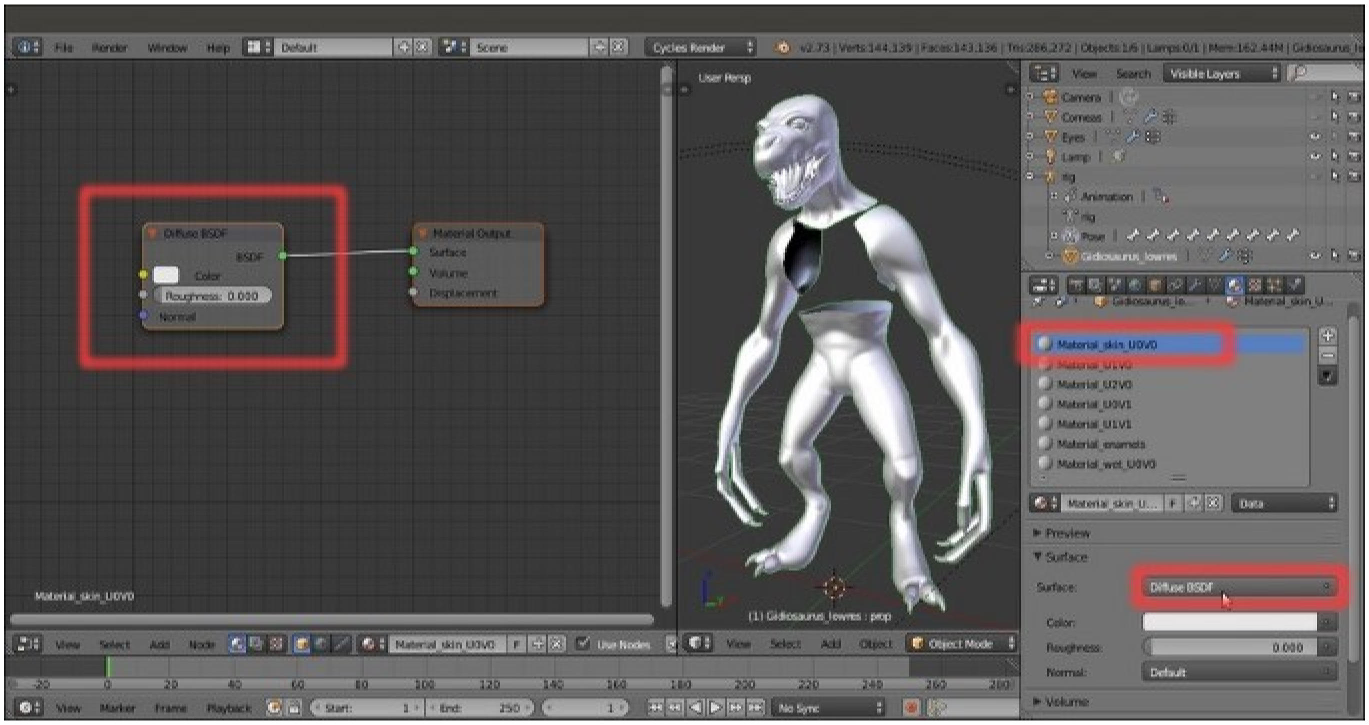
We know that the skin of our character is shared in **5** different materials; we are going to focus on the **head** (`Material_U0V0`), as the more representative one.

Once we are happy with the result, we will also copy (with all the due differences) the material to the other **body** parts.

Therefore, the steps are as follows:

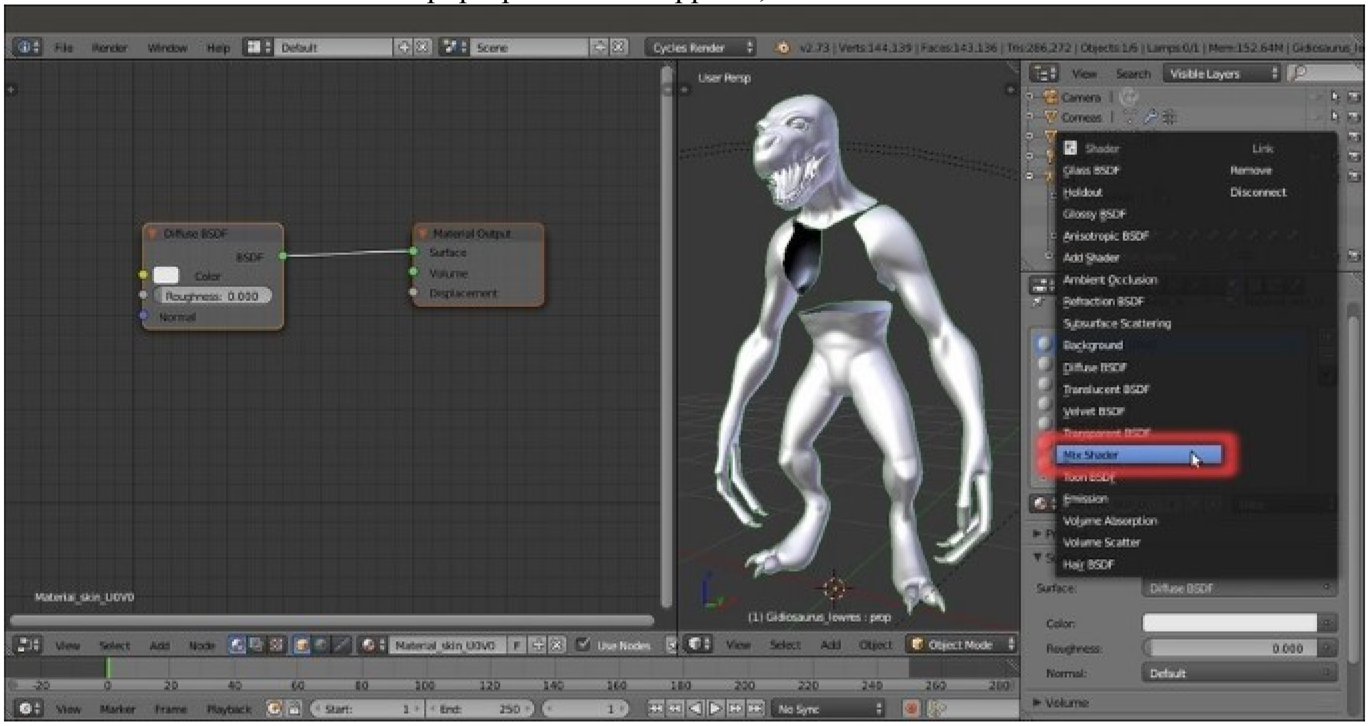
1. In the materials list inside the **Material** window, select the `Material_U0V0` (the first top one) and press *Ctrl* + left-click on it to rename it as `Material_skin_U0V0`; then, move down and click on the **Use Nodes** button inside the **Surface** subpanel.

   Immediately, a **Diffuse BSDF** shader node (already connected to a **Material Output** node) appears inside the **Node Editor** window to the left of the screen and listed in the **Surface** slot inside the **Surface** subpanel to the right:



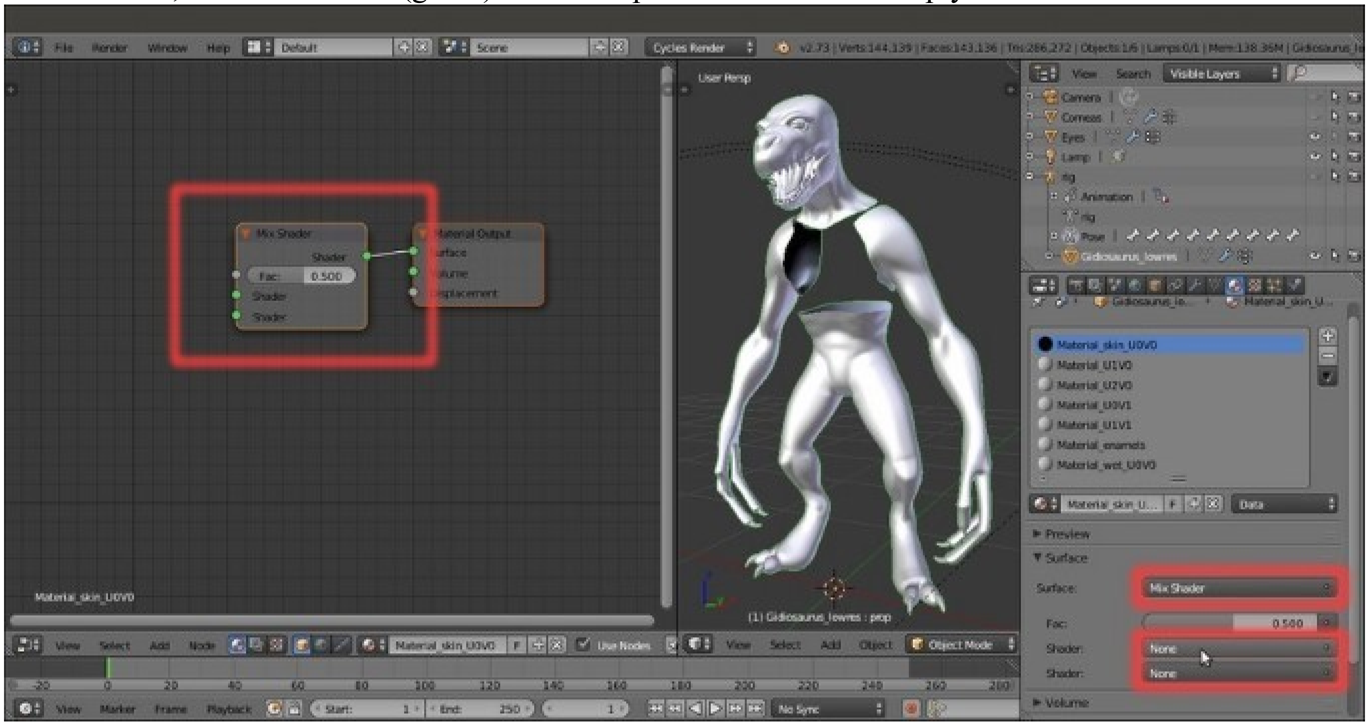
The Diffuse BSDF shader node connected to the Material Output node

2. In the **Surface** subpanel, under the **Material** window, click on the **Surface** slot that now shows the **Diffuse BSDF** shader: in the pop-up menu that appears, select a **Mix Shader** node:



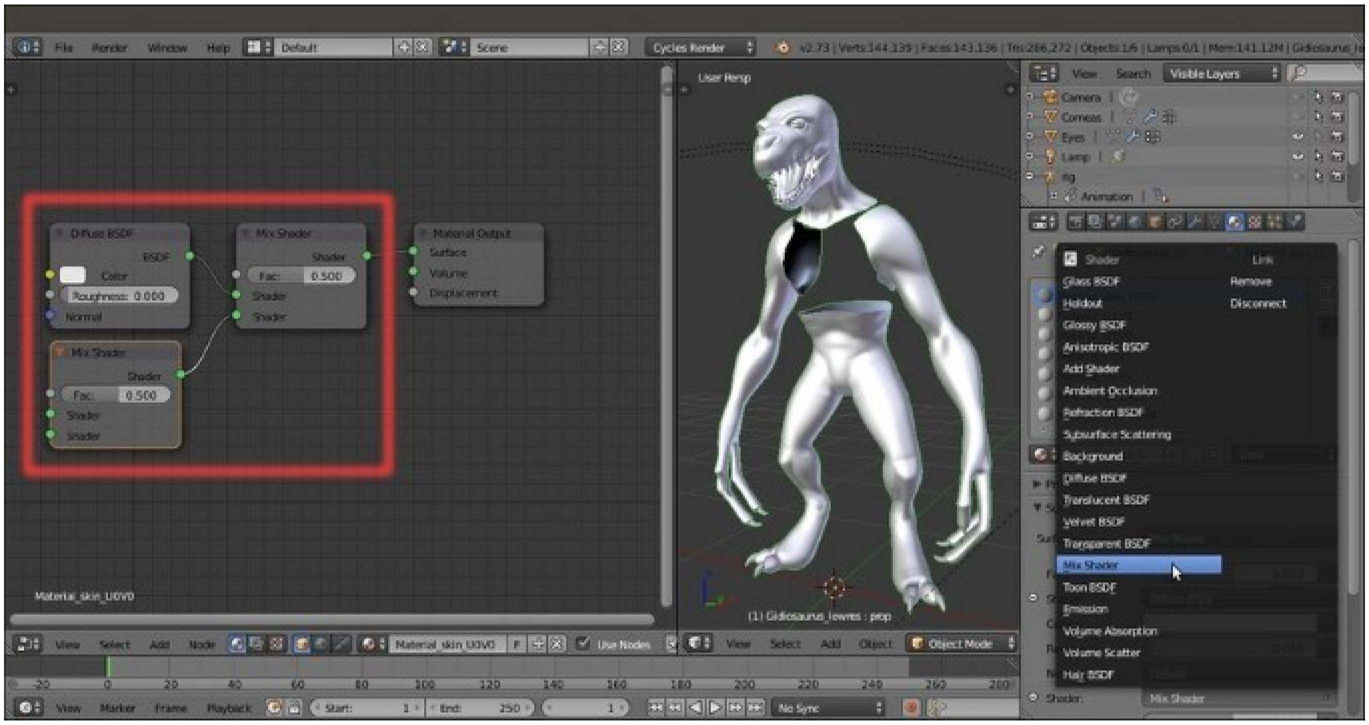
Switching the Diffuse BSDF shader node with a Mix Shader node through the Material window drop-down list

The **Surface** slot now shows the **Mix Shader** node item, and right below there are two new **Shader** slots that at the moment show the **None** item; in fact, looking at the nodes inside the **Node Editor** window, we see that the **Diffuse BSDF** shader node has been replaced by a **Mix Shader** node, and that the two (green) **Shader** input sockets are still empty:



The Mix Shader node with its two shader input sockets in the UV/Image Editor window and in the Material window

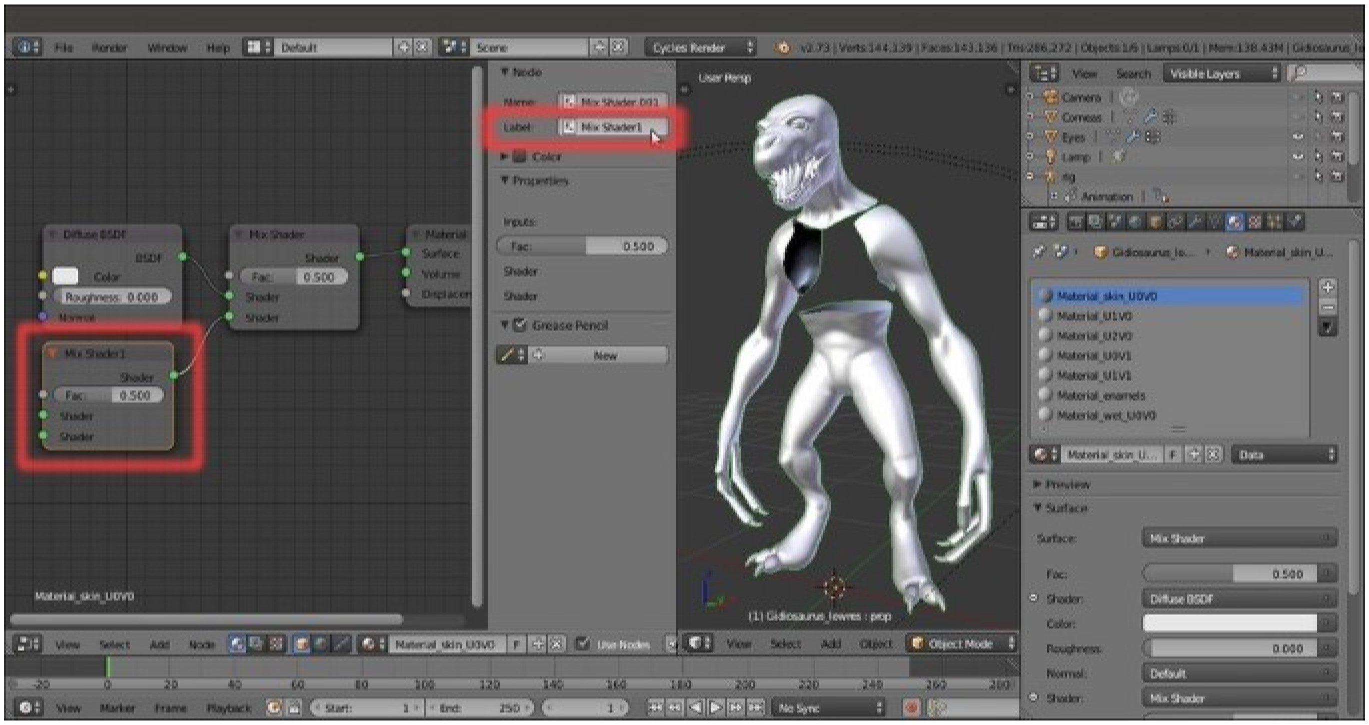
3. Click on the first **Shader** slot under the **Surface** subpanel to select, again from the pop-up menu, a **Diffuse BSDF** shader node; click on the second **Shader** slot and select a **Mix Shader** node; both the two new nodes are added and connected to the proper input socket, as we can see in the **Node Editor** window:



Two new nodes connected to the two shader input sockets of the Mix Shader node

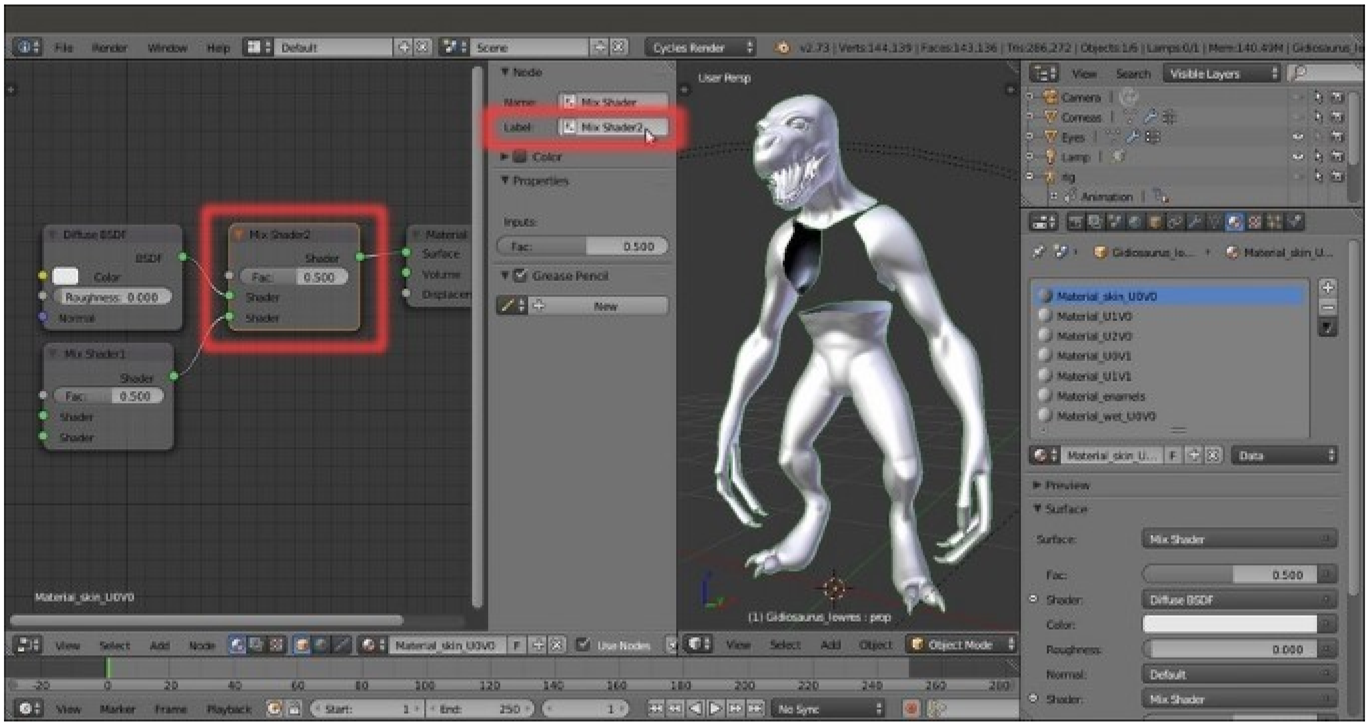
At this point, to avoid confusion, it's already better to start to label the various nodes with meaningful names.

4. Put the mouse pointer inside the **Node Editor** window and press the *N* key to call the **Properties** sidepanel.
5. Select the last **Mix Shader** node we added to the material and then go to click on the **Label** slot inside the top **Name** subpanel of the side **Properties** panel: type **Mix Shader1**:



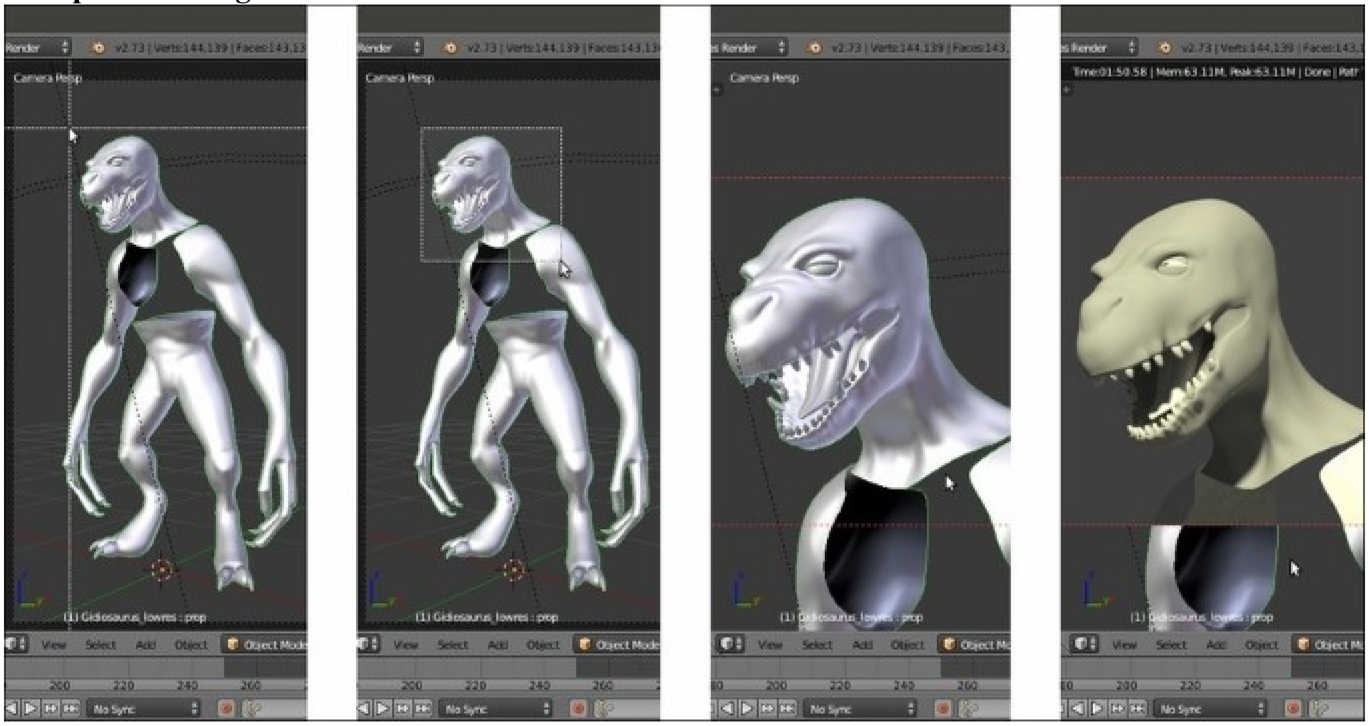
Labeling the nodes

6. Select the other **Mix Shader** node (the *old* one) and repeat the procedure by labeling it as **Mix Shader2**:



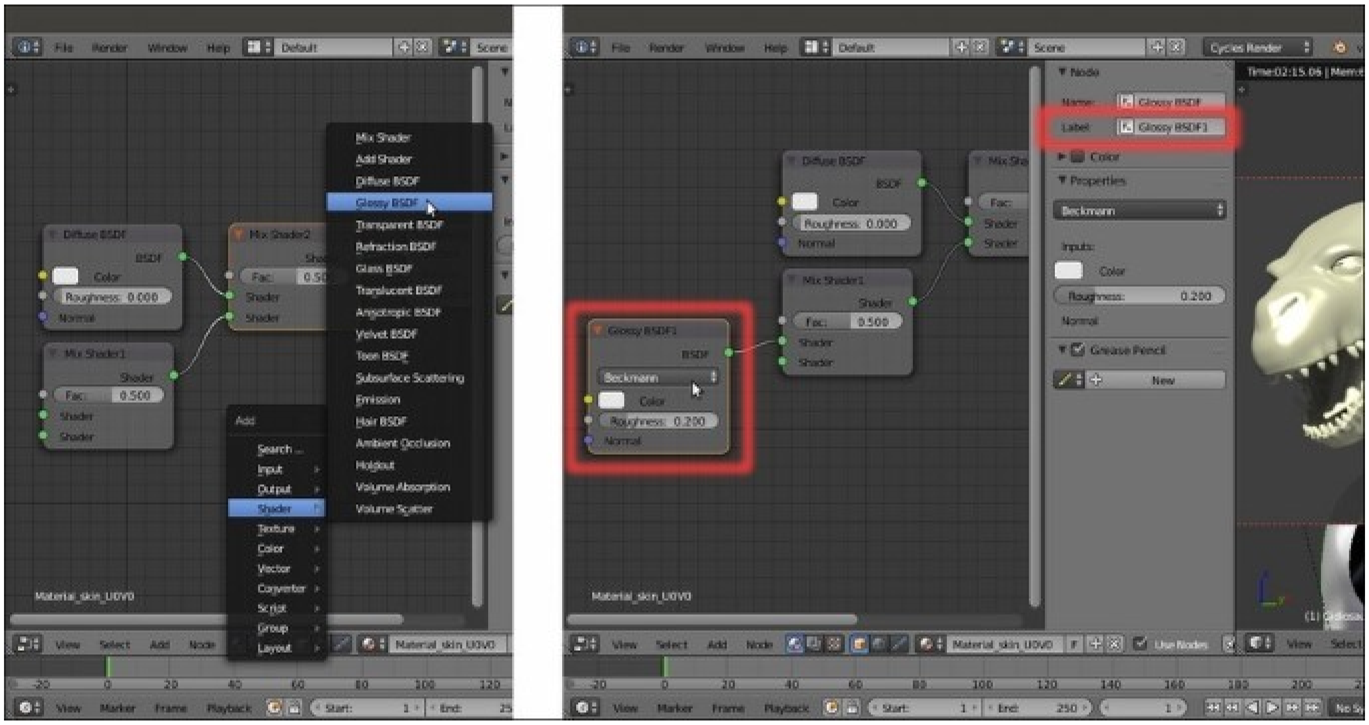
Labeling the nodes again

7. Put the mouse pointer on the 3D viewport and press the *0* key on the numpad to enter the **Camera** view.
8. Press *Shift* + *B* and by left-clicking draw a box around the **head** of the **Gidiosaurus** character to crop the area that can be rendered.
9. Zoom to the red square by scrolling the mouse wheel and then press *Shift* + *Z* to switch the **Viewport Shading** mode to **Rendered**:


Camera Persp
(1) Gidiosaurus_lowres : prop
View
Select
Add
Object
Object Mode
No Sync
v2.73 | Verts:144,139 | Faces:143,13

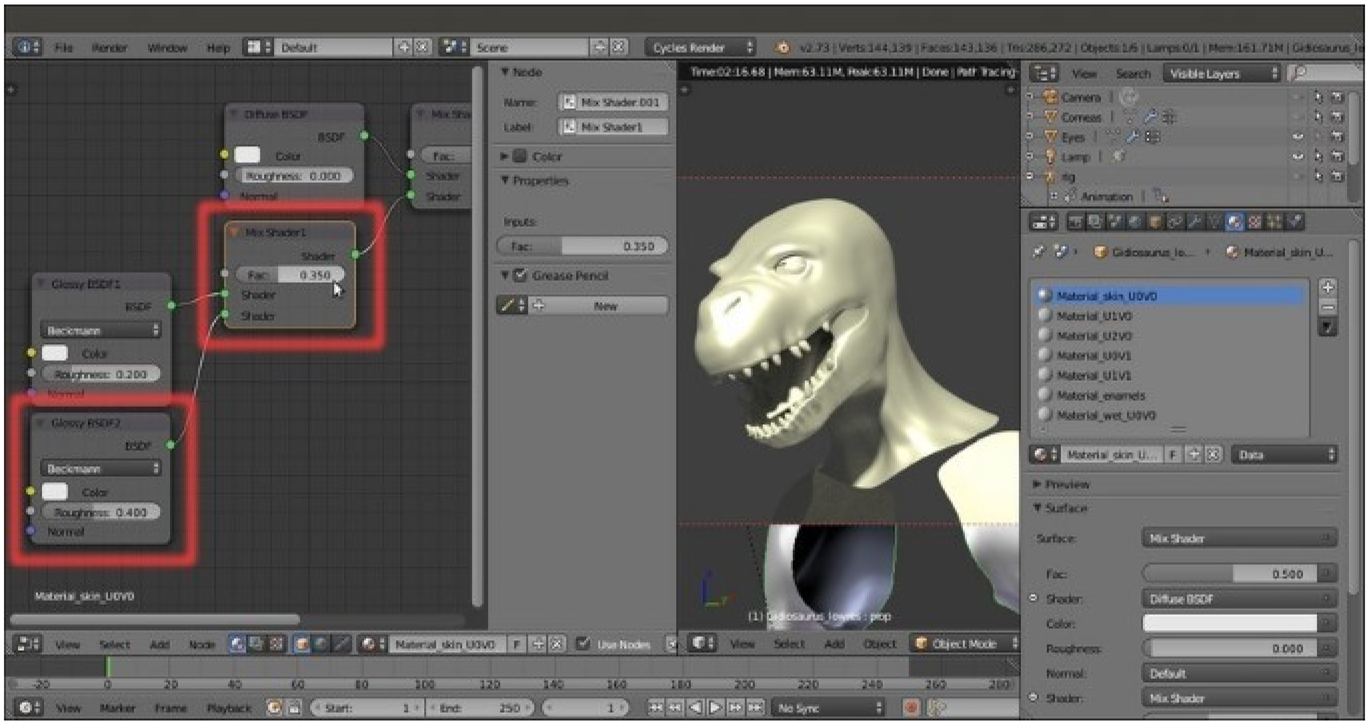
Cropping the renderable area and zooming to it

10. Put the mouse pointer inside the **Node Editor** window and press *Shift* + *A*. In the pop-up panel that appears, navigate to **Shader** and then click on the **Glossy BSDF** item to add the node; as it appears, move the mouse to place it to the left side of the **Mix Shader1** node.
11. Label it as **Glossy BSDF1**, connect its output to the first top **Shader** input socket of the **Mix Shader1** node, and set **Distribution** to **Beckmann**:



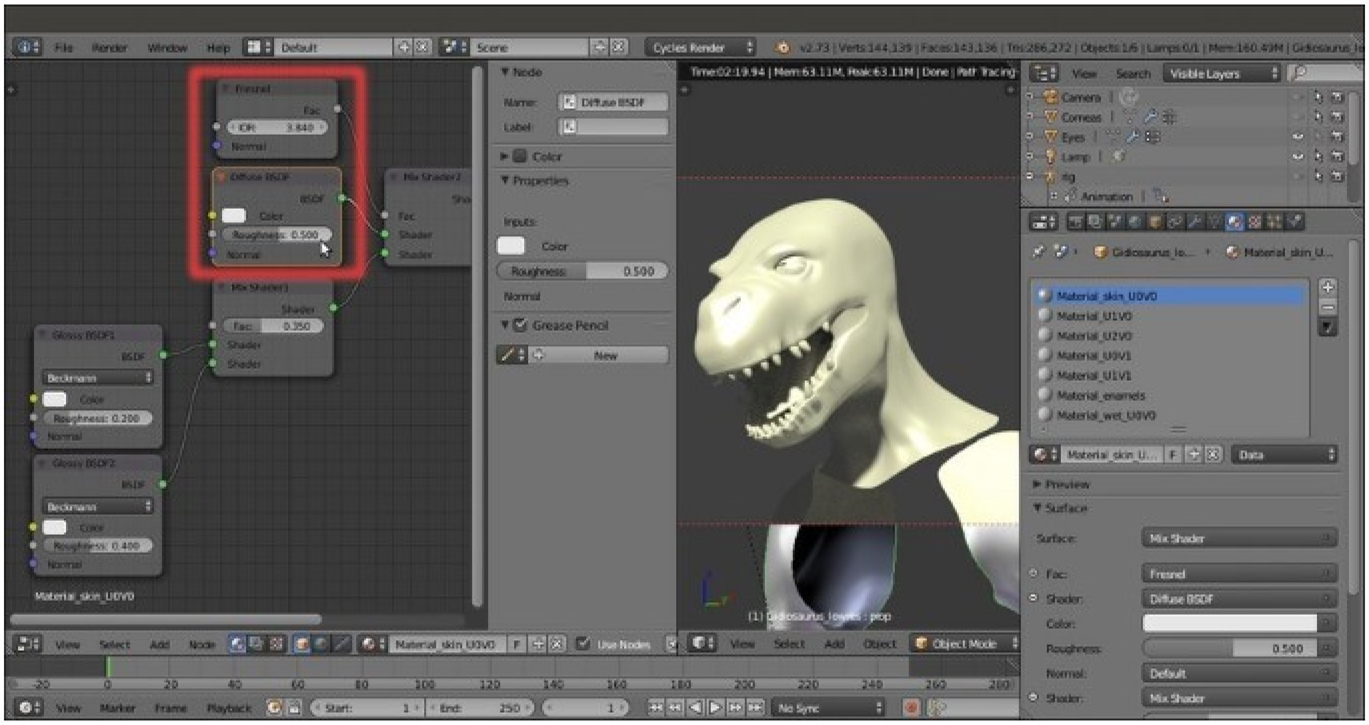
Adding a Glossy BSDF shader node and labeling it

12. Add a second **Glossy BSDF** shader node (*Shift* + *A* | **Shader** | **Glossy BSDF**) and place it right under the previous one; label it as **Glossy BSDF2**, connect its output to the second **Shader** input socket of the **Mix Shader1** node, and set **Distribution** to **Beckmann** as well and the **Roughness** to **0.400**.
13. Set the factor value (**Fac**) of the **Mix Shader1** node to **0.350**:



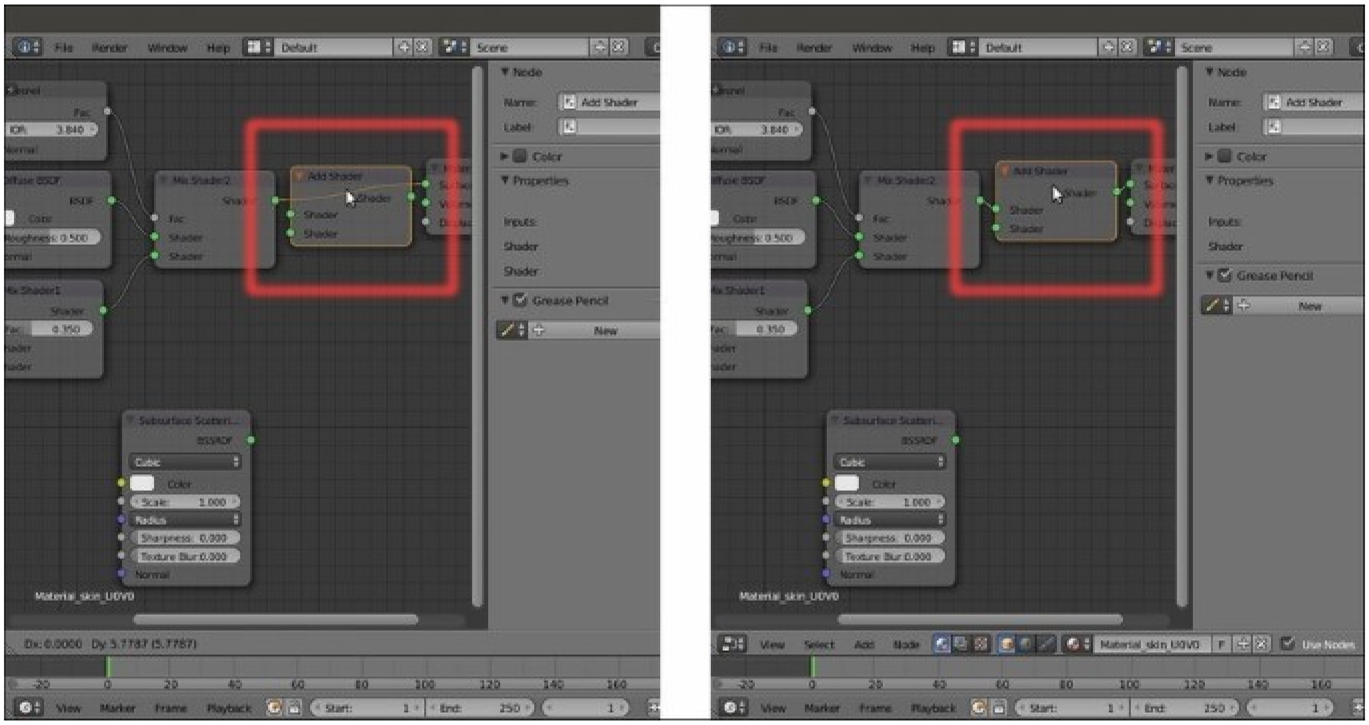
Adding a second Glossy shader node and blending it with the first one through the Fac value of the Mix Shader1 node

14. Add a **Fresnel** node (*Shift* + *A* | **Input** | **Fresnel**) and connect its **Fac** output to the **Fac** input socket of the **Mix Shader2** node; set the **IOR** value to **3.840**. Set the **Roughness** value of the **Diffuse BSDF** shader node to **0.500**:



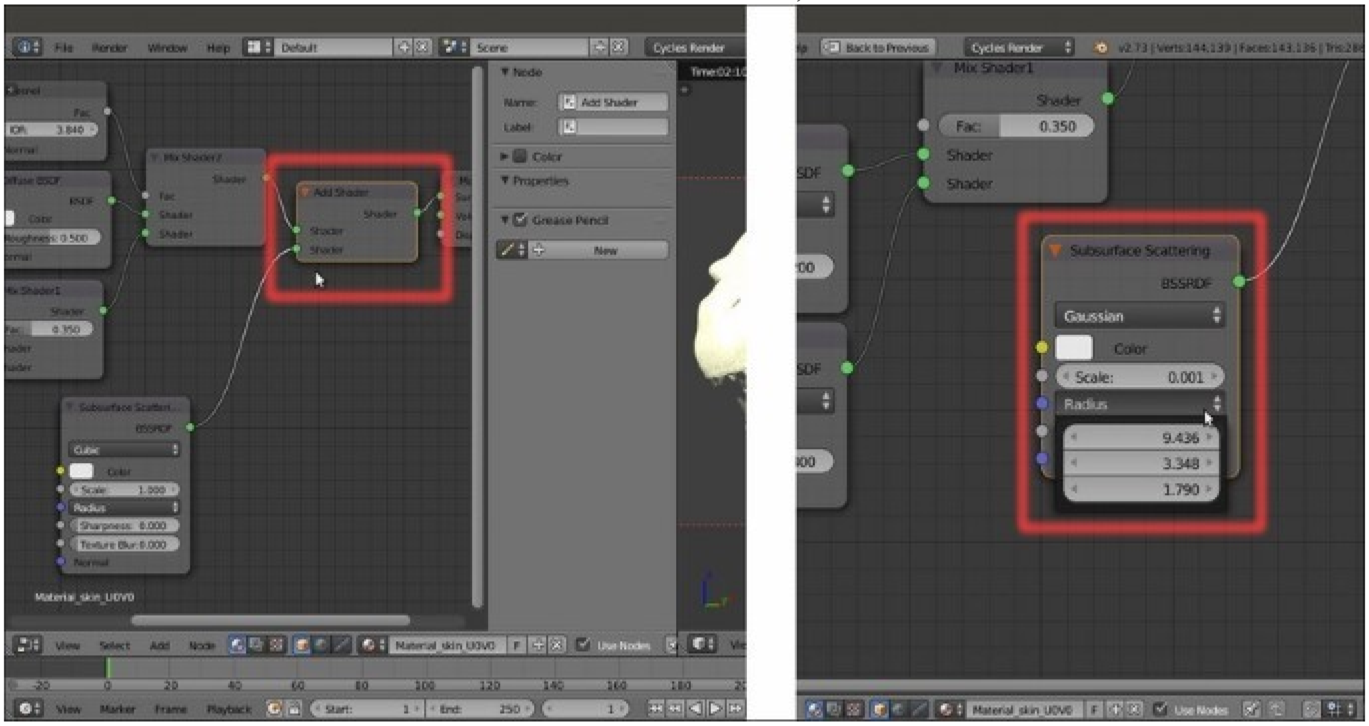
Adding a Fresnel node to set the Index of Refraction value to blend the diffuse with the glossy components

15. Add a **Subsurface Scattering** node (*Shift* + *A* | **Shader** | **Subsurface Scattering**) and an **Add Shader** node (*Shift* + *A* | **Shader** | **Add Shader**). Move this last one to the link that connects the **Mix Shader2** node to the **Material Output** node in order to paste it automatically between the two nodes (automatically when the connection line becomes highlighted):



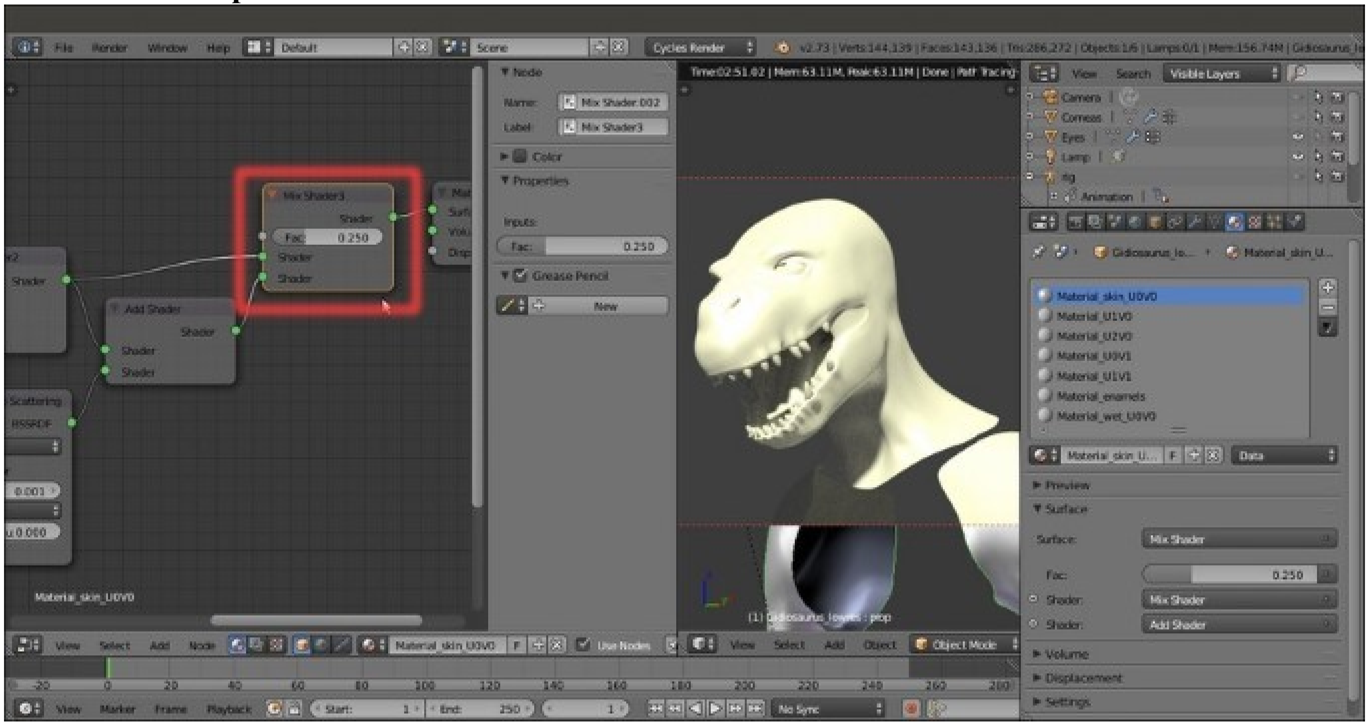
Automatically joining the Add Shader node

16. Connect the output of the **Subsurface Scattering** node to the second **Shader** input socket of the **Add Shader** node. In the **SSS** node, change **Fallof** from **Cubic** to **Gaussian**, set the **Scale** to **0.001** and click on the **Radius** button to set the **RGB** to **9.436**, **3.348** and **1.790**:



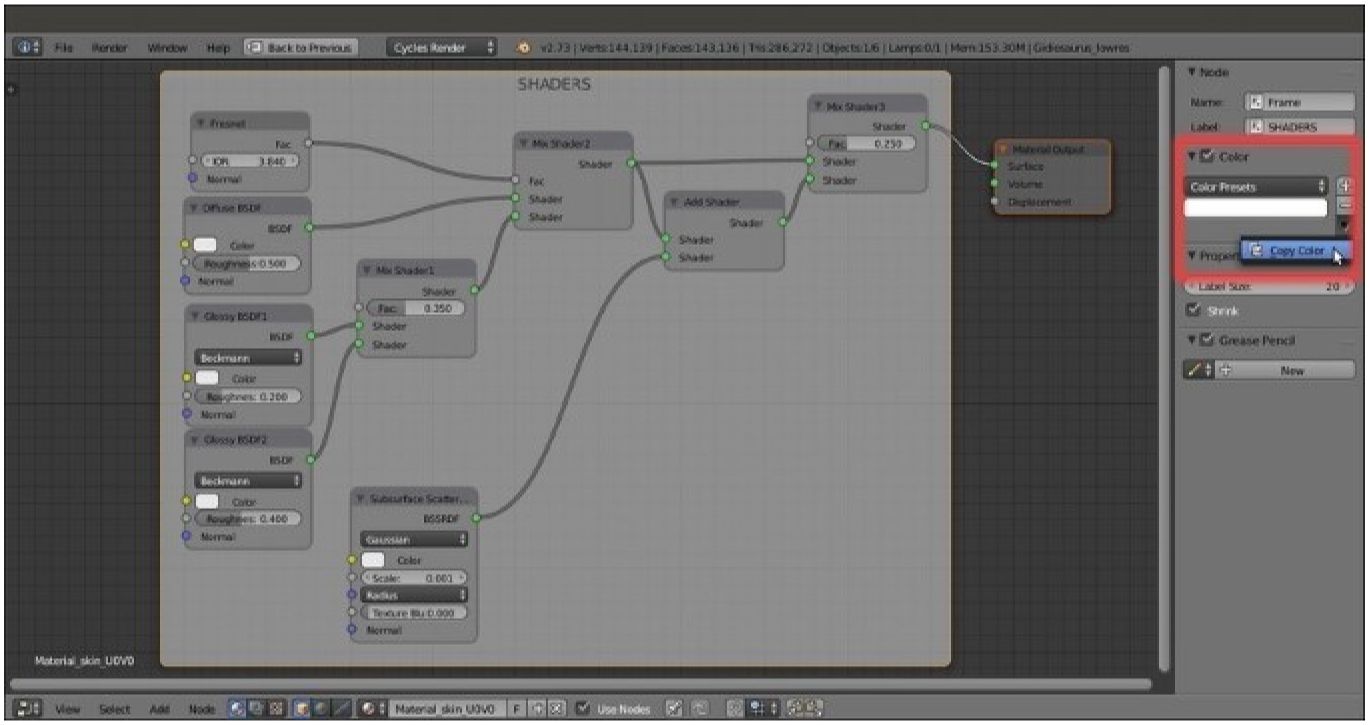
Connecting and setting the SSS node

17. Add a new **Mix Shader** node (*Shift* + *A* | **Shader** | **Mix Shader**) and label it as **Mix Shader3**. Connect the output of the **Mix Shader2** node to the first **Shader** input socket of the **Mix Shader3** node, and the output of the **Add Shader** node to its second **Shader** input socket. Set the **Fac** of the **Mix Shader3** node to **0.250** and connect its output to the **Surface** input socket of the **Material Output** node:



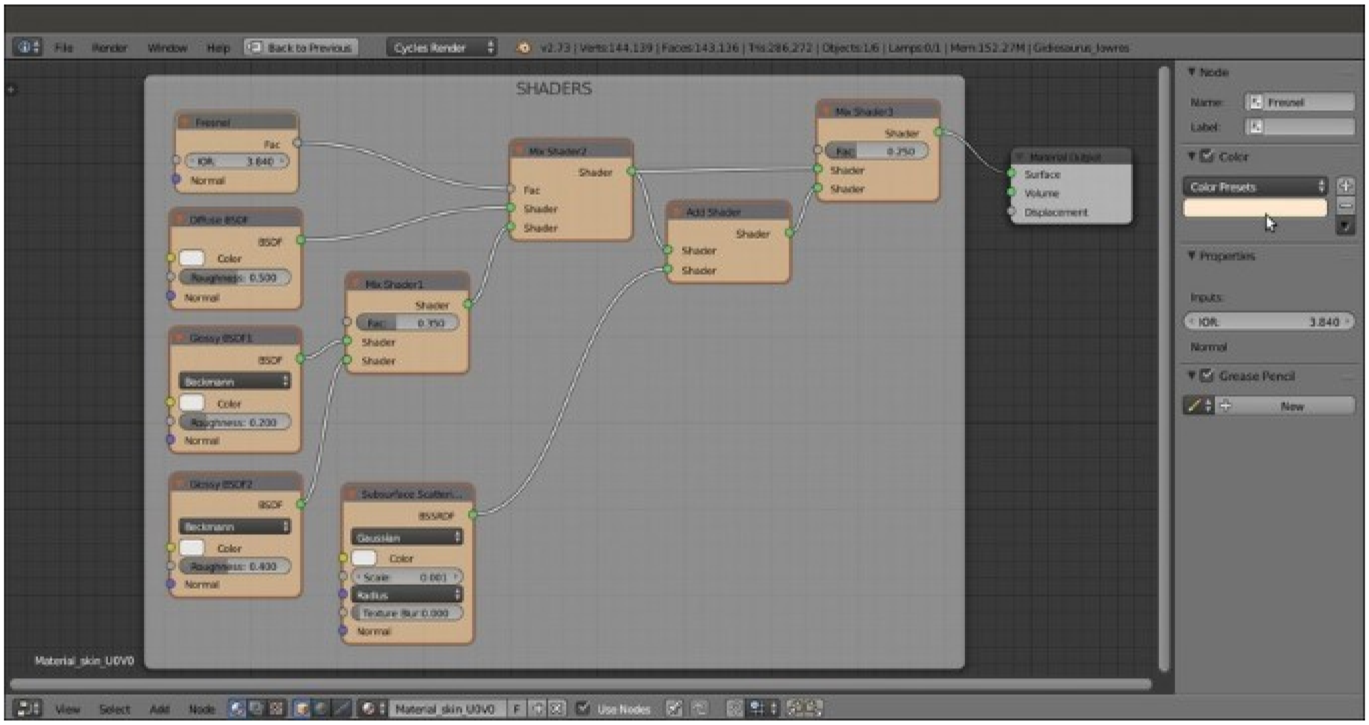
A little trick to tweak the influence of the Add shader node

18. Add a **Frame** (*Shift* + *A* | **Layout** | **Frame**), box-select all the nodes (except the **Material Output** node) and then press *Ctrl* + *P* to parent them to the frame; label the frame as **SHADERS**.
19. Select the **SHADERS** frame and go to the **Properties** sidepanel. Expand the **Color** subpanel (right under the **Node** subpanel) by clicking on the little horizontal black arrow, and enable the **Color** checkbox.
20. Click on the color slot and set a light color of your choice (I set it to **RGB 1.000**, which is totally white). Then click on the **+** icon button to the side and in the **Name** slot of the **Add Node Color Preset** pop-up panel, write **Frame**, then click the big **OK** button.
21. Select the **Material Output** node and then *Shift*-select the **Frame** again, then go to the **Color** subpanel and click on the big vertical arrow under the **+** and **–** icon buttons to the side. Click on the **Copy Color** item to copy the color of the **Frame** to the **Material Output** node:



The SHADERS frame with the nodes and the Copy Color tool under the N sidepanel

22. Select any one of the other nodes, for example the **Fresnel** node, enable the **Color** checkbox and set a new color of your choice (for these nodes, I set it to **R 1.000, G 0.819**, **B 0.617**, which is a light brown).
23. Click on the **+** icon button to the side and in the **Name** slot of the **Add Node Color Preset** pop-up panel, write **Shaders**, then click the big **OK** button.
24. Now box-select all the other nodes inside the frame and click on the **Copy Color** item to copy the color from the **Fresnel** node to all the other selected nodes at once:



Copying the label color from one node to all the other selected nodes

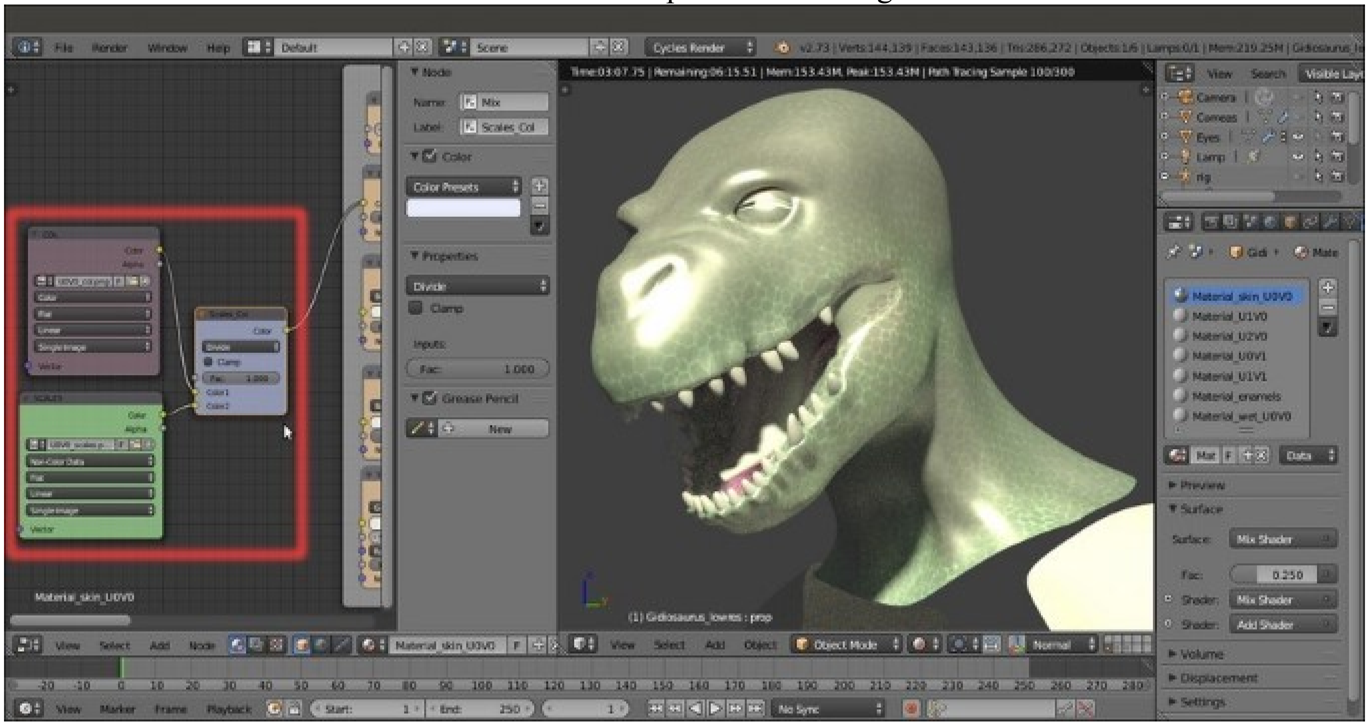
At this point we have completed the basic shader for the skin; what we have to do now is to add the textures we painted in both Chapter 10, *Creating the Textures*, and Chapter 11, *Refining the Textures*.

So:

25. Put the mouse pointer into the **Node Editor** window and add an **Image Texture** node (*Shift* + *A* | **Texture** | **Image Texture**); label it as **COL** and then use *Shift* + *D* to duplicate it; move the duplicated one down and change its label to **SCALES**.

    As you label the newly added nodes, also assign colors to them to make them more easily readable inside the **Node Editor** window, and save these colors as presets as we did at step 20.
26. Click on the **Open** button of the **COL** node and browse to the `textures` folder. There, load the image `U0V0_col.png`.
27. Click on the **Open** button of the **SCALES** node and browse to the `textures` folder. There, load the image `U0V0_scales.png`; set the **Color Space** to **Non-Color Data**.
28. Add a **MixRGB** node (*Shift* + *A* | **Color** | **MixRGB**) and label it as **Scales_Col**; connect the **Color** output of the **COL** node to the **Color1** input socket of the **Scales_Col** node and the **Color** output of the **SCALES** node to its **Color2** input socket. Set the **Fac** to **1.000** and the **Blend Type** to **Divide**.
29. Connect the output of the **Scales_Col** node to the **Color** input socket of the **Diffuse BSDF** shader node inside the **SHADERS** frame.

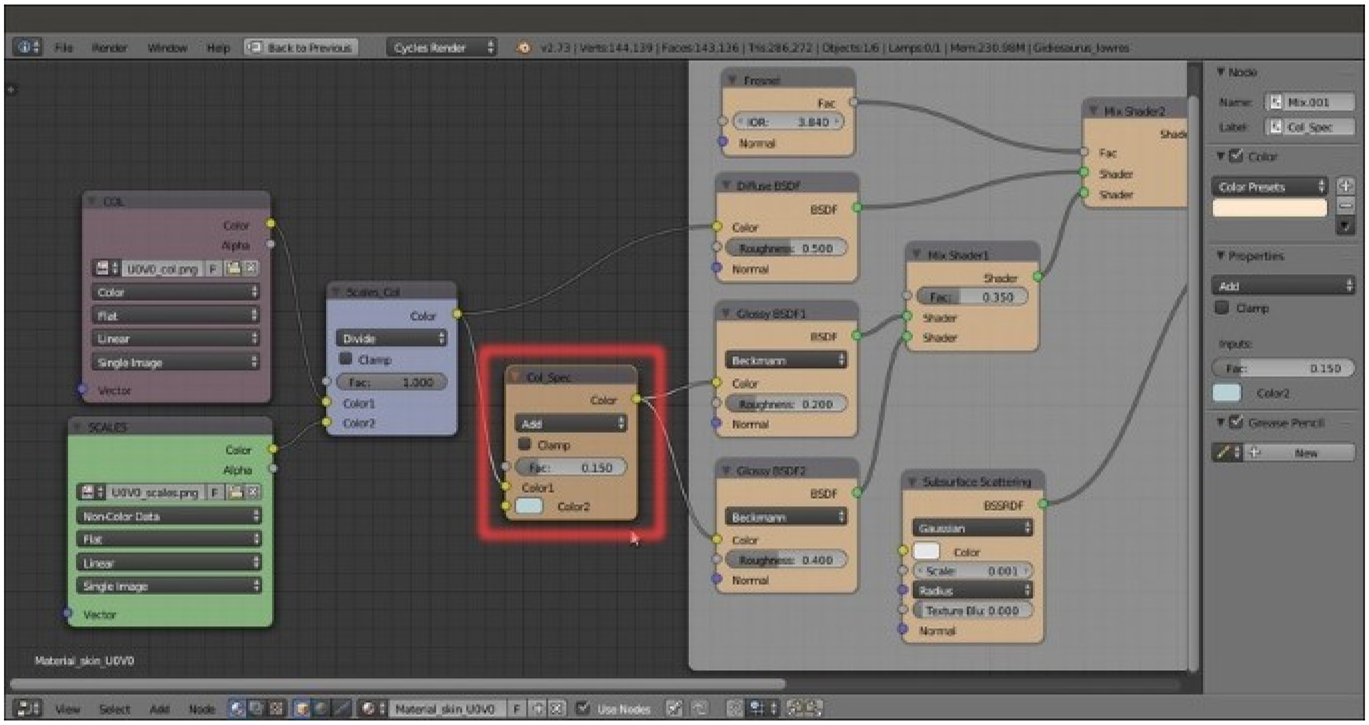
The result so far is visible in the real-time rendered preview to the right:



The rendered result of the two combined image texture nodes

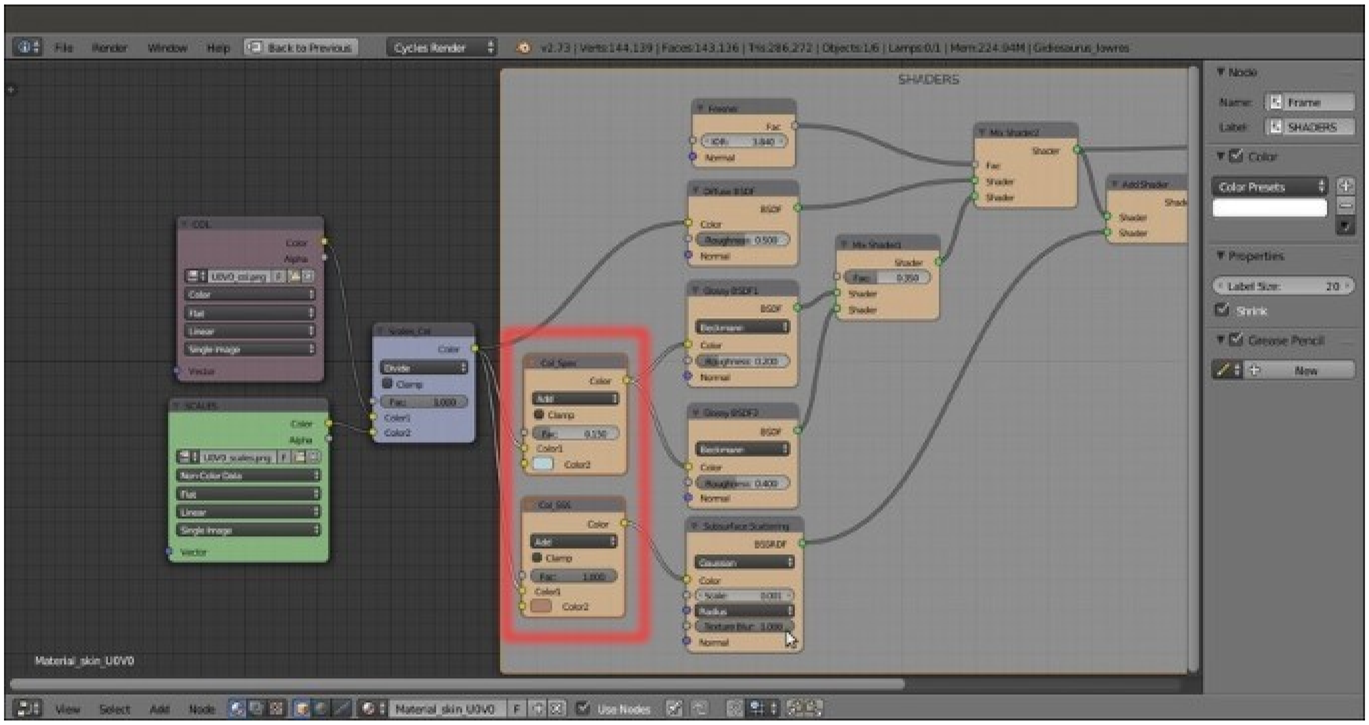
As you can see, the glossy component is strong in this one! We must lessen the effect, to obtain a more natural look.

30. Add a new **MixRGB** node (*Shift* + *A* | **Color** | **MixRGB**) and label it as **Col_Spec**; set the **Color2** to **R 0.474**, **G 0.642**, **B 0.683**, then also connect the output of the **Scales_Col** node to the **Color1** input socket of the **Col_Spec** node.
31. Set the **Fac** value to **0.150** and the **Blend Type** to **Add**, then connect its output to the **Color** input sockets of both the **Glossy BSDF1** and **Glossy BSDF2** nodes:



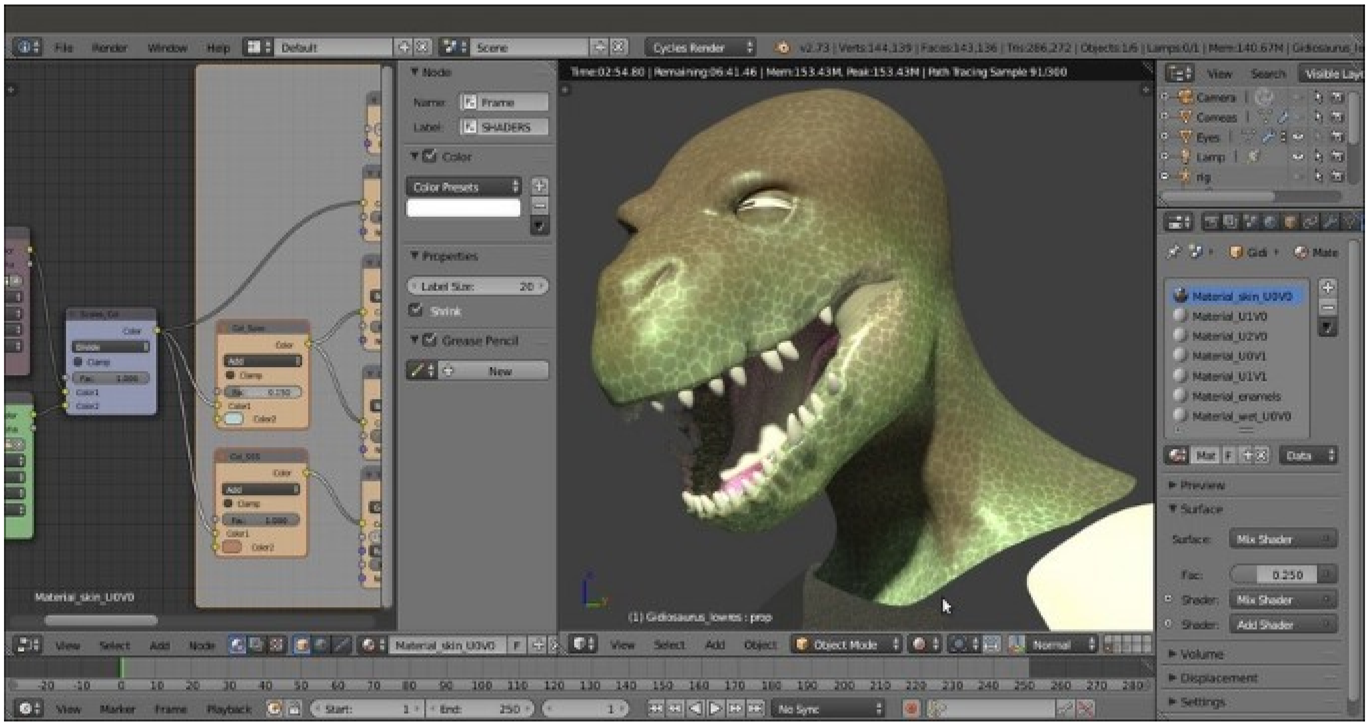
Varying the textures color output for the glossy component

32. Press *Shift* + *D* to duplicate the **Col_Spec** node and label the duplicate as **Col_SSS**; set the **Fac** value to **1.000** and the **Color2** to **R 0.439**, **G 0.216**, **B 0.141**. Connect the **Color** output of the **Scales_Col** node to the **Color1** input socket of the **Col_SSS** node and the output of this latter node to the **Color** input socket of the **Subsurface Scattering** node; increase its **Texture Blur** to the maximum value.
33. *Shift*-select the **Col_Spec** and the **Col_SSS** nodes and then also the **SHADERS** frame, and press *Ctrl* + *P* to parent them:



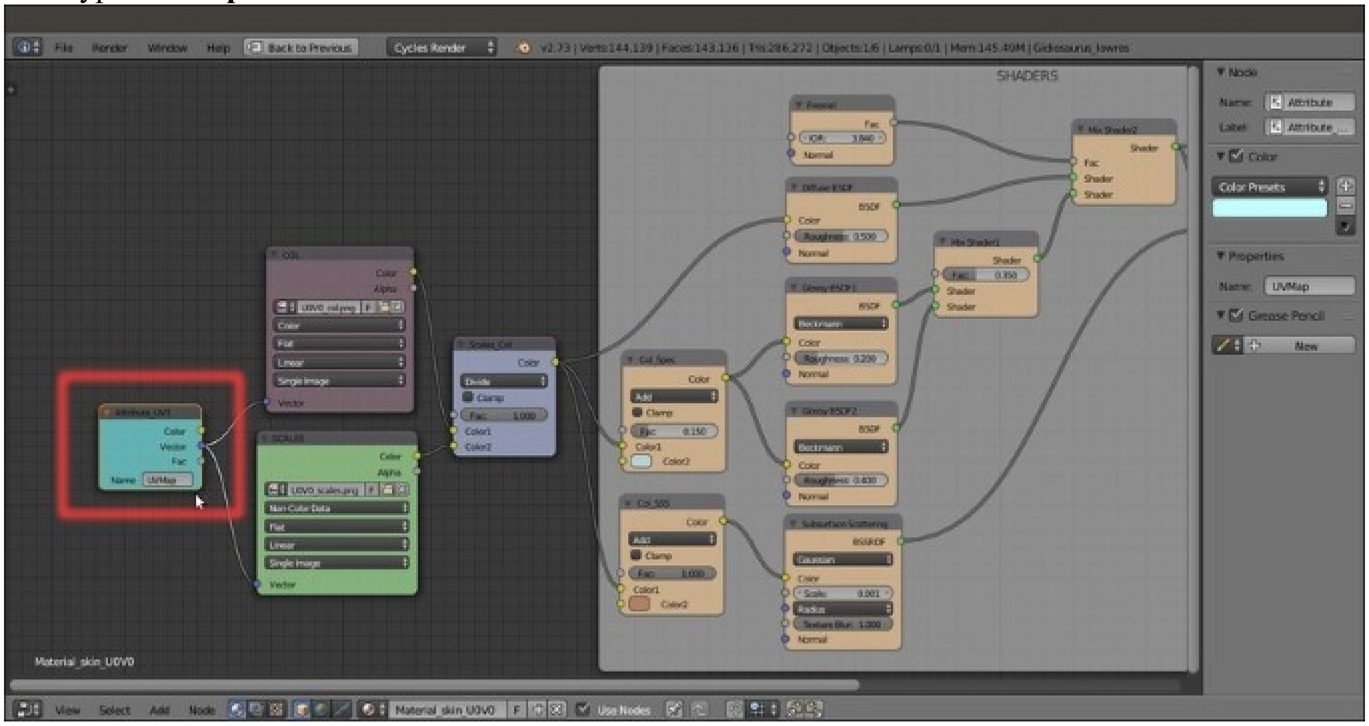
Varying the textures color output also for the SSS node

The new result looks a lot better:



A better result

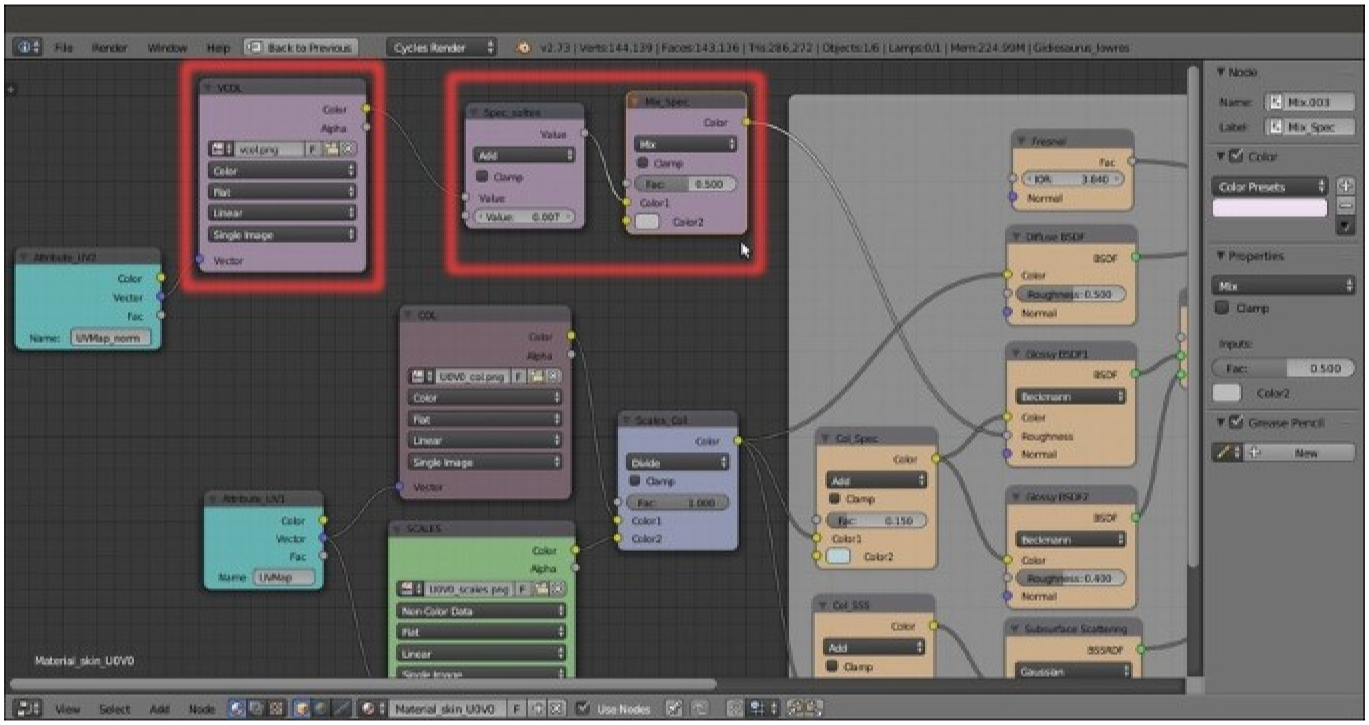
34. Add an **Attribute** node (*Shift* + *A* | **Input** | **Attribute**) and label it as **Attribute_UV1**. Connect its **Vector** output to the **Vector** input sockets of the **COL** and **SCALES** nodes and in the name field type **UVMap**:



Adding the Attribute node to establish the UV coordinates layer to be used

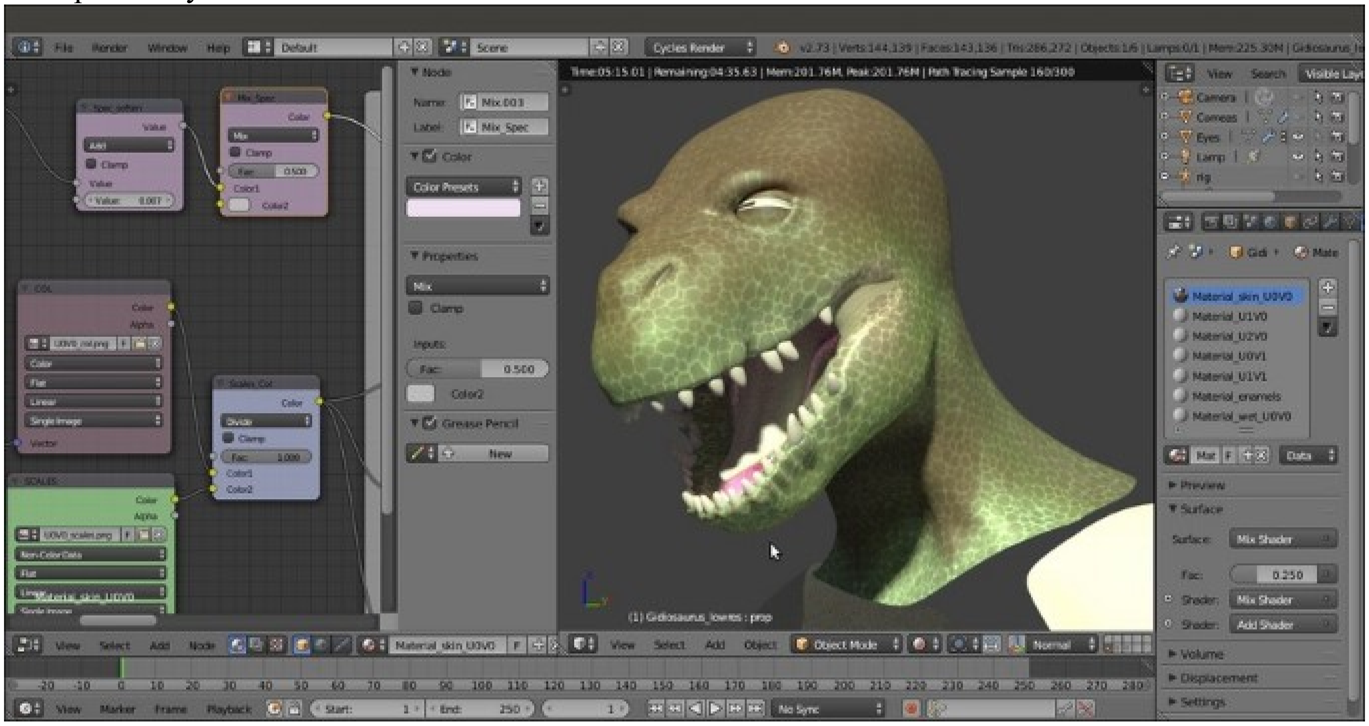
By the way, the glossy component is still a little unnatural.

35. Add a new **Image Texture** node (*Shift* + *A* | **Texture** | **Image Texture**) and label it as **VCOL**. Click on the **Open** button, browse to the `texture` folder and load the image `vcol.png`.
36. Press *Shift* + *D* to duplicate the **Attribute** node, change the label to **Attribute_UV2**, and change the **Name** field to **UVMap_norm**. Connect its **Vector** output to the **Vector** input of the **VCOL** node.
37. Add a **Math** node (*Shift* + *A* | **Converter** | **Math**) and a **MixRGB** node (*Shift* + *A* | **Color** | **MixRGB**); connect the **Color** output of the **VCOL** node to the first **Value** input socket of the **Math** node; label this one as **Spec_soften** and set the second **Value** to **0.007**. Connect its **Value** output to the **Color1** input socket of the **MixRGB** node, which is now labeled as **Mix_Spec**.
38. Connect the **Color** output of the **Mix_Spec** node to the **Roughness** input socket of the **Glossy BSDF1** node:



Using the baked Vertex Color image to "soften" the character's skin specularity

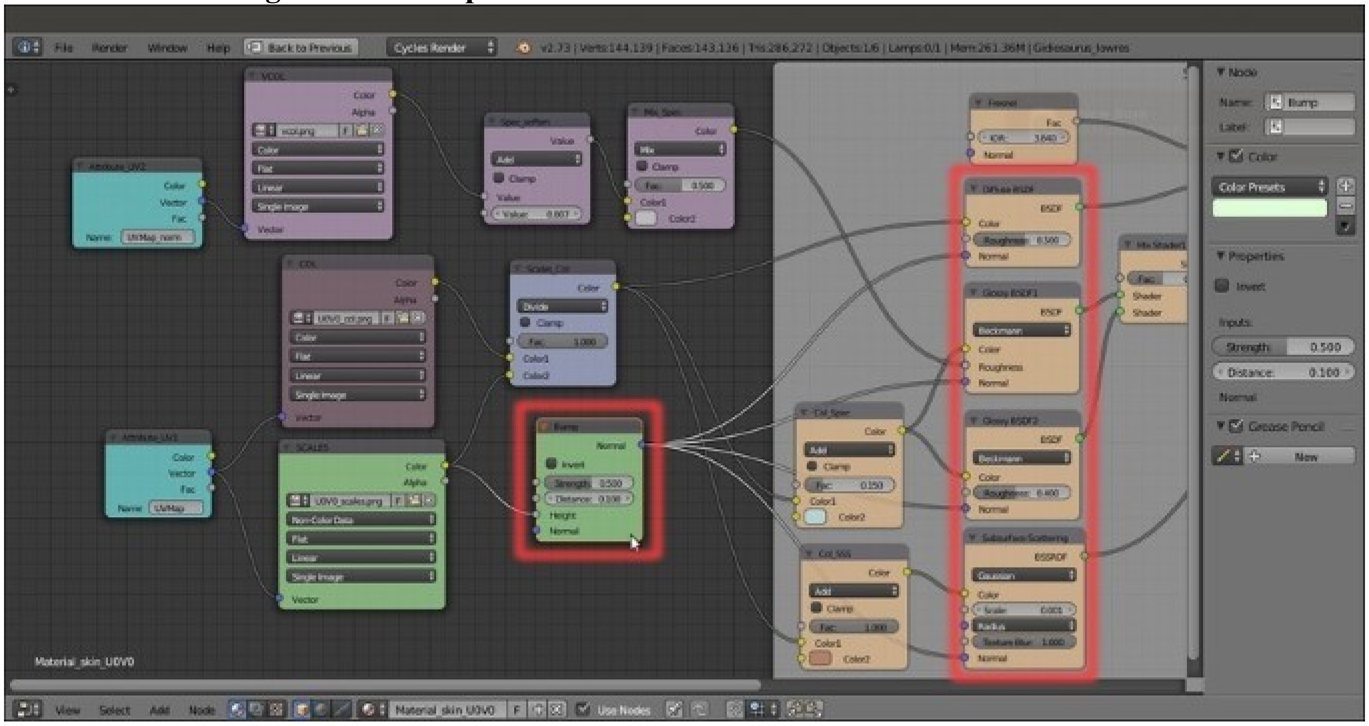
The specularity is now a bit more realistic:



And the rendered result of this operation

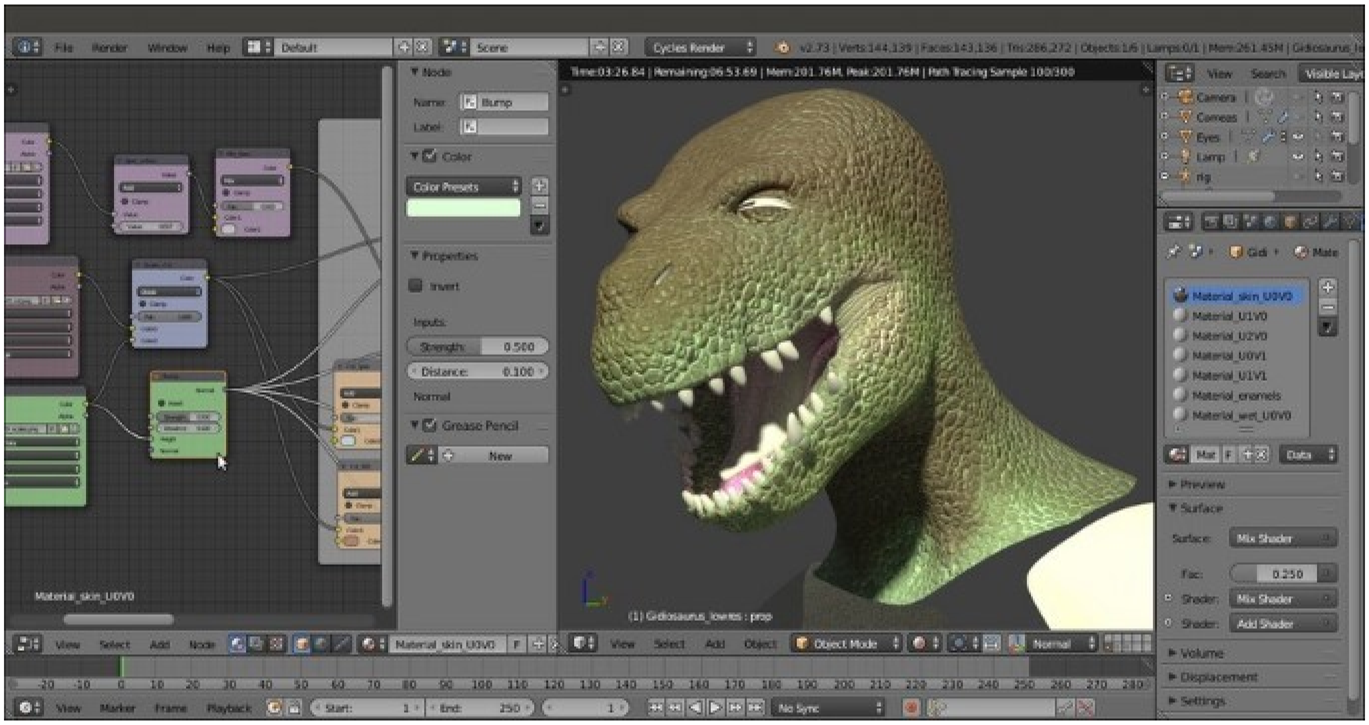
Anyway, it's still missing the contribution of the bump effect.

39. Add a **Bump** node (*Shift* + *A* | **Vector** | **Bump**); connect the output of the **SCALES** node to the **Height** input socket of the **Bump** node and the **Normal** output of this latter node to the **Normal** input socket of the **Diffuse BSDF**, **Glossy BSDF1**, **Glossy BSDF2**, and **Subsurface Scattering** nodes. Set the **Strength** of the **Bump** node to **0.500**:



Adding the bump pattern to the shaders

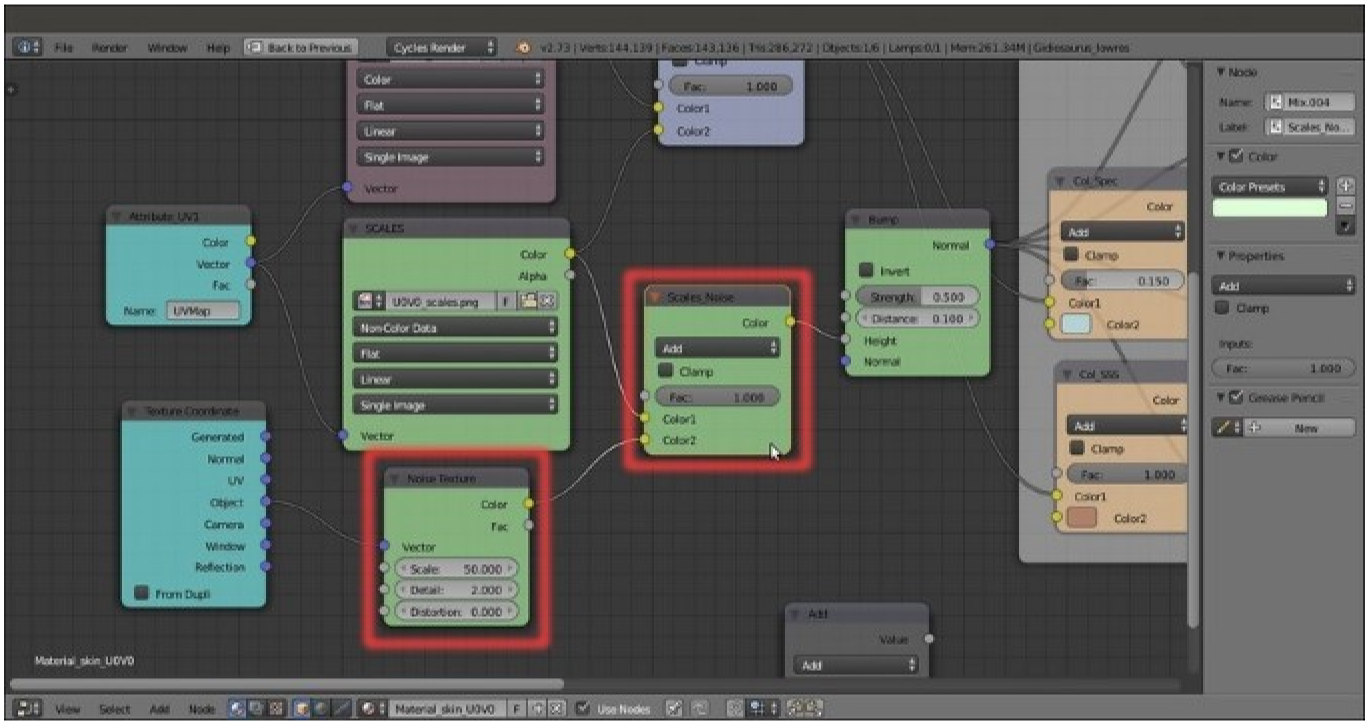
Now we start to see something!



The bump effect in the rendered preview

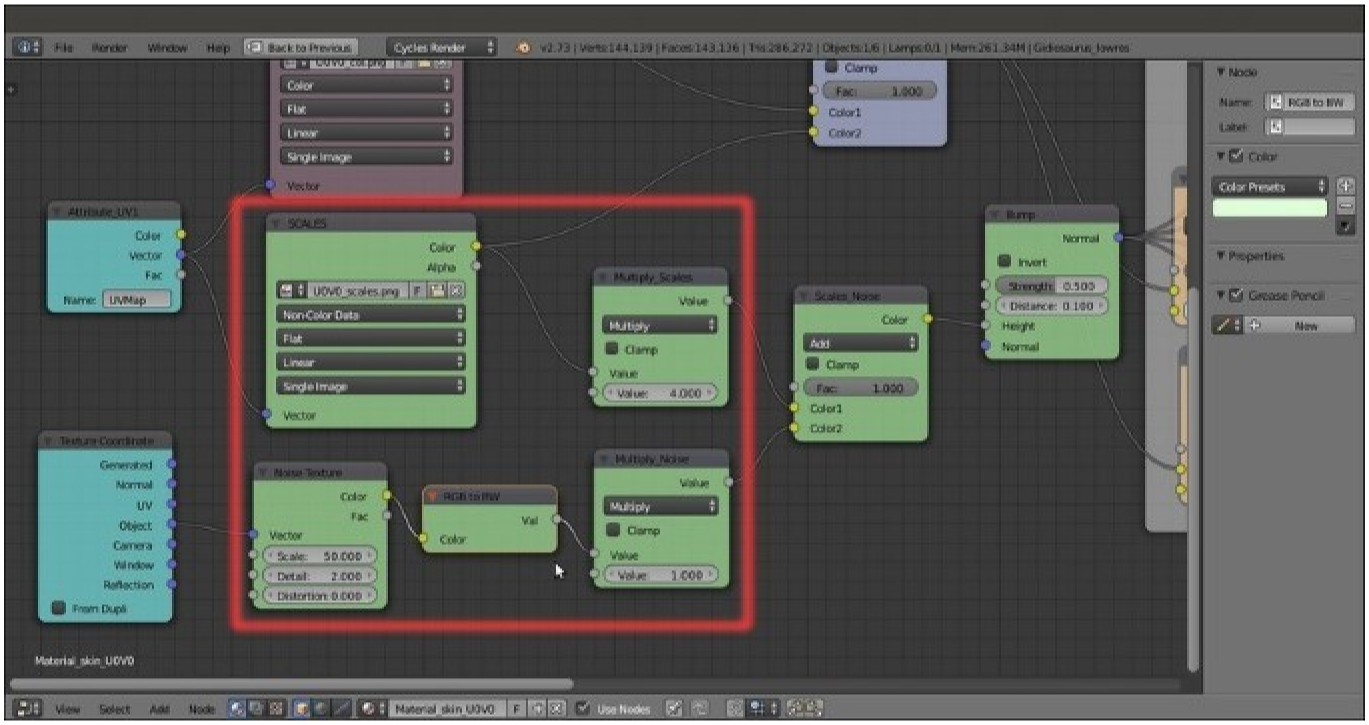
By the way, the bump pattern is too even and, therefore, unrealistic; we must therefore *break* it in some way.

40. Add a **Noise Texture** node (*Shift* + *A* | **Texture** | **Noise Texture**) and a **Texture Coordinate** node (*Shift* + *A* | **Input** | **Texture Coordinate**). Connect the **Object** output of the **Texture Coordinate** node to the **Vector** input socket of the **Noise Texture** node, then set the **Scale** of the texture to **50.000**.
41. Add a **Math** node (*Shift* + *A* | **Converter** | **Math**) and a **MixRGB** node (*Shift* + *A* | **Color** | **MixRGB**). Connect the **Color** output of the **SCALES** node to the **Color1** input socket of the **MixRGB** node, and the **Color** output of the **Noise Texture** to the **Color2** input socket.
42. Set the **MixRGB** blend type to **Add**, the **Fac** value to **1.000** and label it as **Scales_Noise**. To see the effect, connect its **Color** output to the **Height** input socket of the **Bump** node (but this is going to change very soon, so it's not mandatory at this step):



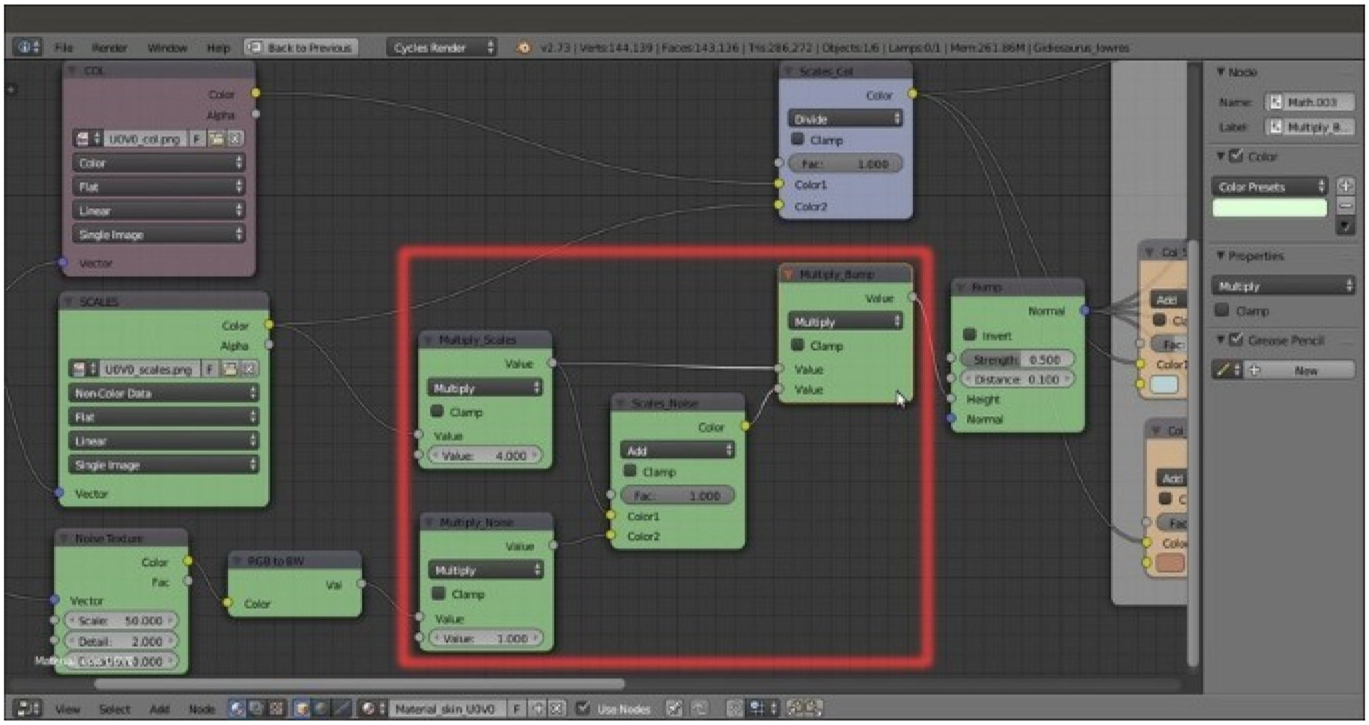
Adding some noise to the bump pattern part 1

43. Select the **Math** node and move it on the link connecting the **Noise Texture** node with the **Scales_Noise** node to paste it in between them: set the **Operation** to **Multiply**, the second **Value** to **1.000**, and label it as **Multiply_Noise**.
44. Press *Shift* + *D* to duplicate the **Multiply_Noise** node, change the label to **Multiply_Scales** and the second **Value** to **4.000**; paste it between the **SCALES** node and the **Scales_Noise** node.
45. Add an **RGB to BW** node (*Shift* + *A* | **Converter** | **RGB to BW**) and paste it between the **Noise Texture** node and the **Multiply_Noise** one:



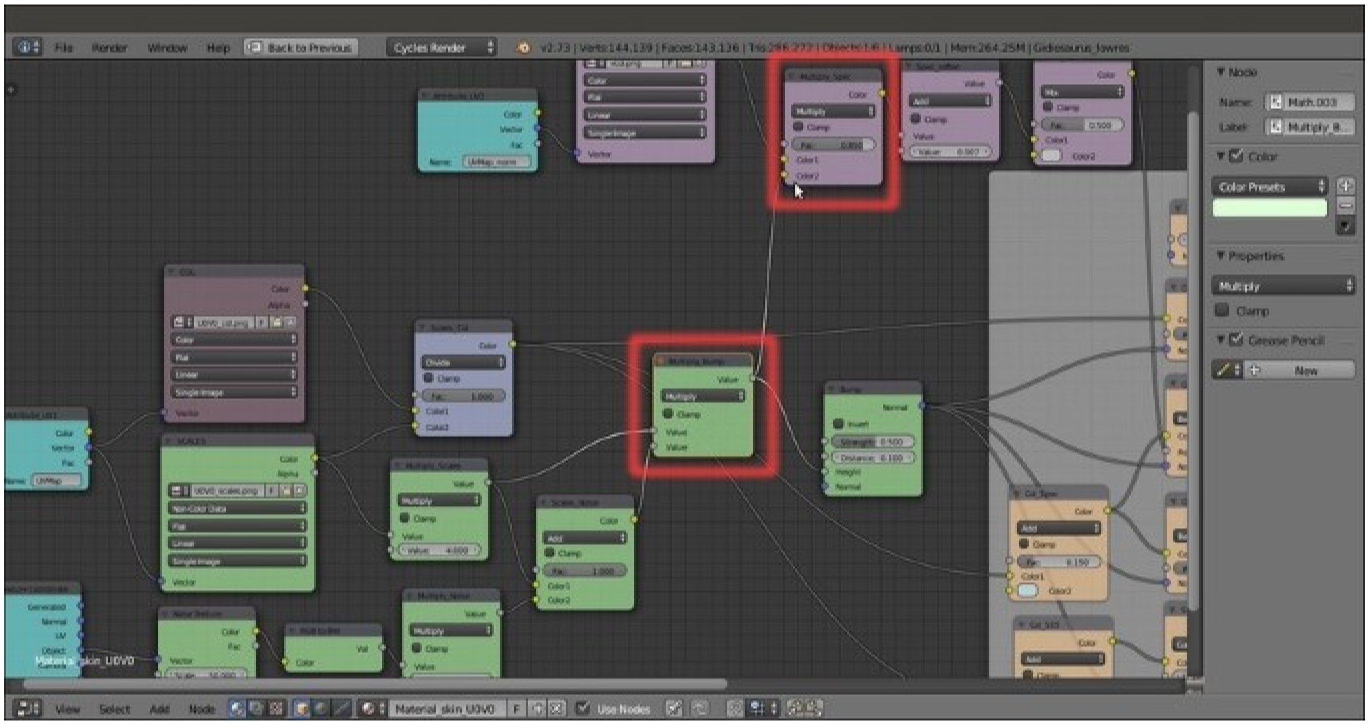
Adding some noise to the bump pattern part 2

46. Press *Shift* + *D* to duplicate the **Multiply_Scales** node and change the duplicate label to **Multiply_Bump**; connect the output of the **Multiply_Scales** to the first **Value** input socket of the **Multiply_Bump** node and the output of the **Scales_Noise** node to the second **Value** input socket. Connect the output of the **Multiply_Bump** node to the **Height** input socket of the **Bump** node:



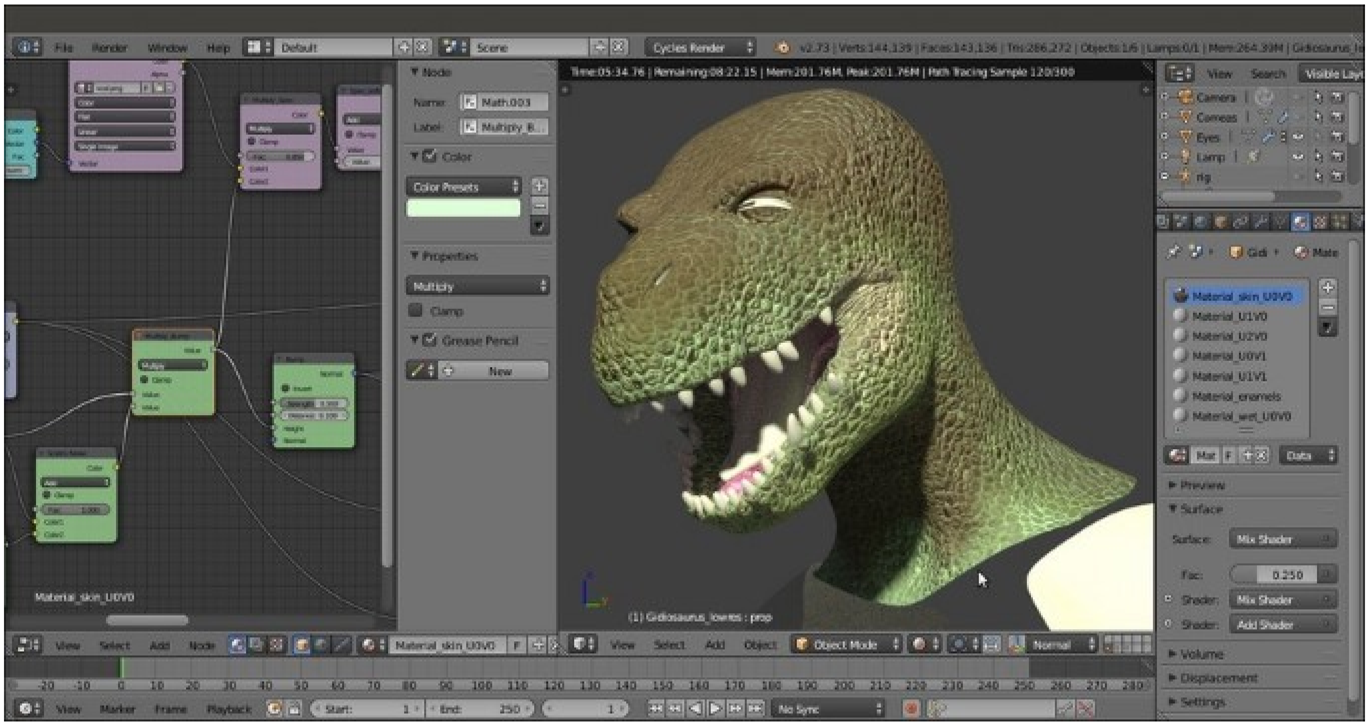
Adding some noise to the bump pattern part 3

47. Add a **MixRGB** node (*Shift* + *A* | **Color** | **MixRGB**) and paste it between the **VCOL** node and the **Spec_soften** node; label it as **Multiply_Spec**, set the **Blend Type** to **Multiply** and the **Fac** value to **0.850**; connect the output of the **Multiply_Bump** node to the **Color2** input socket of the **Multiply_Spec** node:



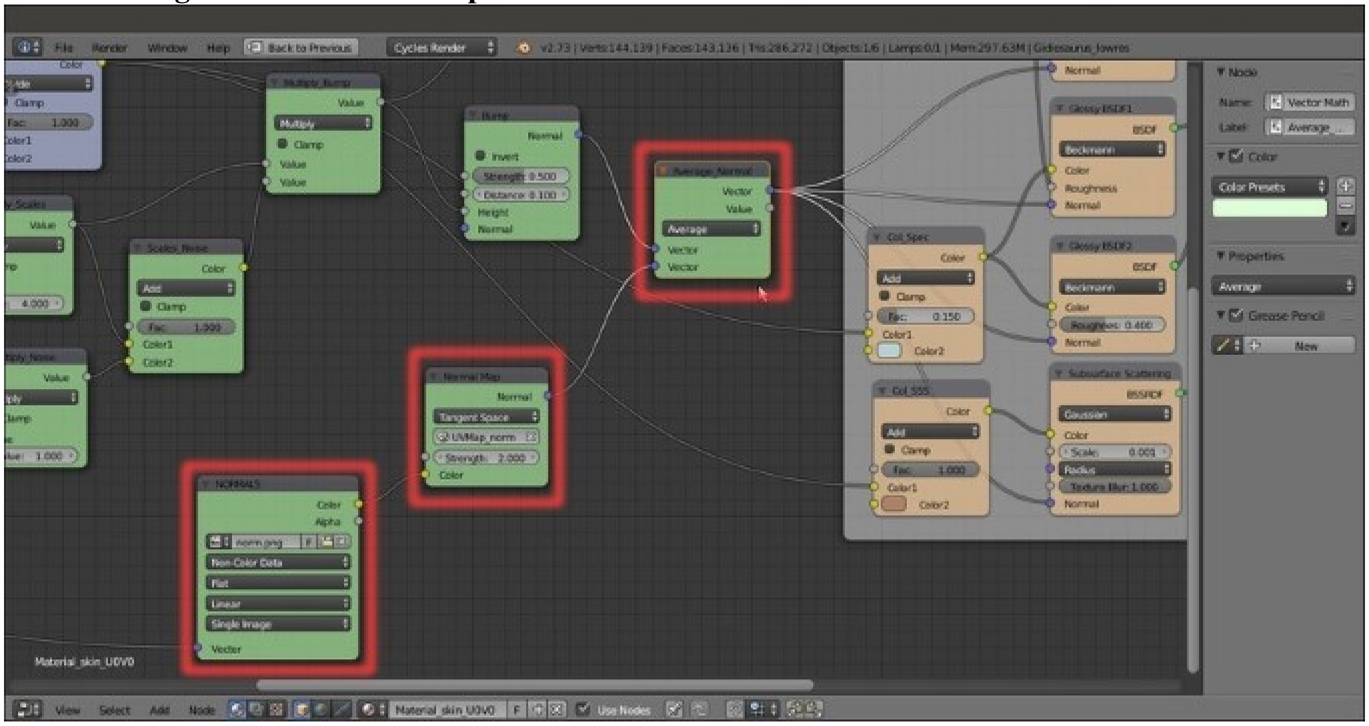
Modulating the specularity with the aid of the bump pattern output

The overall bump effect is almost completed:



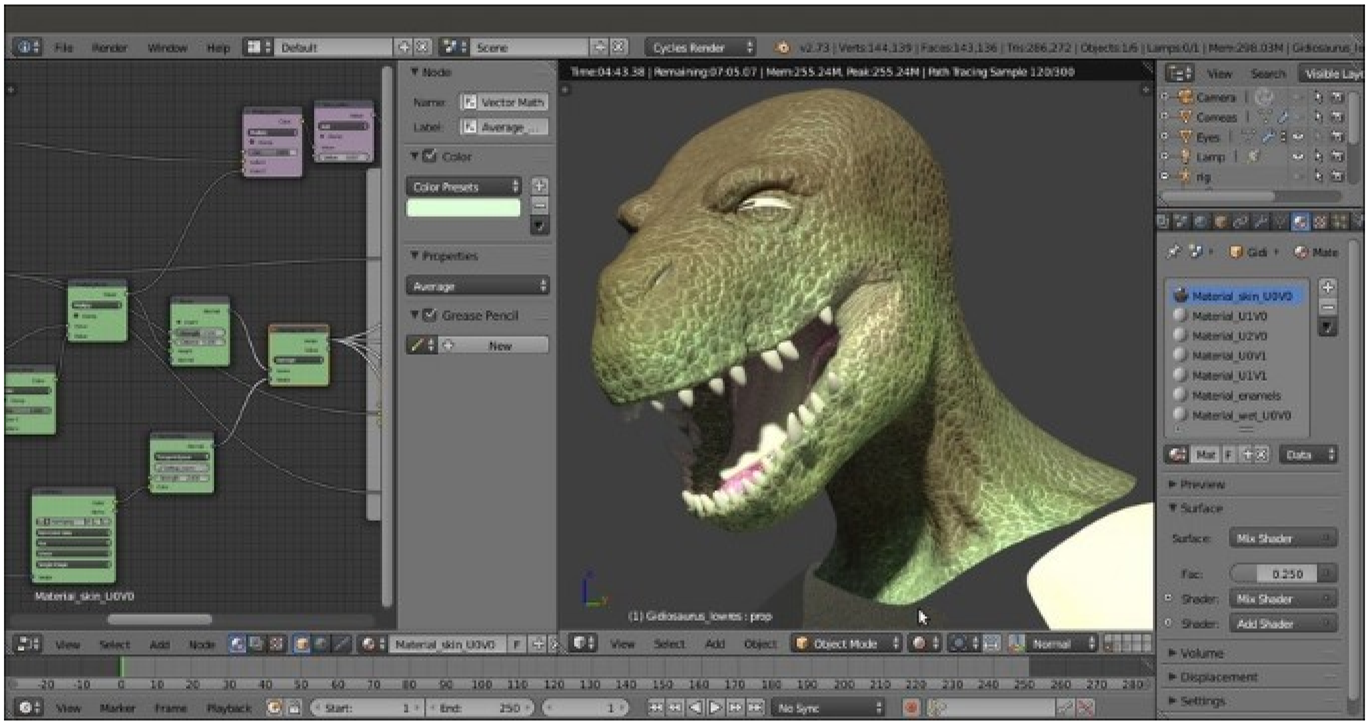
What is still missing now is the normal map we obtained from the sculpted **Gidiosaurus** mesh in Chapter 11, *Refining the Textures*.

48. Add a new **Image Texture** node (*Shift* + *A* | **Texture** | **Image Texture**) and a **Normal Map** node (*Shift* + *A* | **Vector** | **Normal Map**). Label the **Image Texture** node as **NORMALS**, then connect the **Vector** output of the **Attribute_UV2** node to the **Vector** input socket of the **NORMALS** node.
49. Connect the **Color** output of the **NORMALS** node to the **Color** input socket of the **Normal Map** node, then click on the **Open** button on the **NORMALS** node, browse to the `textures` folder and load the image `norm.png`. Set the **Color Space** of the **NORMALS** node to **Non-Color Data** and click on the empty slot in the **Normal Map** node to select the **UVMap_norm** coordinates layer.
50. Add a **Vector Math** node (*Shift* + *A* | **Converter** | **Vector Math**), label it as **Average_Normals** and paste it right after the **Bump** node; connect the output of the **Normal Map** node to the second **Value** input socket of the **Average_Normals** node.
51. Set the **Operation** of the **Average_Normals** node to **Average** and connect its **Vector** output to the **Vector** input sockets of the **Diffuse BSDF**, **Glossy BSDF1**, **Glossy BSDF2**, and **Subsurface Scattering** nodes.
52. Set the **Strength** of the **Normal Map** to **2.000**:



Adding the normal map output to the bump pattern

Finally we have completed the first skin material!

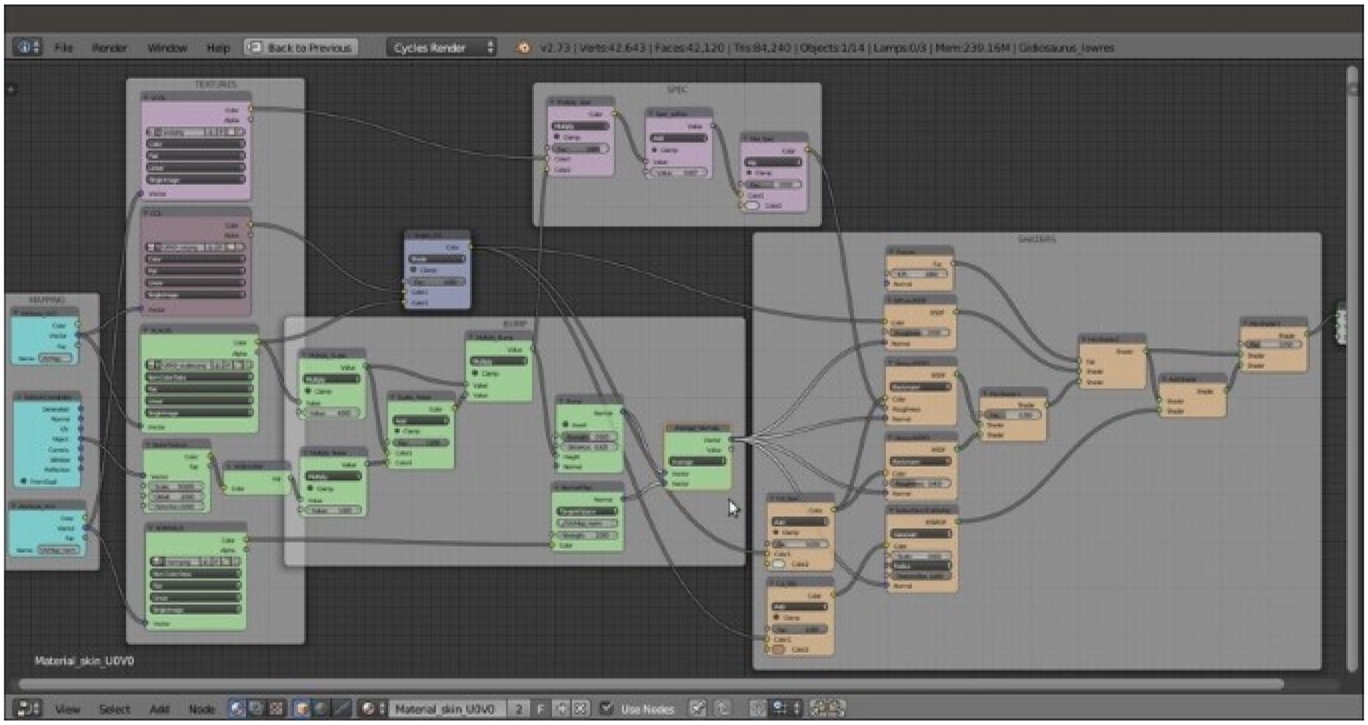


The completed Material_skin_U0V0

53. Save the file as `Gidiosaurus_skin_Cycles.blend`.

How it works…

This material can at first glance appear a bit complex, but actually the design behind it is quite simple as you can see in the following screenshot, where each component has been visually grouped by colors and frames (open the provided `Gidiosaurus_skin_Cycles_01.blend` file to have a better look):



The total skin material network

- From step 1 to step 18 we built the **SHADERS** part of the material, that is, the combination of the diffuse with the glossy component and the addition of the subsurface scattering effect.
- Note that the glossy component (the *specularity*) is obtained by mixing **two** glossy shaders with different roughness values; by setting the factor value of the **Mix Shader1** node to **0.350**, we give prevalence to the **Glossy BSDF1** node effect, which is to the node connected to the first top **Shader** input socket.
- Also, we added the subsurface scattering effect by the **Add Shader** node, and to further tweak the blending of the effect with the rest of the shader, we added the **Mix Shader3** node, to give prevalence to the output of the **Mix Shader2** node (that is the output of the diffuse plus the glossy components).
- From step 19 to step 24 we saw some not mandatory but useful tips for assigning colors to the nodes, in order to visually distinguish and/or group them and make the whole material network more easily readable.
- At step 25 we started to add the textures, first the diffuse color one and then the grayscale scales image that we used here to add details to the coloration (and later for the bump effect). By mixing the scales with the diffuse color through the **MixRGB** node set to a **Divide** blend type, we automatically obtained a scales pattern on the skin itself.
- From step 30 to step 33 we tweaked the diffuse color map to also affect the glossy and the subsurface scattering components, but with different hues.
- Note that at step 34 we used an **Attribute** node to set the UV coordinates layer to be used for the mapping of the textures. It would have been unnecessary in this case, with the **UVMap** coordinates layer being the first one and therefore the default one. Cycles, in fact, in the case of

image textures, automatically uses any existing UV coordinates layer. But, because later we also used a different UV coordinates layer, it was better to specify it.

- From step 35 to step 38 we improved the glossiness effect of the skin, by using the output of the `vcol.png` image we had previously baked and tweaked through the nodes inside the **SPEC** frame.
- From step 39 to step 47 we built the **BUMP** effect, by using the output of the **SCALES** image texture added through a **MixRGB** node to the output of a procedural **Noise Texture**. The **RGB to BW** node simply converts the colored output of the procedural noise to a grayscale output (and if you think we could have used the **Fac** output instead, well, it's not the same thing), and the **Multiply_Scales** and **Multiply_Noise** nodes set the strength of the outputs before the adding process. Through the **Multiply_Bump** node we also added the grayscale output of the combined bump to the glossy component.
- From step 48 to step 52 we also added the effect of the normal map we baked from the sculpted high resolution **Gidiosaurus** mesh to the bump pattern. The normal map is averaged, through the **Vector Math** node, with the bump output. Because of this averaging, the strength value of the normal map had to be set to double (**2.000**) to have full effect.

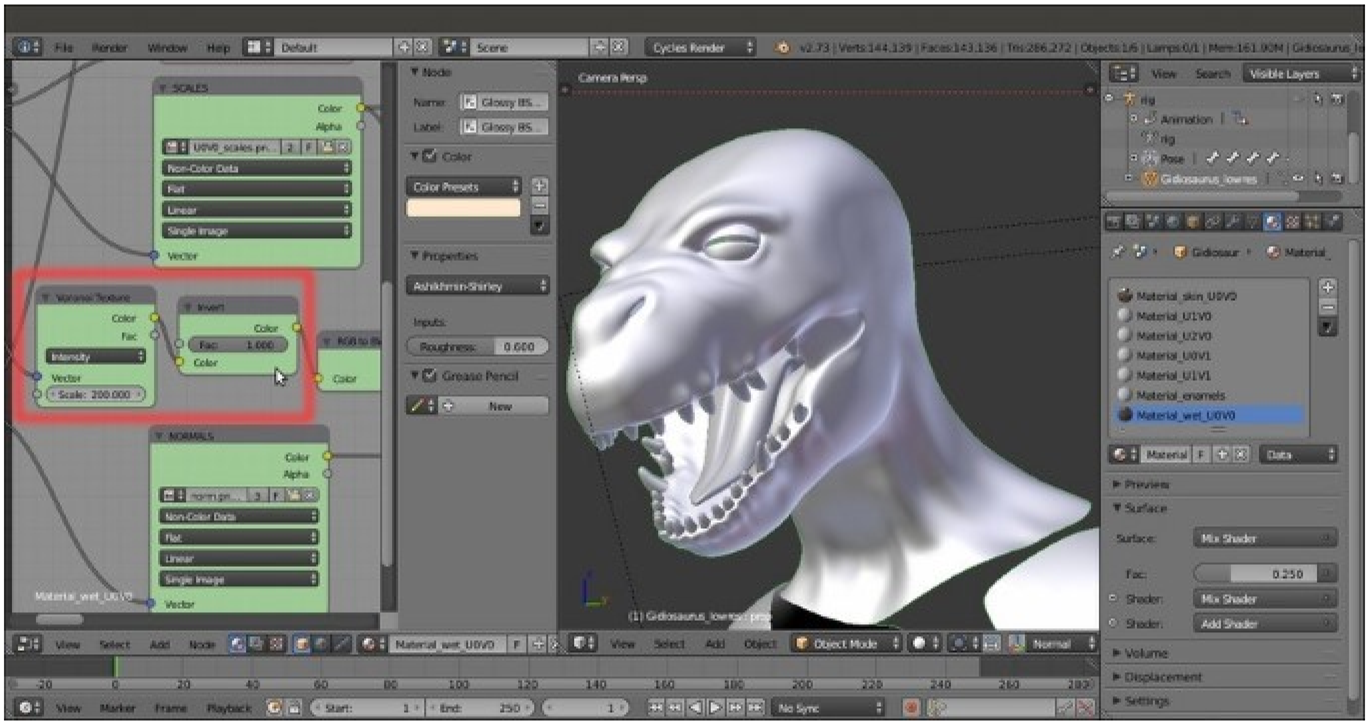
There's more…

Still focusing on the character's **head**, there is a material we can obtain from the skin material with some modification, the material for the wet parts of the character's skin (inner **eyelids**, **tongue**, inner **nostrils**).

Going on from the previously saved file:

1. If you think this is the case, especially if your computer (like mine) isn't very powerful, temporarily disable the **Rendered** preview by moving the mouse cursor inside the 3D viewport and pressing *Shift* + *Z*.
2. In the **Material** window, click on the `Material_wet_U0V0` material to select it.
3. Put the mouse pointer inside the **Node Editor** window, select the default two nodes already assigned to the material and delete them by pressing the *X* key.
4. Now, in the **Material** window, re-select the `Material_skin_U0V0`; put the mouse in the **Node Editor** window, press *A* twice to select everything, and press *Ctrl* + *C*.
5. Re-select the `Material_wet_U0V0`, put the mouse pointer inside the empty **Node Editor** window and press *Ctrl* + *V* to paste the copied material nodes.

   Now we have copied the nodes of the skin material to the material assigned to the parts that need to appear wet; it's enough now to tweak this material a bit to modify the bump pattern and the glossiness:
6. In the **Node Editor**, zoom to the **Noise Texture** node inside the **BUMP** frame; left-click on it to select it and then press the *X* key to delete it.
7. Press *Shift* + *A* and add a **Voronoi Texture** node (*Shift* + *A* | **Texture** | **Voronoi Texture**); left-click on the node and, by keeping the mouse button pressed, move the node a little bit on the frame, so it should automatically be parented to it.
8. Connect the **Object** output of the **Texture Coordinate** node to the **Vector** input socket of the **Voronoi Texture** node and the **Color** output of this latter node to the **RGB to BW** node input socket; set the **Voronoi Scale** to **200.000**.
9. Add an **Invert** node (*Shift* + *A* | **Color** | **Invert**) and paste it between the **Voronoi Texture** and the **RGB to BW** nodes:

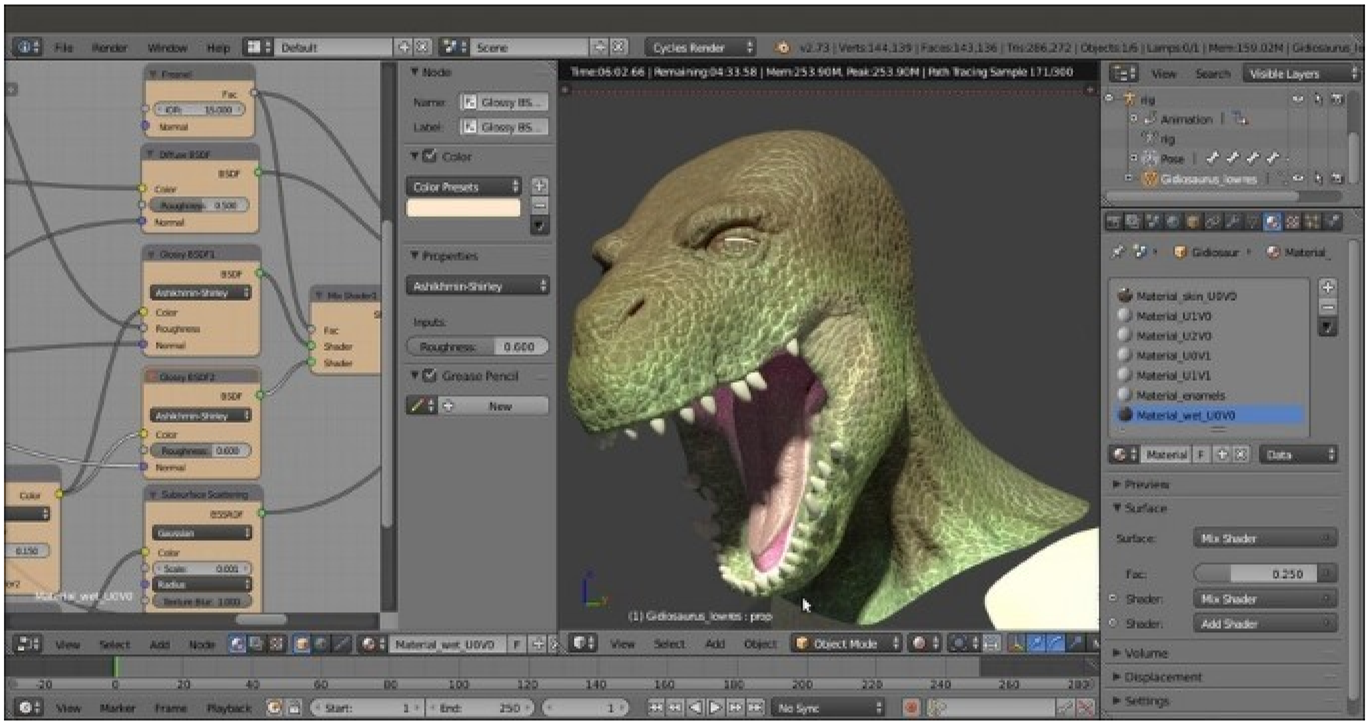


The different texture nodes of the "Material_wet_U0V0"

10. Scroll the **Node Editor** window a bit to the right to find the **Multiply_Noise** node: change the label to **Multiply_Voronoi** and the second **Value** to **0.025**.
11. Find the **Scales_Col** node and change **Blend Type** from **Divide** to **Multiply**.
12. Now go to the **SHADERS** frame; change the **IOR** value of the **Fresnel** node to **15.000** and connect its output to the **Fac** input socket of the **Mix Shader1** node; change the **Distribution** of both the **Glossy BSDF1** and **Glossy BSDF2** nodes to **Ashikhmin-Shirley** and set the **Roughness** of the **Glossy BSDF2** node to **0.600**.

    We substituted the **Noise Texture** node with a **Voronoi Texture** node to give a kind of organic look to the surface of the **tongue** of the creature.

    In the following screenshot, we can see the result of the wet material; note that for the occasion I opened the mouth wide, to make the inside more visible:



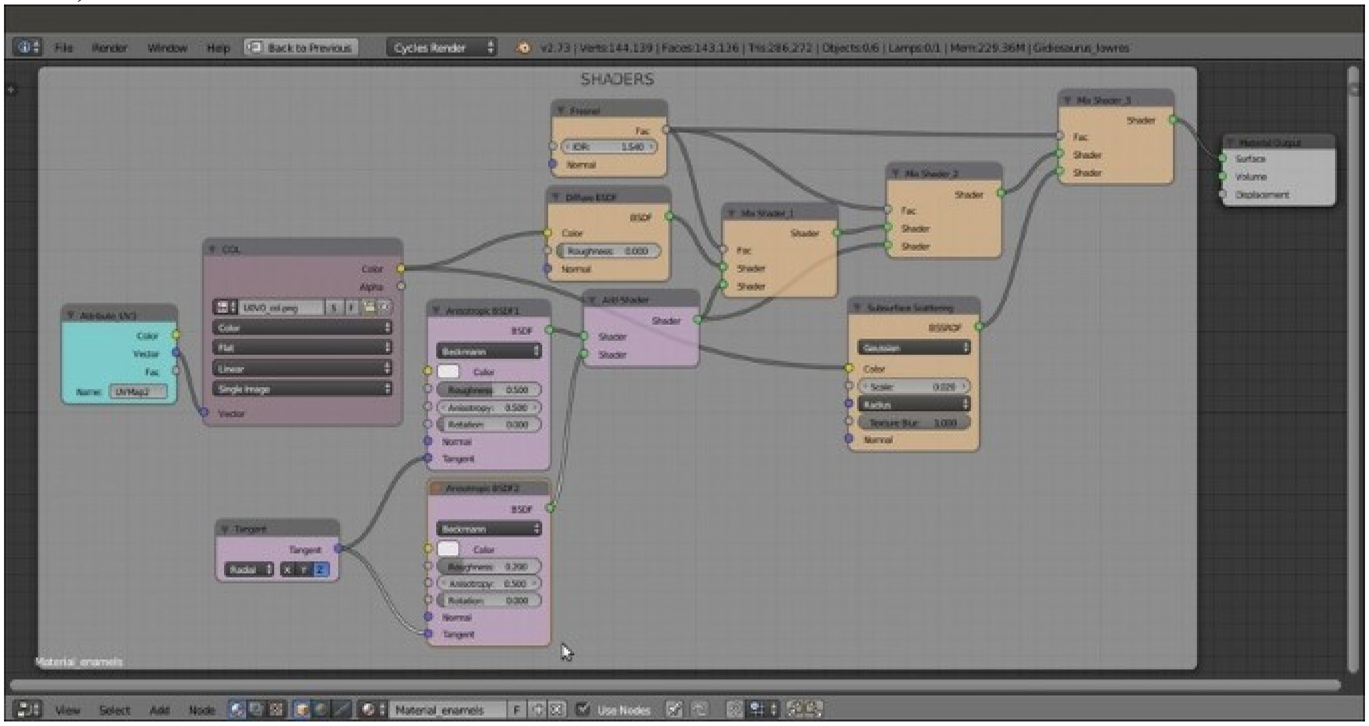
The rendered wet material

One more material we are going to create in this section of the recipe is the **Material_enamels** for **teeth** and **talons**; in this case, we just need mostly the **SHADERS** frame's nodes with the single contribution of the color image texture `U0V0_col.png`, here using the **UVMap2** coordinates layer to avoid having to create **5** different materials for the **talons** alone (originally distributed in different tiles). By the way, nothing is stopping you from creating several talon materials, if you prefer.

13. Again, select, copy and paste the skin material to the enamels material slot through the **Node Editor** window, as we have already done in steps 3, 4 and 5.
14. This time, just delete the unnecessary nodes, in short keeping only the **Attribute** node, the **COL** node and the **SHADERS** frame with its parented nodes.
15. Change the UV coordinates layer in the **Name** slot of the **Attribute** node to **UVMap2** (and the label to **Attribute_UV3**). Lower the **Roughness** value of the **Diffuse BSDF** node to **0.000**.
16. Go to the **SHADERS** frame; select and delete the **Col_Spec** and **Col_SSS** nodes, then connect the **Color** output of the **COL** node also to the **Color** input socket of the **Subsurface Scattering** node.
17. Select and delete the **Glossy BSDF1** and the **Glossy BSDF2** nodes.
18. Add 2 **Anisotropic BSDF** shader nodes (*Shift* + *A* | **Shader** | **Anisotropic BSDF**), a **Tangent** node (*Shift* + *A* | **Input** | **Tangent**) and detach the **Add Shader** node from the **Mix Shader3** node.
19. Label the two **Anisotropic BSDF** shader nodes as **Anisotropic BSDF1** and **Anisotropic BSDF2** and connect them to the two **Shader** input sockets of the **Add Shader** node. Connect

the output of the **Tangent** node to the **Tangent** input sockets of the two **Anisotropic** shader nodes.

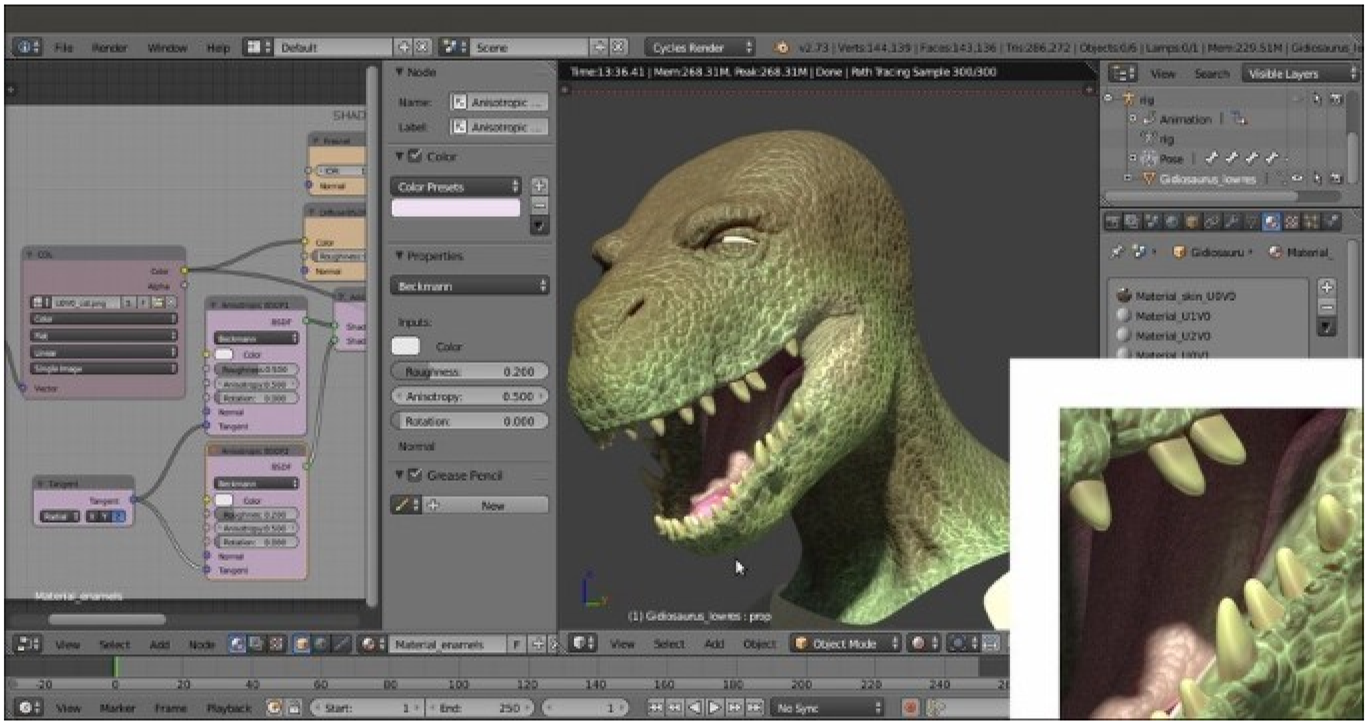
20. Set the **Tangent** of the **Tangent** node to **Z**. Set the **Anisotropy** of both the **Anisotropic** nodes to **0.500**, the **Roughness** of the **Anisotropic BSDF1** node to **0.500** and the **Roughness** of the **Anisotropic BSDF2** node to **0.200**.
21. Connect the **Add Shader** output to both the second **Shader** input sockets of the **Mix Shader1** and **Mix Shader2** nodes.
22. Set the **IOR** value of the **Fresnel** node to **1.540** and connect the **Fresnel** output to the **Fac** input sockets of the **Mix Shader1**, **Mix Shader2**, and **Mix Shader3** nodes.
23. Connect the output of the **Diffuse BSDF** shader node to the first **Shader** input socket of the **Mix Shader1** node, then connect the output of the **Mix Shader1** node to the first **Shader** input socket of the **Mix Shader2** node.
24. Connect the output of the **Subsurface Scattering** node to the second **Shader** input socket of the **Mix Shader3** node.
25. In the **Subsurface Scattering** node, change the **Scale** to **0.020** and the **Radius** to **R 1.000**, **G 0.400**, **B 0.100**.



The "Material_enamels" network

26. Save the file.

Thanks to the two **Anisotropic** shaders with their different roughness values, we obtained a nice specularity effect along the length of the **teeth** (and therefore also of the **talons**):



The rendered preview of the teeth (and talons) shader

See also

- Shameless self-promotion—one other cookbook, published by Packt Publishing, explaining the logic behind Cycles materials and textures and with several material recipes (https://www.packtpub.com/hardware-and-creative/blender-cycles-materials-and-textures-cookbook-third-edition)
- The online documentation (http://wiki.blender.org/index.php/Doc:2.6/Manual/Render/Cycles/Nodes)

Making a node group of the skin shader to reuse it

Once we are satisfied with the reptile skin shader created for the character's **head**, we can copy it to the other parts of the **body**, that is to the other material slots, and then apply the necessary modifications. Those, in this case, just consist of different color and scales image textures.

This means that all the other shader parts can be reused as they are. In this recipe, in fact, we are going to make a node group of these parts so as to easily re-use the shader for the other materials slots.

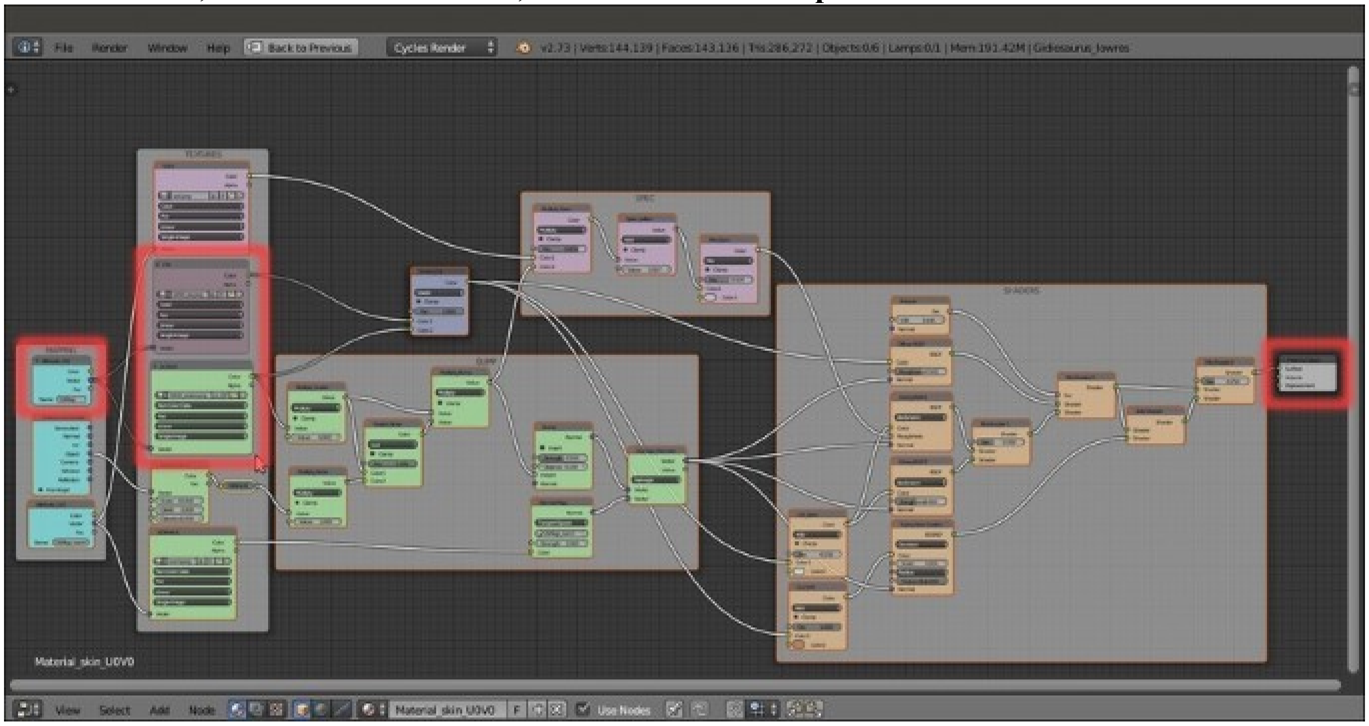
Getting ready

Just start Blender and re-open the previously saved `Gidiosaurus_skin_Cycles_01.blend` file.

How to do it…

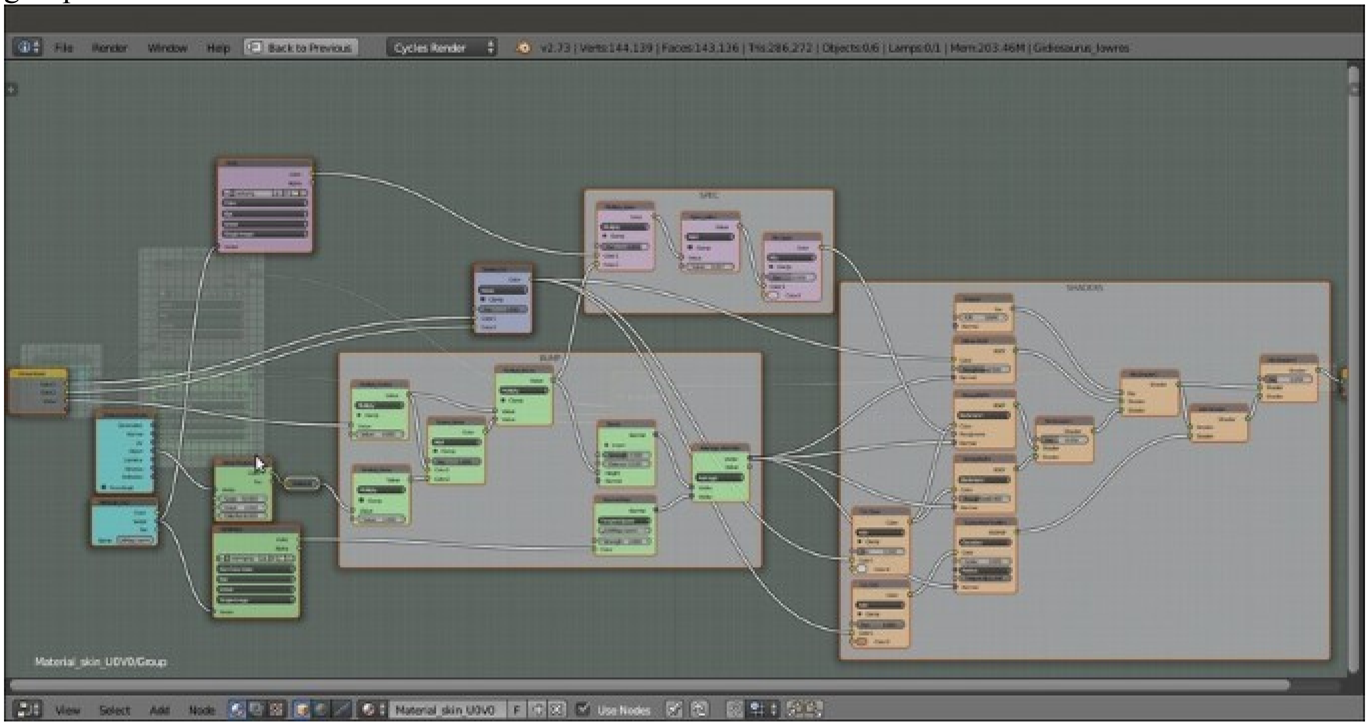
Let's start to create our skin node group:

1. In the **Material** window, select the slot of the `Material_skin_U0V0`.
2. Put the mouse pointer in the **Node Editor** window, press the *B* key and left-click to box-select all the nodes with their respective frames. Then, press the *Shift* key and right-click (twice for each one) to deselect the **Attribute_UV1** node, the **MAPPING** frame, the **COL** node, the **SCALES** node, the **TEXTURES** frame, and the **Material Output** node:



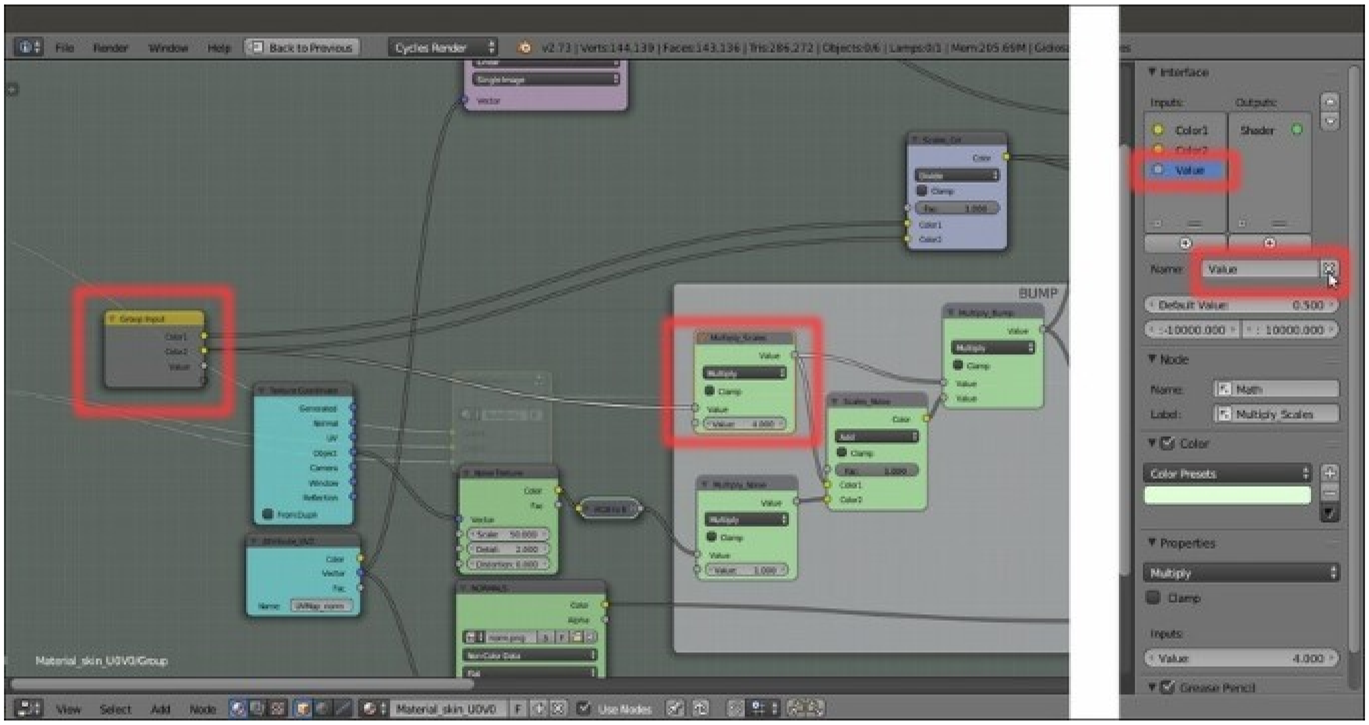
The box-selected nodes and the highlighted deselected ones

3. Press *Ctrl* + *G* to make a node group of the selected nodes; automatically you are inside the group in **Edit Mode**:



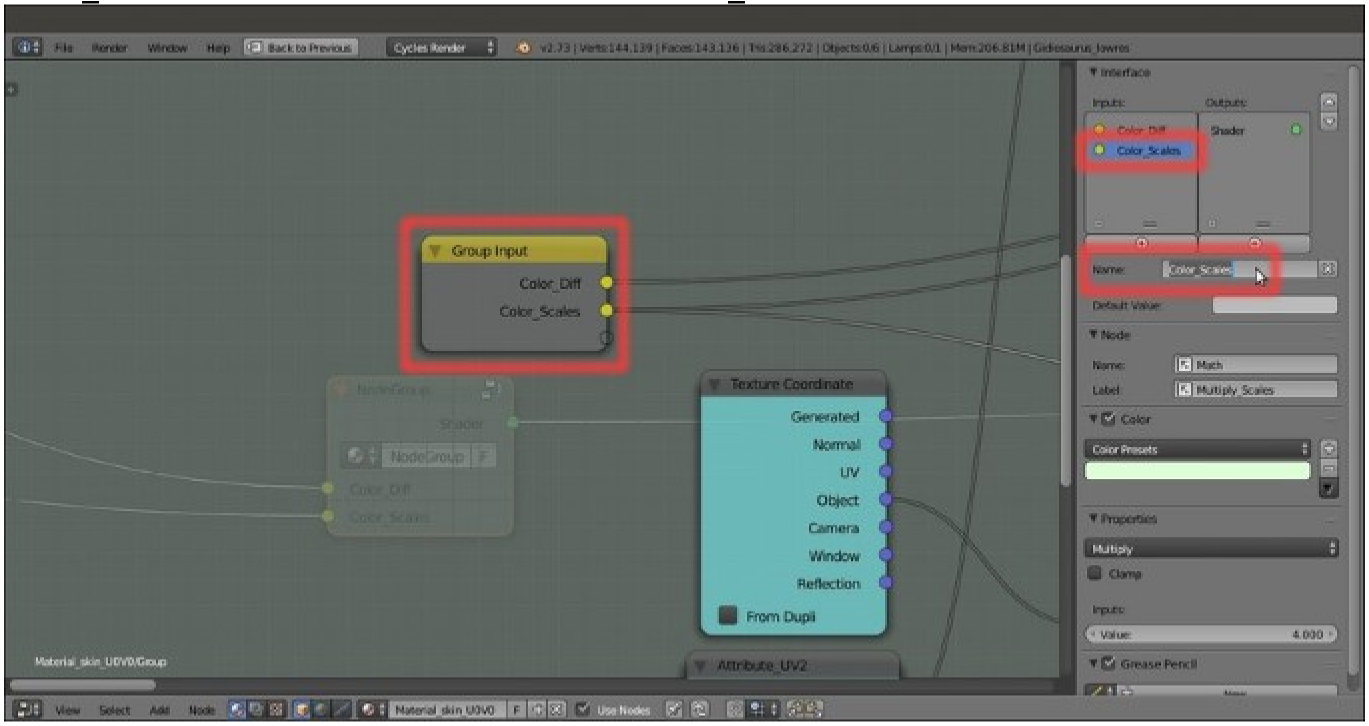
Inside the node group in Edit Mode

4. Click on the **Group Input** node to select it and zoom in on it, then press *N* to call the **Properties** sidepanel. Connect the **Color2** output socket to the first **Value** input socket of the **Multiply_Scales** node, replacing the connection coming from the **Value** output.
5. Go to the **Properties** sidepanel and in the **Interface** subpanel, click on the **Value** item inside the little **Inputs** window; then go down to the **Name** slot and click on the **X** icon button to delete the socket from the **Group Input** node:



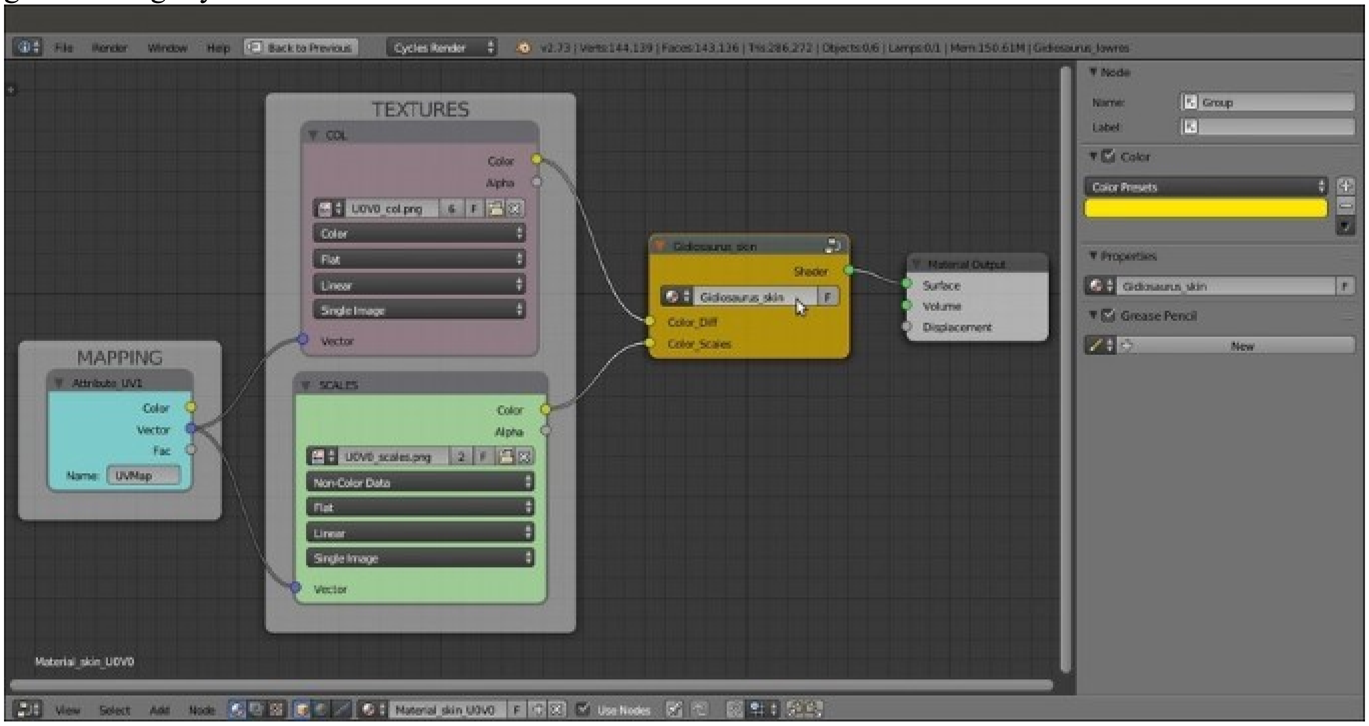
Tweaking the node group input socket connections

6. Still in the **Name** slot in the **Interface** subpanel, select the **Color1** item and rename it **Color_Diff**. Select the **Color2** item and rename it **Color_Scales**:



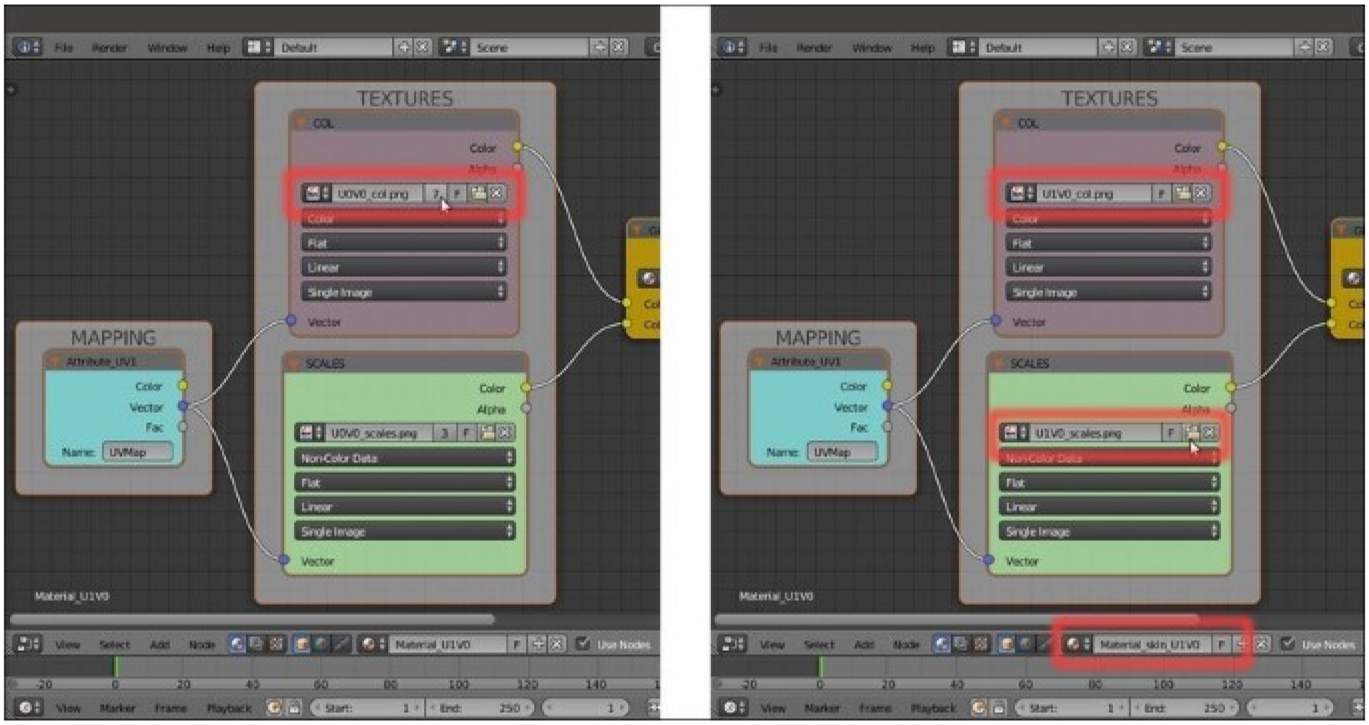
Renaming the node group input sockets

7. Press *Tab* to exit out of **Edit Mode** and close the node group; rename it **Gidiosaurus_skin** and give it a bright yellow color:



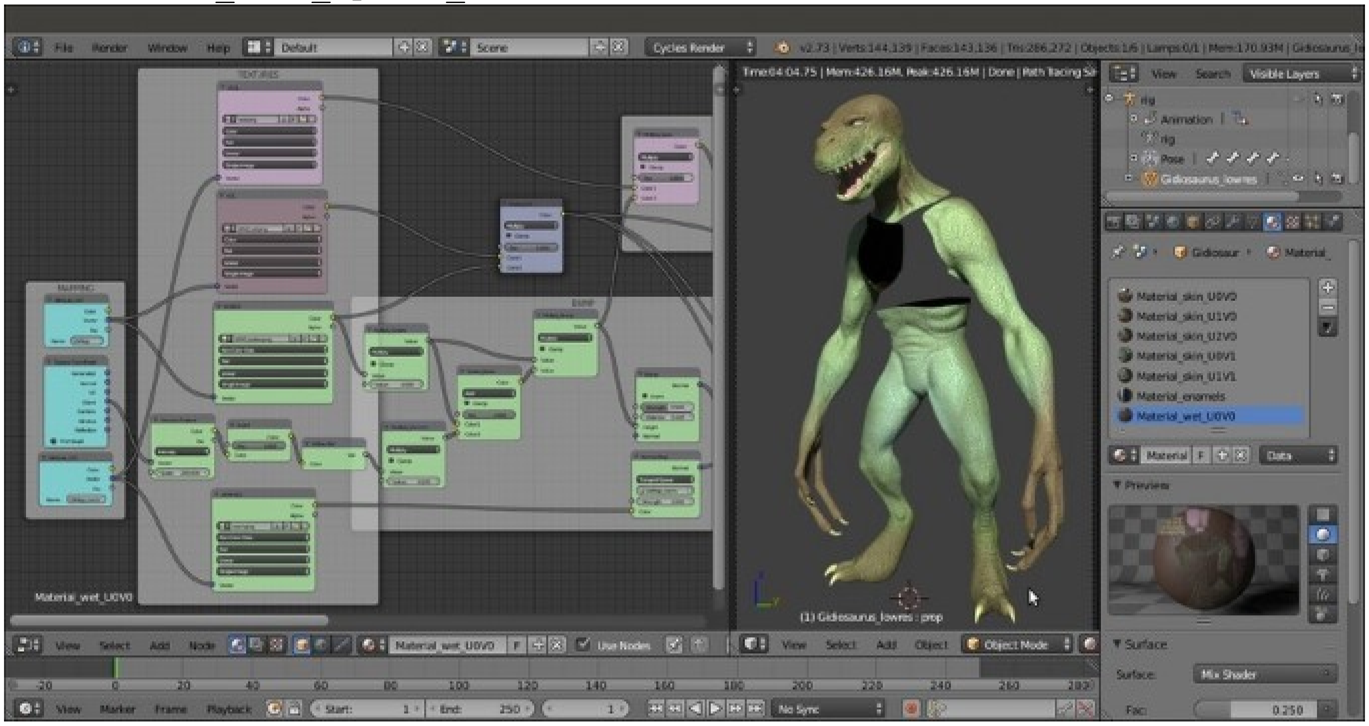
The "Gidiosaurus_skin" node group

8. Now select everything by pressing the *A* key twice and then press *Ctrl* + *C*.
9. In the **Material** window, select the `Material_U1V0`, then click on the **Use Nodes** button in the **Surface** subpanel.
10. Put the mouse pointer in the **Node Editor** window and delete the already selected default nodes (the **Diffuse BSDF** connected to the **Material Output** nodes), then press *Ctrl* + *V* to paste all the copied nodes inside the window.
11. Zoom on the **COL** and **SCALES** nodes. Click on the numbered button to the right side of the texture name to make them single users, then click on the folder icon buttons to browse to the `textures` folder and load the proper images according to the material name, that is, the `U1V0_col.png` and `U1V0_scales.png` image textures.
12. Rename the material as `Material_skin_U1V0`:



Making the copied textures single users and loading the right image textures for the "Material_skin_U1V0"

13. Repeat step 8 to step 12 for the other remaining **3** material slots and then save the file as `Gidiosaurus_skin_Cycles_02.blend`.



The "Material_wet_U0V0" network and the completed Gidiosaurus shading in the rendered preview

How it works…

Of course, it wasn't mandatory to make a group of the skin shader to reuse it for the other material slots; we could just have selected, copied and pasted all the nodes and frames as they were at the end of the previous recipe.

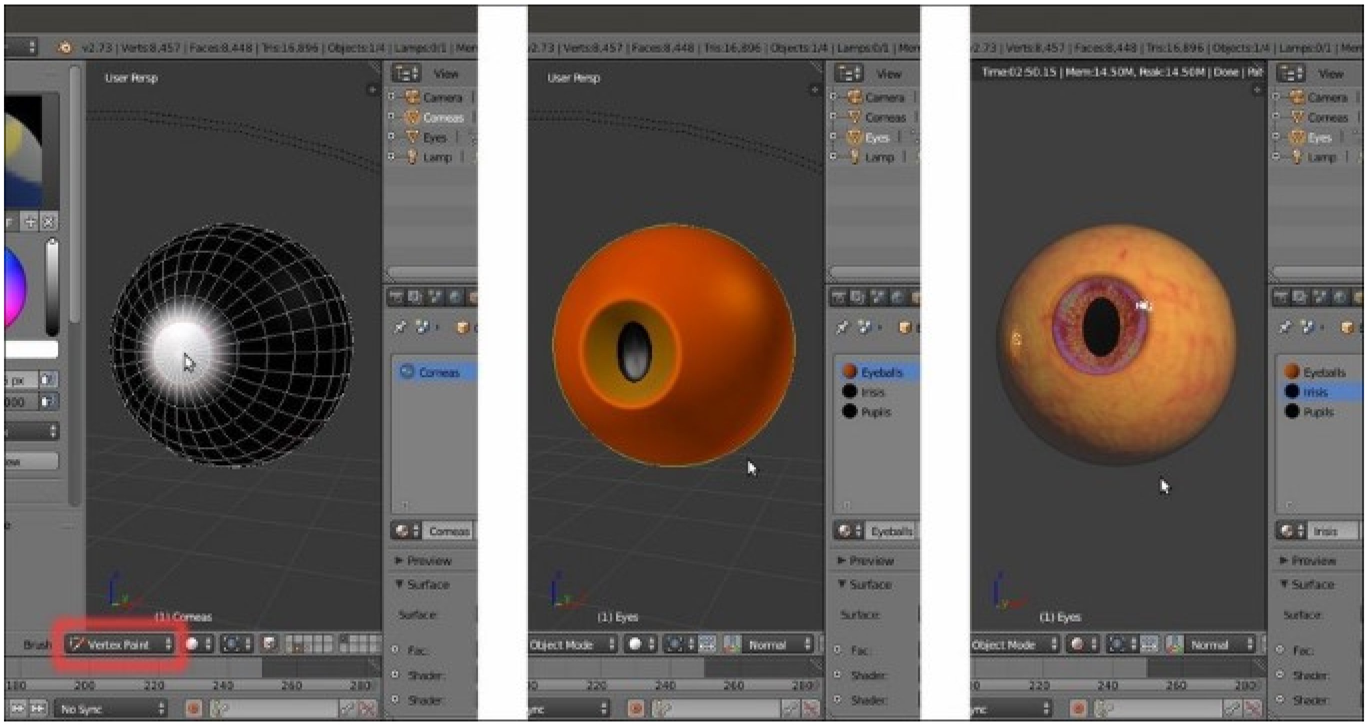
The (quite big) advantage in having a node group instanced in different materials is that if you need to change something in the internal network, you don't have to repeat the modifications in the node group of each material. It's enough to do it in **Edit Mode** in one of the instances, and all the internal modifications will be reflected in all the instances of the node group used by the other materials.

Building the eyes' shaders in Cycles

The character's **eyes** are made up of two **UV Spheres**, the **Corneas** and the **Eyes** objects: the bigger **Corneas** one enveloping a smaller **Eyes** sphere, which in turn is made up of **three** parts: the **eyeballs**, the **irises**, and the **pupils**.

The **Corneas** sphere was first painted with a totally black **Vertex Color** layer, then painted with a white color only to the vertices corresponding to the front crystalline lens.

The **Eyes** sphere has **three** different materials assigned to the **three** different parts:



The Corneas object in Vertex Paint mode, the Eyes object with its three materials and the Rendered preview of the textured objects together

Getting ready

Start Blender and open the `Gidiosaurus_skin_Cycle_02.blend` file; save it as `Gidiosaurus_shaders_Cycles.blend`.

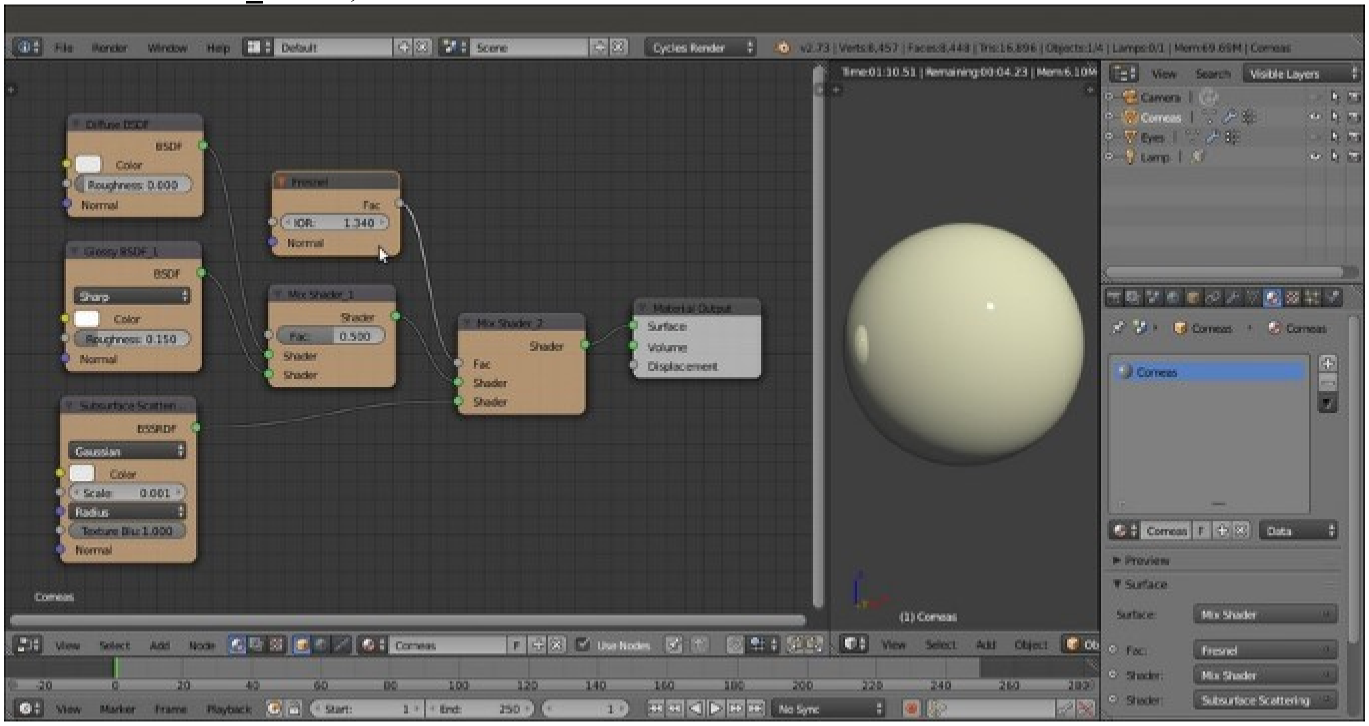
1. Enable only the **6th** and the **12th** scene layer, in order to have visible only the **Corneas**, the **Eyes** and the **Lamp** objects (actually the **Camera** is also on the **6th** scene layer, but it's hidden and at the moment we don't need it).
2. Zoom the 3D view onto the **Corneas** and **Eyes** objects, and press *Shift* + *Z* to start the **Rendered** preview.

3. In the **Outliner**, disable the *Restrict view-port visibility* button of the **Eyes** object to hide it.
4. Select the **Corneas** object and go to the **Material** window.

How to do it…

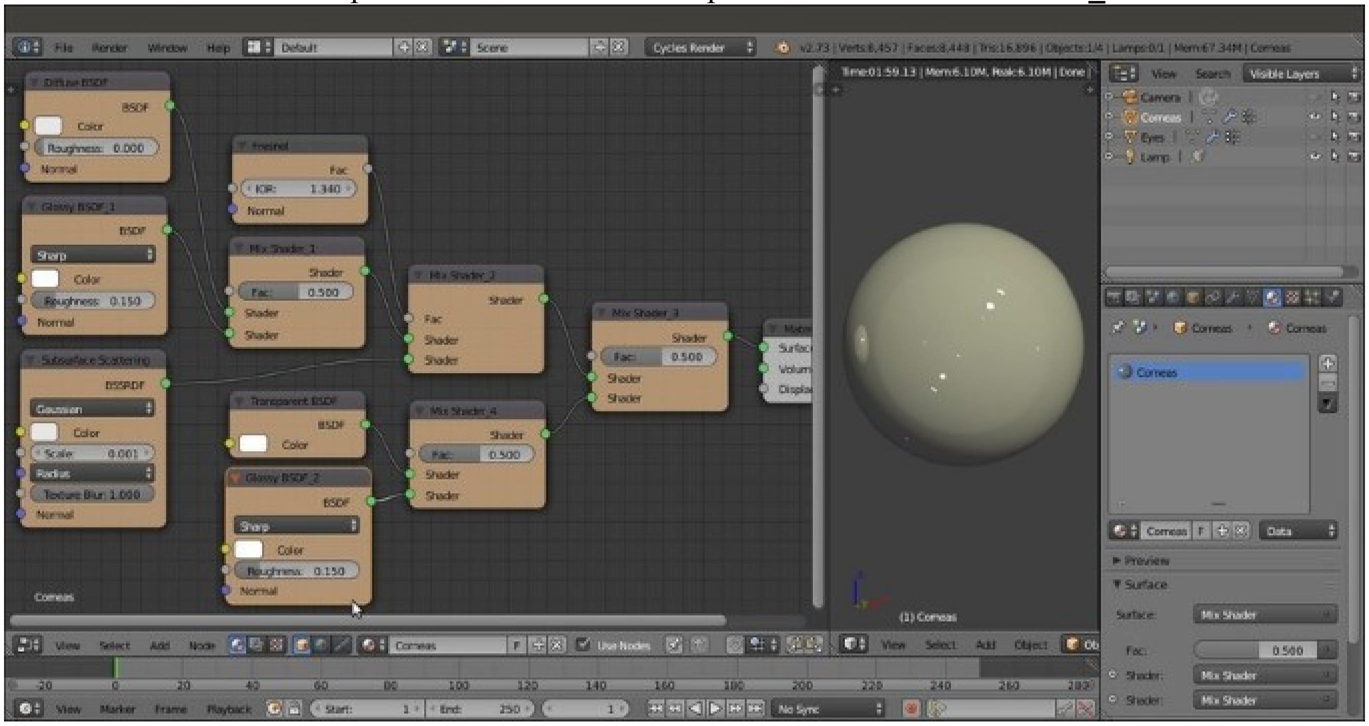
So, let's start with the **Corneas** material, first:

1. In the **Material** window, click on the **New** button in the **Surface** subpanel. Rename the material as `Corneas`.
2. In the **Material** window, switch the **Diffuse BSDF** shader node with a **Mix Shader** node (label it as **Mix Shader_1**). In the first **Shader** slot, select a **Diffuse BSDF** shader node and in the second one select a **Glossy BSDF** shader (label it as **Glossy BSDF1**).
3. Go to the **Node Editor** window and set the **Roughness** of the **Glossy BSDF1** shader to **0.150**, the **Color** to pure white and the **Distribution** to **Sharp**.
4. Select the **Mix Shader_1** node and press *Shift* + *D* to duplicate it. Label the duplicate as **Mix Shader_2**, then add a **Subsurface Scattering** node (*Shift* + *A* | **Shader** | **Subsurface Scattering**). Connect the output of the **Mix Shader_1** node to the first **Shader** input socket of the **Mix Shader_2** node and the output of the **Subsurface Scattering** shader node to the second **Shader** input socket.
5. Change the **Subsurface Scattering** falloff from **Cubic** to **Gaussian**, set the **Scale** to **0.001** and the **Radius** to **R 9.436**, **G 3.348**, **B 1.790**.
6. Add a **Fresnel** node (*Shift* + *A* | **Input** | **Fresnel**) and connect its output to the **Fac** input socket of the **Mix Shader_2** node; set the **IOR** to **1.340**:



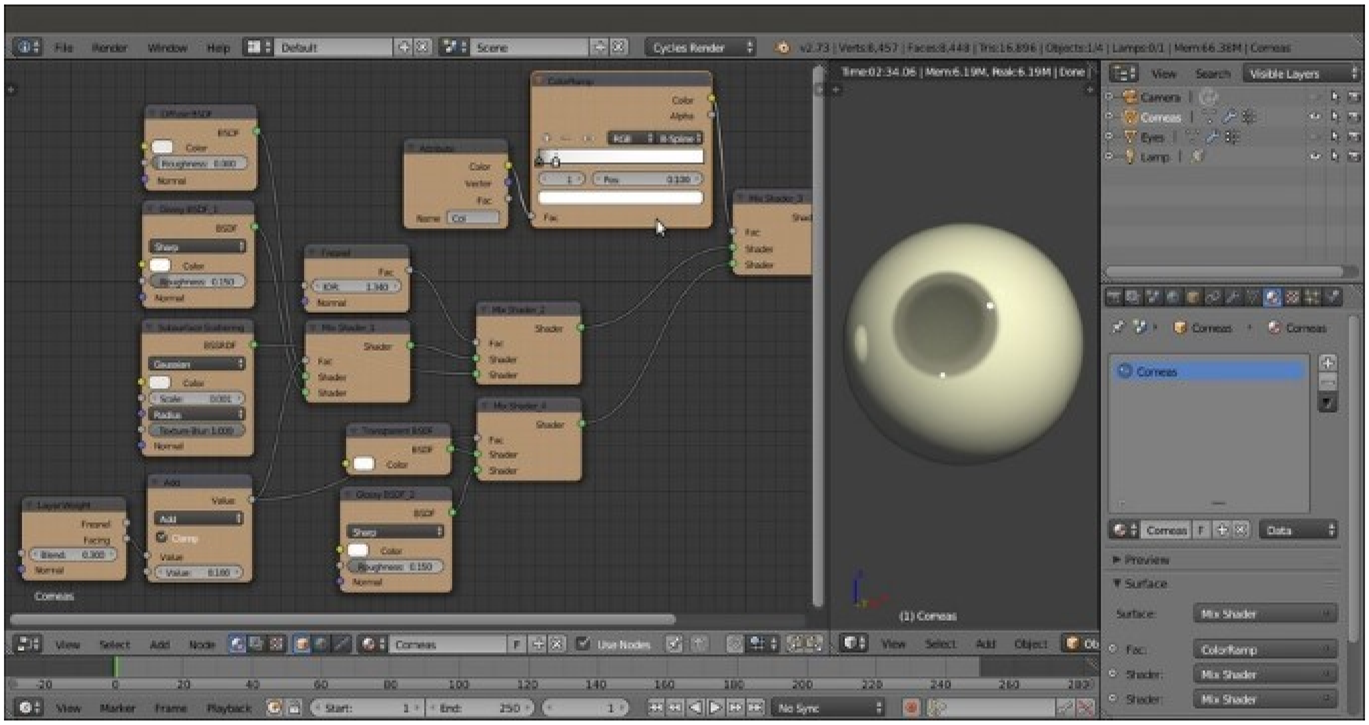
The basic starting "Corneas" shader

7. Add a new **Mix Shader** node (*Shift* + *A* | **Shader** | **Mix Shader**), label it as **Mix Shader_3** and paste it between the **Mix Shader_2** and the **Material Output** node.
8. Add a new **Mix Shader** node (*Shift* + *A* | **Shader** | **Mix Shader**), label it as **Mix Shader_4** and connect its output to the second **Shader** input socket of the **Mix Shader_3** node.
9. Add a **Transparent BSDF** shader (*Shift* + *A* | **Shader** | **Transparent BSDF**) and connect it to the first **Shader** input socket of the **Mix Shader_4** node.
10. Select and press *Shift* + *D* to duplicate the **Glossy BSDF1** node; label the duplicate as **Glossy BSDF2** and connect its output to the second **Shader** input socket of the **Mix Shader_4** node:



The "Corneas" shader with the added transparency nodes

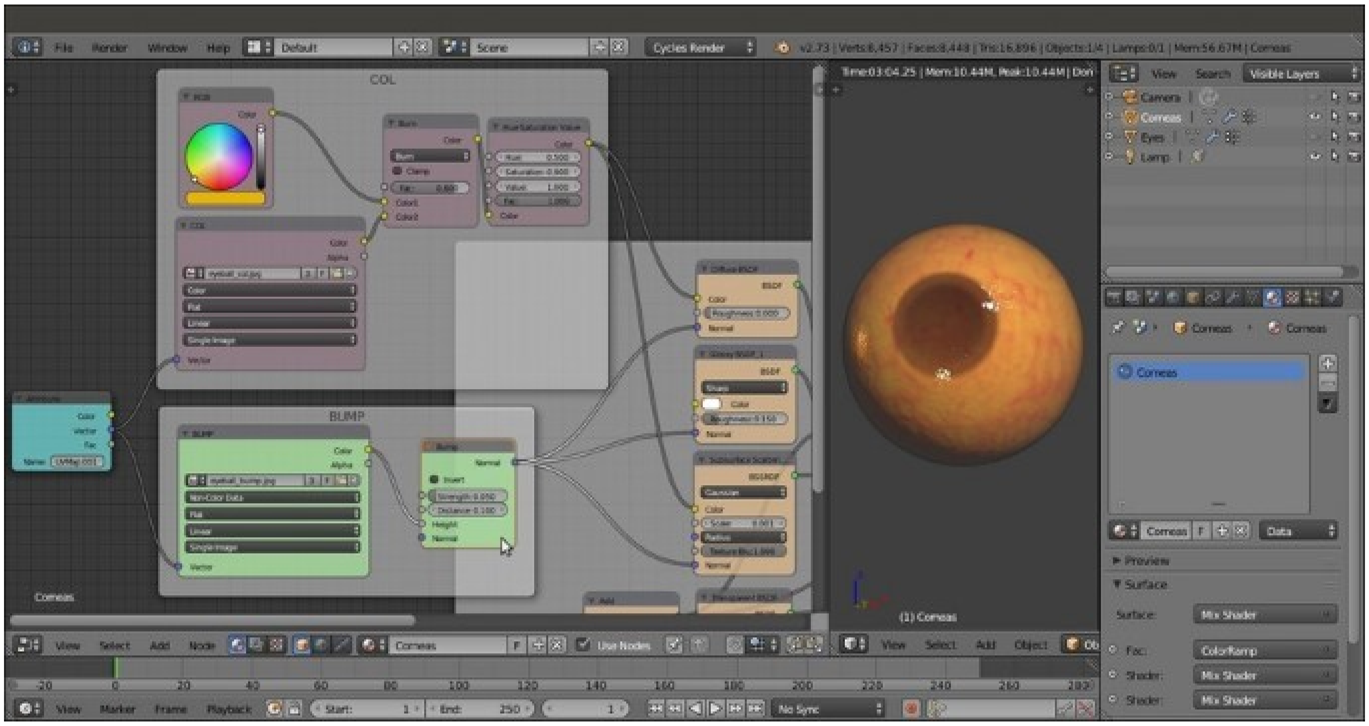
11. Add a **Layer Weight** node (*Shift* + *A* | **Input** | **Layer Weight**) and a **Math** node (*Shift* + *A* | **Converter** | **Math**); set the **Blend** factor of the **Layer Weight** to **0.300** and connect its **Facing** output to the first **Value** input socket of the **Math** node, then set the second **Value** to **0.100** and check the **Clamp** item.
12. Connect the **Math** (labeled as **Add**) output to the **Fac** input sockets of the **Mix Shader_1** and **Mix Shader_4** nodes.
13. Add an **Attribute** node (*Shift* + *A* | **Input** | **Attribute**) and a **ColorRamp** node (*Shift* + *A* | **Converter** | **ColorRamp**). In the **Name** slot of the **Attribute** node, type **Col**, then connect its **Color** output to the **Fac** input socket of the **ColorRamp** node.
14. In the **ColorRamp** node, set the **Interpolation** to **B-Spline** and move the white color stop to position **0.100**. Connect its **Color** output to the **Fac** input socket of the **Mix Shader_3** node.



The "Corneas" shader with the transparency area located by the Vertex Color layer

15. Add two **Image Textures** nodes (*Shift* + *A* | **Texture** | **Image Texture**) and label them respectively as **COL** and **BUMP**.
16. Add an **Attribute** node (*Shift* + *A* | **Input** | **Attribute**); in the **Name** slot type **UVMap.001** and connect its **Vector** output to the **Vector** input sockets of the two image texture nodes.
17. Add an **RGB** node (*Shift* + *A* | **Input** | **RGB**), a **MixRGB** node (*Shift* + *A* | **Input** | **MixRGB**) and a **Hue Saturation Value** node (*Shift* + *A* | **Input** | **Hue/Saturation**).
18. Click on the **Open** button of the **COL** node to browse to the `textures` folder and load the image `eyeball_col.jpg`.
19. Connect the **Color** output of the **COL** node to the **Color2** input socket of the **MixRGB** node and the output of the **RGB** node to the **Color1** input socket; set the **Blend Type** to **Burn** and the **Fac** value to **0.800**.
20. Connect the **Color** output of the **MixRGB** node to the **Color** input socket of the **Hue Saturation Value** node, and the output of this latter node to the **Color** input sockets of the **Diffuse BSDF** and **Subsurface Scattering** nodes.
21. Set the **RGB** node color to **R 0.800**, **G 0.466**, **B 0.000**; set the **Saturation** value of the **Hue Saturation Value** node to **0.900**.
22. Click on the **Open** button of the **BUMP** node to browse to the `textures` folder and load the image `eyeball_bump.jpg`; set **Color Space** to **Non-Color Data**.
23. Add a **Bump** node (*Shift* + *A* | **Vector** | **Bump**) and connect the **Color** output of the **BUMP** node to the **Height** input socket of the **Bump** node. Connect the **Normal** output of this latter node to the **Normal** input sockets of the **Diffuse BSDF**, **Glossy BSDF1**, and **Subsurface Scattering** nodes, and set the **Strength** to **0.050**.

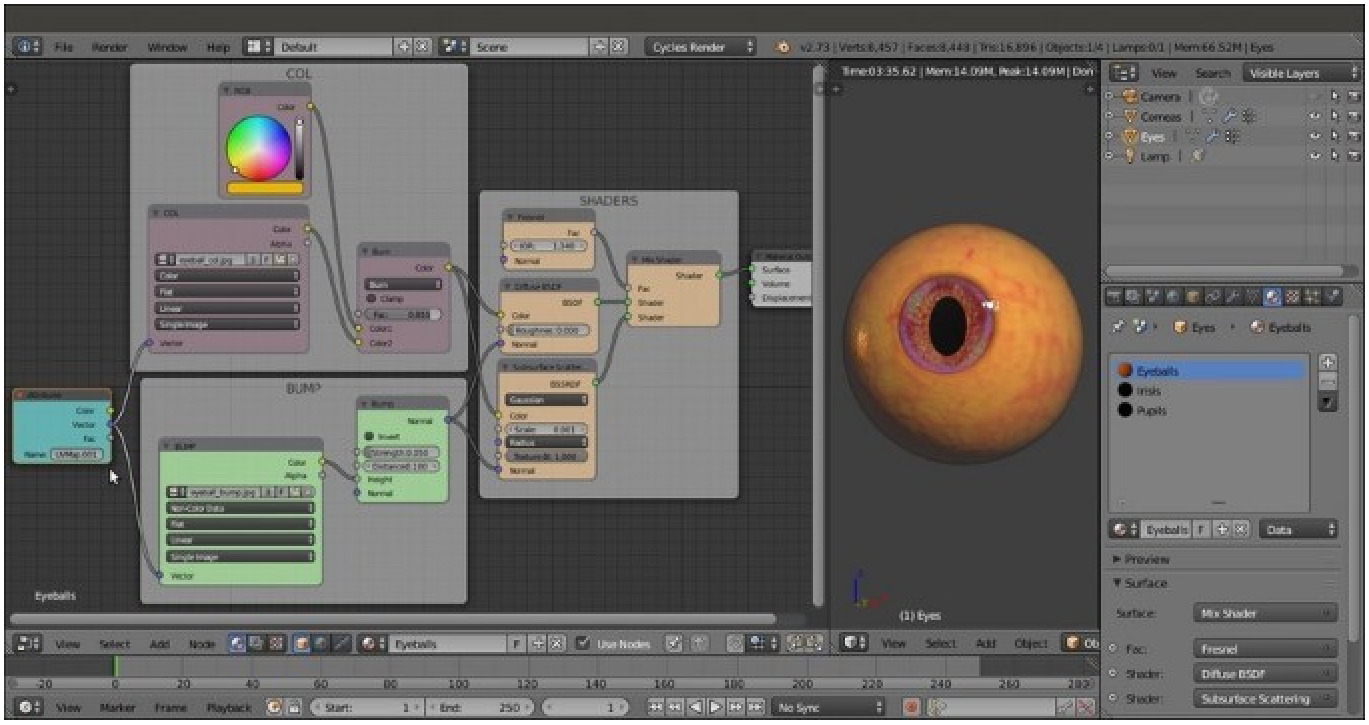
24. If you wish, add frames and colors to the different components to make the shader more easily readable:



The textured "Corneas" material

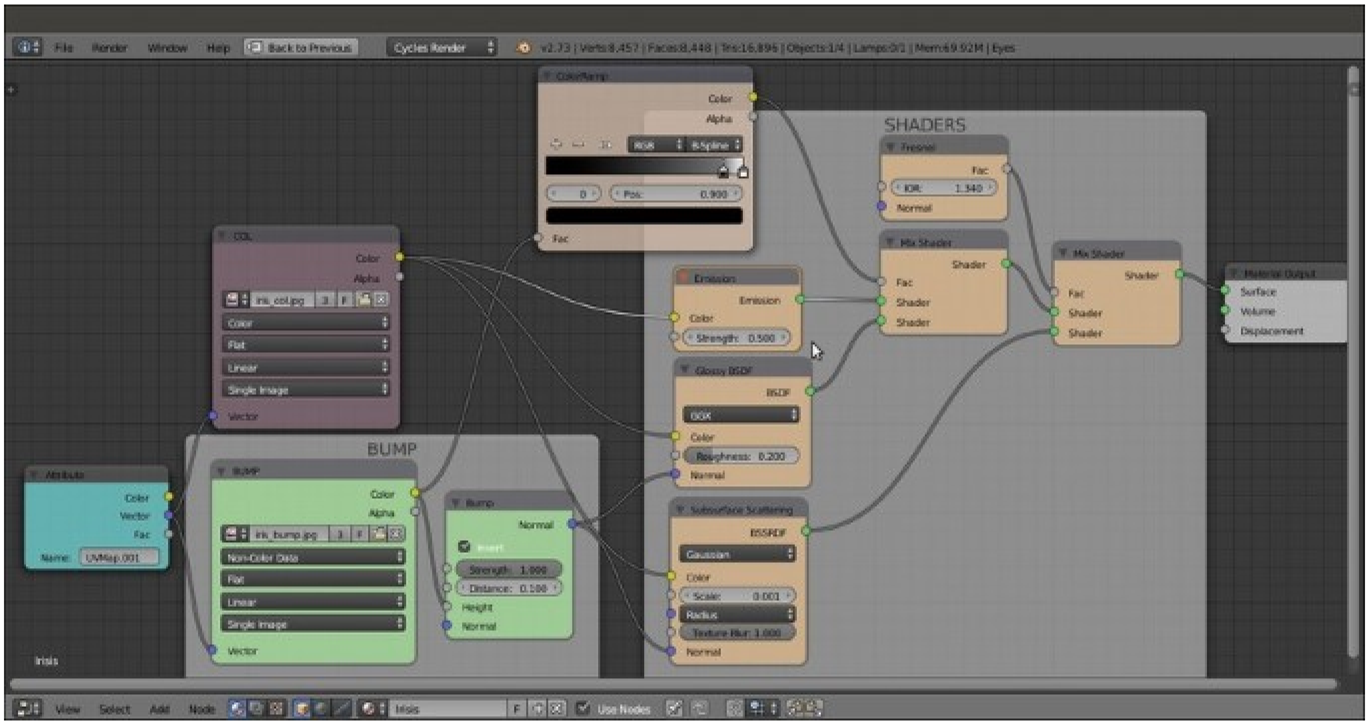
Now let's quickly see the materials for the **Eyes** object:

- As you can see in the following screenshot, the `Eyeballs` material is essentially the same as we just made for the **Corneas** except for the transparent part; this material, by the way, is obsolete because it's hidden behind the **Corneas'** opaque surface, so can be safely omitted (but I left it in place in case you want to try the totally transparent **Cornea**):



The "Eyeballs" material and the completed rendered eye

- The `Irises` material follows the same scheme; the only differences are in the fact that it uses different image textures (`iris_col.jpg` and `iris_bump.jpg`) and that a contrasted (by a **ColorRamp** node) version of the bump image is used as a factor for the mixing of an **Emission** shader; note that the color map is also connected to this **Emission** shader:



The "Irises" material network

- The Pupils are a simple, basic, black diffuse material.

To have a look at these materials, open the `Gidiosaurus_shaders_Cycles.blend` file and select the **Corneas** and **Eyes** objects in the **Outliner**.

How it works…

These shaders are quite simple; the more complex one is the shader for the **Corneas**, essentially because it's made up of **two** materials, one with a slight bump effect and one totally smooth, mixed on the ground of the black and white **Vertex Color** layer that takes care also of the distribution of the transparent and opaque materials on the **Corneas** object itself.

If you are wondering why we didn't use the `Eyeball` material on the underlying **Eyes** sphere, leaving the **Corneas** object totally transparent, the reason is simple: in Cycles, to have a material transparent but also reflecting the environment, you need to use a **Transparent** shader mixed with a **Glass** or a **Glossy** shader node, that inevitably will make whatever material is behind appear darker; sometimes this can look right, in this case I preferred to use a different approach.

Note

Note that the transparent part in front of the iris of the cornea, to be anatomically correct, should be a convex, bulging half sphere; instead, we modeled the cornea as a simple spherical sheath around the eyeball to avoid complications with the open/closed movements of the eyelids.

Building the armor shaders in Cycles

The last thing to do, for this chapter, is to create the shaders for the **Armor** object, made up of metallic **plates** and leather **tiers**.

Getting ready

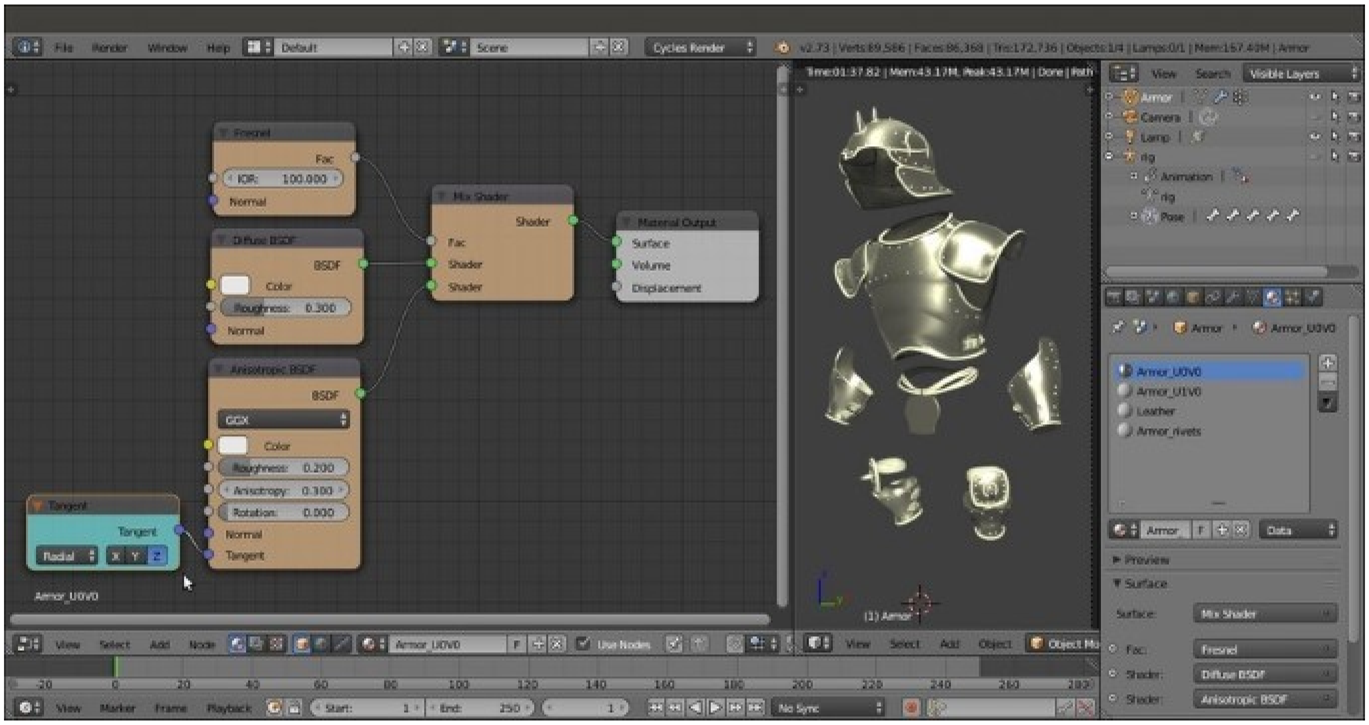
Continuing from the previously saved blend file:

1. Enable the **6th** and the **13th** scene layer and select the **Armor** object in the **Outliner**.
2. Put the mouse pointer in the 3D viewport and press the *0* key on the numpad to go into **Camera** view; fit the window into the field of view.
3. Go to the **Material** window and press the **+** icon button to the right side to add **four** empty material slots to the **armor**. Select the first material slot and click on the **New** button in the **Surface** subpanel, and rename the material `Armor_U0V0`.
4. Select the **second** material slot, click on the **New** button and rename the material as `Armor_U1V0`; repeat for the **third** slot and rename the material as `Leather` and repeat also for the **fourth** slot and rename the material `Armor_rivets`.
5. Switch the **Node Editor** window temporarily with a **UV/Image Editor** window, then press *Tab* to go into **Edit Mode**; go to the **UV Maps** subpanel under the **Object Data** window to be sure you have the **UVMap** coordinates layer (the first one) as the active one, then enable the *Keep UV and edit mode mesh selection in sync* button on the **UV/Image Editor** toolbar.
6. In the **Node Editor** window, box-select the UV islands of the **U1V0** tile, then in the **Material** window, select the `Armor_U1V0` material and click on the **Assign** button.
7. Still in **Edit Mode**, select all the tiers vertices and then select the `Leather` material, click on the **Assign** button; repeat the operation by selecting all the rivets and assigning them to the `Armor_rivets` material.
8. Disable the *Keep UV and edit mode mesh selection in sync* button on the **UV/Image Editor** toolbar, go out of **Edit Mode** and switch the **UV/Image Editor** window back to the **Node Editor** window.
9. Put the mouse pointer inside the 3D viewport and press *Shift* + *Z* to start the **Rendered** preview.

How to do it…

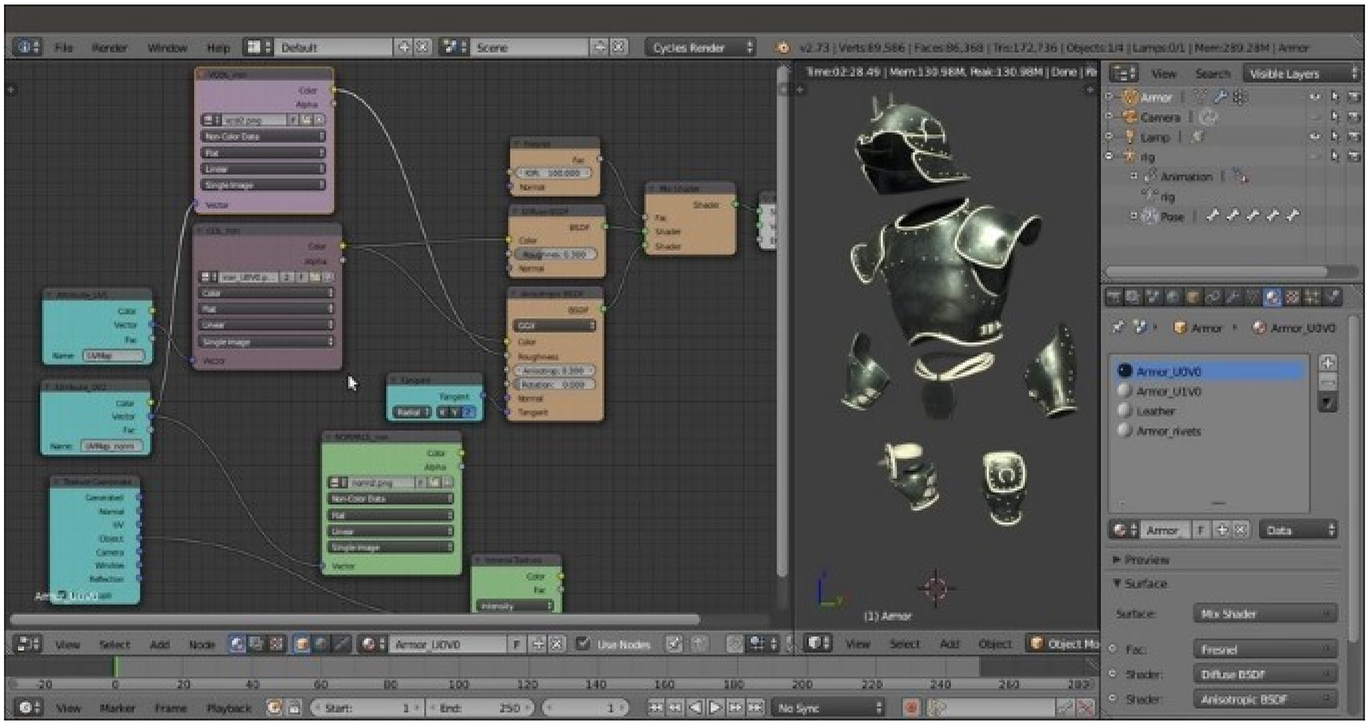
We are first going to create the shader for the metal **plates**:

1. In the **Material** window, select the `Armor_U0V0` material slot.
2. Go to the **Node Editor** window and switch the **Diffuse BSDF** shader node with a **Mix Shader** node; in the first **Shader** slot, select a **Diffuse BSDF** shader node and in the second one, select an **Anisotropic BSDF** shader.
3. Go to the **Node Editor** window and set the **Roughness** of the **Diffuse BSDF** shader to **0.300** and the **Anisotropy** of the **Anisotropic BSDF** shader to **0.300**.
4. Add a **Fresnel** node (*Shift* + *A* | **Input** | **Fresnel**) and connect its output to the **Fac** input socket of the **Mix Shader** node; set the **IOR** to **100.000**.
5. Add a **Tangent** node (*Shift* + *A* | **Input** | **Tangent**) and connect its output to the **Tangent** input socket of the **Anisotropic BSDF** shader node; set the **Tangent** to **Z**.



Starting to build the metal shader for the armor

6. Add a **Frame** (*Shift* + *A* | **Layout** | **Frame**) and parent the nodes, except the **Material Output**, to it, then label it as **SHADERS**.
7. Add three **Image Textures** nodes (*Shift* + *A* | **Texture** | **Image Texture**) and a **Voronoi Texture** node (*Shift* + *A* | **Texture** | **Voronoi Texture**), then add two **Attribute** nodes (*Shift* + *A* | **Input** | **Attribute**) and a **Texture Coordinate** node (*Shift* + *A* | **Input** | **Texture Coordinate**).
8. Label the **Attribute** nodes as **Attribute_UV1** and **Attribute_UV2**. Label the **Image Texture** nodes as **COL_iron**, **NORMALS_iron**, and **VCOL_iron**.
9. Connect the **Vector** output of the **Attribute_UV1** node to the **Vector** input socket of the **COL_iron** node. Connect the **Vector** output of the **Attribute_UV2** to the **Vector** input sockets of both the **VCOL_iron** and **NORMALS_iron** nodes. Connect the **Object** output of the **Texture Coordinate** node to the **Vector** input socket of the **Voronoi Texture** node.
10. Click on the **Open** button of the **VCOL_iron** node, browse to the `textures` folder and load the image `vcol2.png`. Set the **Color Space** to **Non-Color Data**. Connect its **Color** output to the **Roughness** input socket of the **Anisotropic BSDF** shader node.
11. Click on the **Open** button of the **COL_iron** node, browse to the `textures` folder and load the image `iron_U0V0.png`. Connect its **Color** output to the **Color** input sockets of the **Diffuse BSDF** and **Anisotropic BSDF** shader nodes.
12. Click on the **Open** button of the **NORMALS_iron** node, browse to the `textures` folder and load the image `norm2.png`. Set the **Color Space** to **Non-Color Data**.
13. Set the **Scale** of the **Voronoi Texture** to **15.000**:

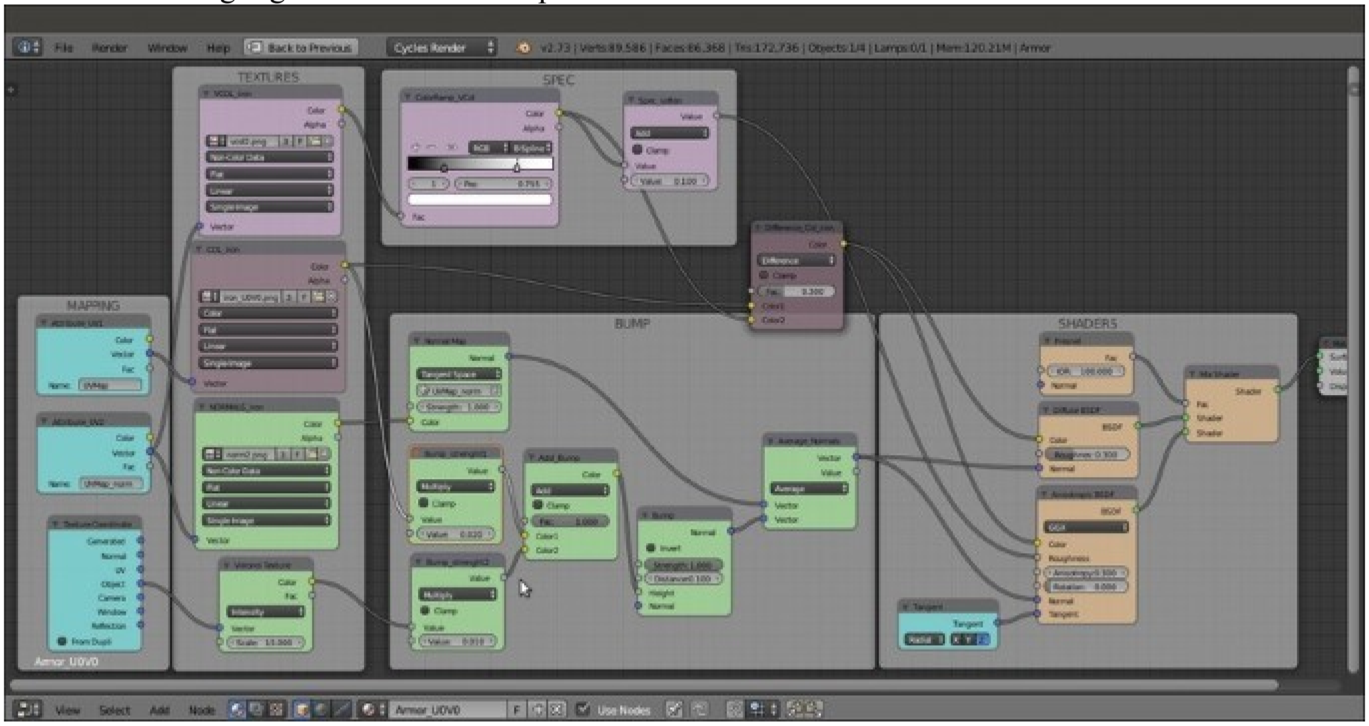


Adding the textures to the "Armor_U0V0" material

14. Add a **ColorRamp** node (*Shift* + *A* | **Converter** | **ColorRamp**) and a **Math** node (*Shift* + *A* | **Converter** | **Math**). Paste the **ColorRamp** node right after the **VCOL** node, and the **Math** node right after the **ColorRamp**.
15. Label the **ColorRamp** as **ColorRamp_Vcol** and set the **Interpolation** to **B-Spline**, then move the black color stop to position **0.245** and the white color stop to position **0.755**.
16. Label the **Math** node as **Spec_soften** and set the second **Value** to **0.100**.
17. Add a **MixRGB** node (*Shift* + *A* | **Color** | **MixRGB**) and label it as **Difference_Col_iron**; set the **Blend Type** to **Difference** and the **Fac** value to **0.300**.
18. Connect the **Color** output of the **COL** node to the **Color1** input socket and the **Color** output of the **ColorRamp_Vcol** node to the **Color2** input socket. Connect the **Color** output of the **Difference_Col_iron** node to the **Color** input sockets of the **Diffuse BSDF** and the **Anisotropic BSDF** shader nodes, replacing the old connections.
19. Add a **Normal Map** node (*Shift* + *A* | **Vector** | **Normal Map**), a **Bump** node (*Shift* + *A* | **Vector** | **Bump**), and a **Vector Math** node (*Shift* + *A* | **Converter** | **Vector Math**).
20. Connect the **Color** output of the **NORMALS_iron** node to the **Color** input socket of the **Normal Map** node; click on the empty slot (*UV Map for tangent space maps*) on this latter node to select the **UVMap_norm** item.
21. Connect the **Normal** output of the **Normal Map** node to the first **Vector** input socket of the **Vector Math** node; label this latter as **Average_Normals** and set the **Operation** to **Average**, then connect its **Vector** output to the **Normal** input sockets of the **Diffuse BSDF** and **Anisotropic BSDF** shader nodes.
22. Add a **MixRGB** node (*Shift* + *A* | **Color** | **MixRGB**), label it as **Add_Bump**, set the **Blend Type** to **Add** and the **Fac** value to **1.000**. Connect the **Color** output of the **COL** node to the **Color1**

input socket of the **Add_Bump** node also, and the **Color** output of the **Voronoi Texture** node to the **Color2** input socket.

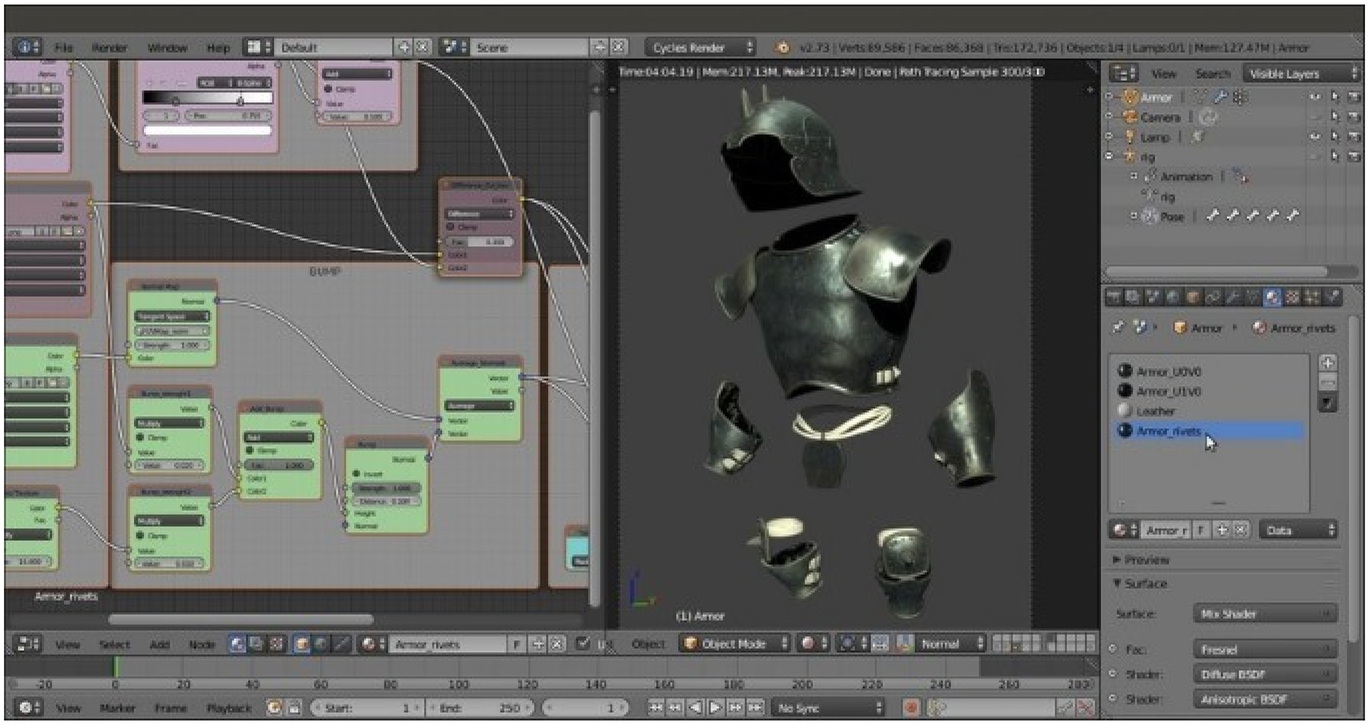
23. Connect the **Color** output of the **Add_Bump** node to the **Height** input socket of the **Bump** node, and the **Normal** output of this latter node to the second **Vector** input socket of the **Average_Normals** node. Set the **Strength** of the **Bump** node to **1.000**.
24. Add two **Math** nodes (*Shift* + *A* | **Converter** | **Math**), label them respectively as **Bump_strength1** and **Bump_strength2**; set the **Operation** to **Multiply** for both, then paste the **Bump_strength1** node between the **COL_iron** and the **Add_Bump** nodes and set the second **Value** to **0.020**. Paste the **Bump_strength2** node between the **Voronoi_Texture** and the **Add_Bump** nodes, and set the second **Value** to **0.010**.
25. Add frames to highlight the different components:



The completed "Armor_U0V0" material

The first **Armor** shader is ready! Now it's very easy to obtain the others:

26. Press *A* twice to select all the nodes, then press *Ctrl* + *C* to copy them.
27. In the **Material** window, select the `Armor_U1V0` material slot and in the **Node Editor** window, delete the default **Diffuse** and **Material Output** nodes; then press *Ctrl* + *V* to paste the nodes copied from the other material.
28. Zoom to the **COL** node and click on the numbered button to the right side of the texture name slot to make it single user, then click on the folder icon button to browse to the `texture` folder and load the image `iron_U1V0.png`.
29. Reselect the `Armor_U0V0` material slot and repeat the step 26 and 27, this time pasting the nodes inside the `Armor_rivets` material slot:



The "Armor_rivets" material and the rendered completed armor

30. Save the file.

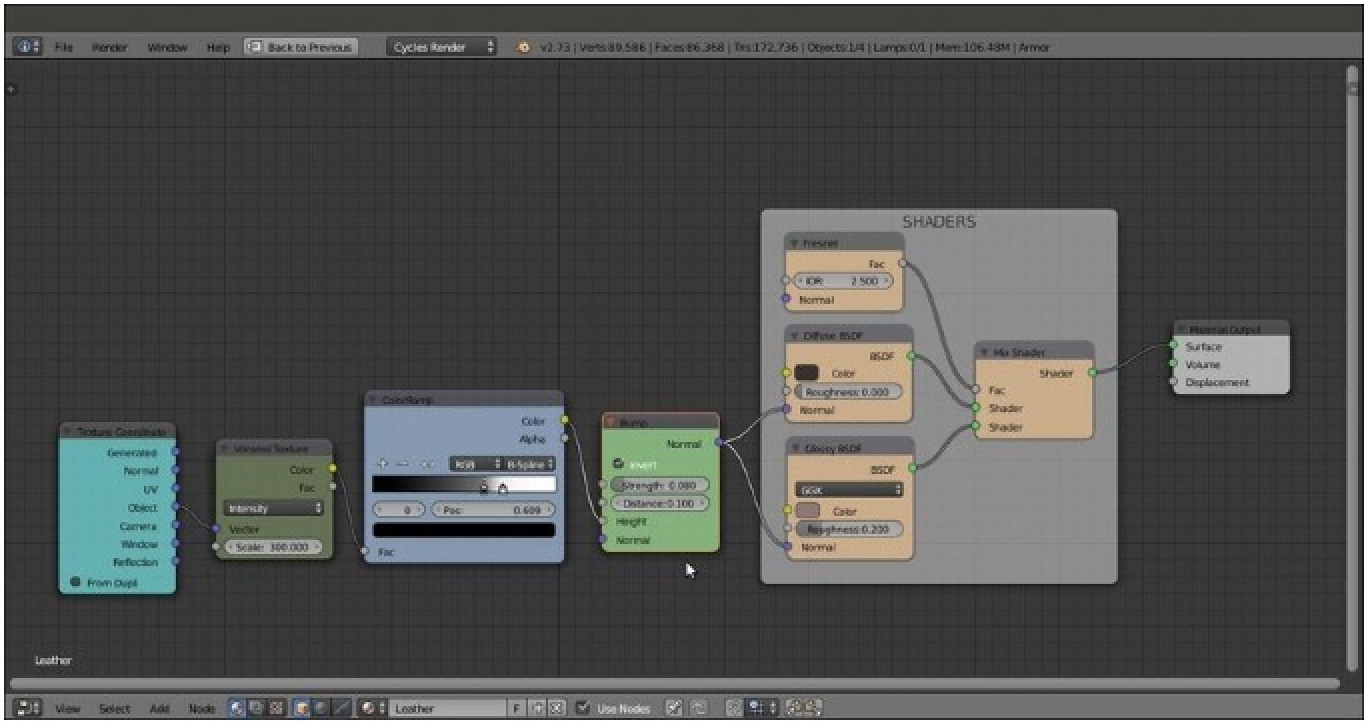
How it works…

The construction of the metallic **armor plates** material follows basically the same scheme we used for the other materials:

- First the shaders were produced, where the metallic look is mainly due to the **Anisotropic BSDF** shader mixed with the diffuse component with a quite high **IOR** value (metals can often have values from **20.000** to **200.000**; we used a midway value of **100.000**).
- The shininess of the metallic surface has been modulated through the output of the `vcol2.png` image, a **Dirty Vertex Color** layer we had previously baked to an image.
- The color of the **Armor** surface has been modulated as well through a **Difference** node with the same `vcol2.png` image.
- The bump pattern works by first adding the **Voronoi** and the **color map** output and then averaging the result with the **normal map** output.

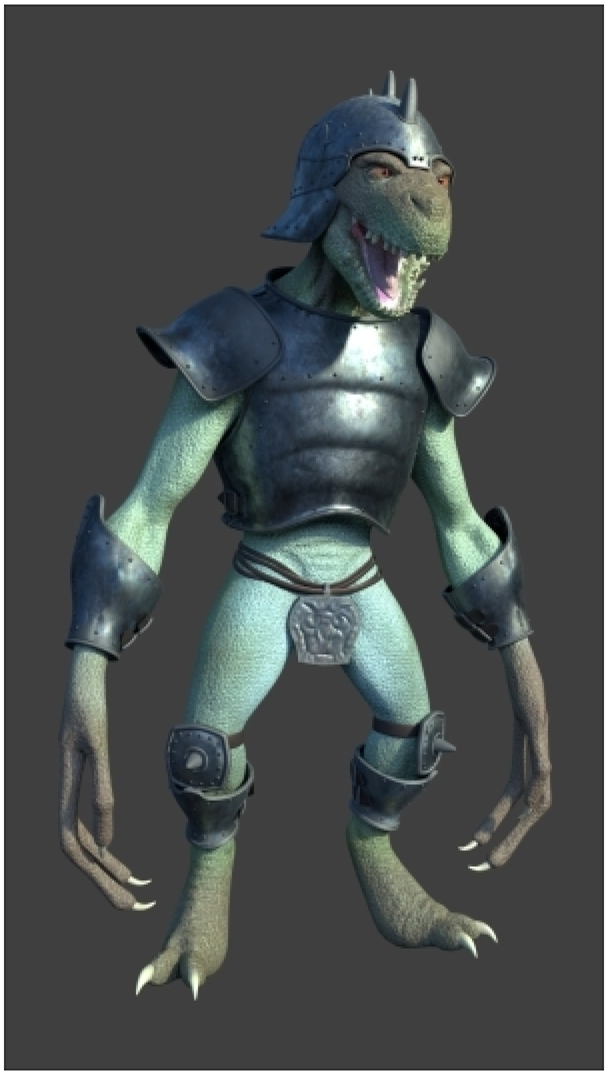
There's more…

The last material created for our character is a very simple leather material made mainly from the output of a **Voronoi Texture** node, contrasted, inverted, and used as bump pattern:



The simple "Leather" material

This completes the creation of the **Gidiosaurus** shaders in **Cycles**:



The completed Gidiosaurus character in Cycles

Of course, reflecting materials, for example, the metallic **armor** surface or the **corneas** (but to some extent also the reptile **skin**), need something to reflect to show them at their best; we'll see this in the last chapter of this cookbook.

In the next chapter, which is the penultimate chapter, we'll see the creation of the same materials in **Blender Internal**.

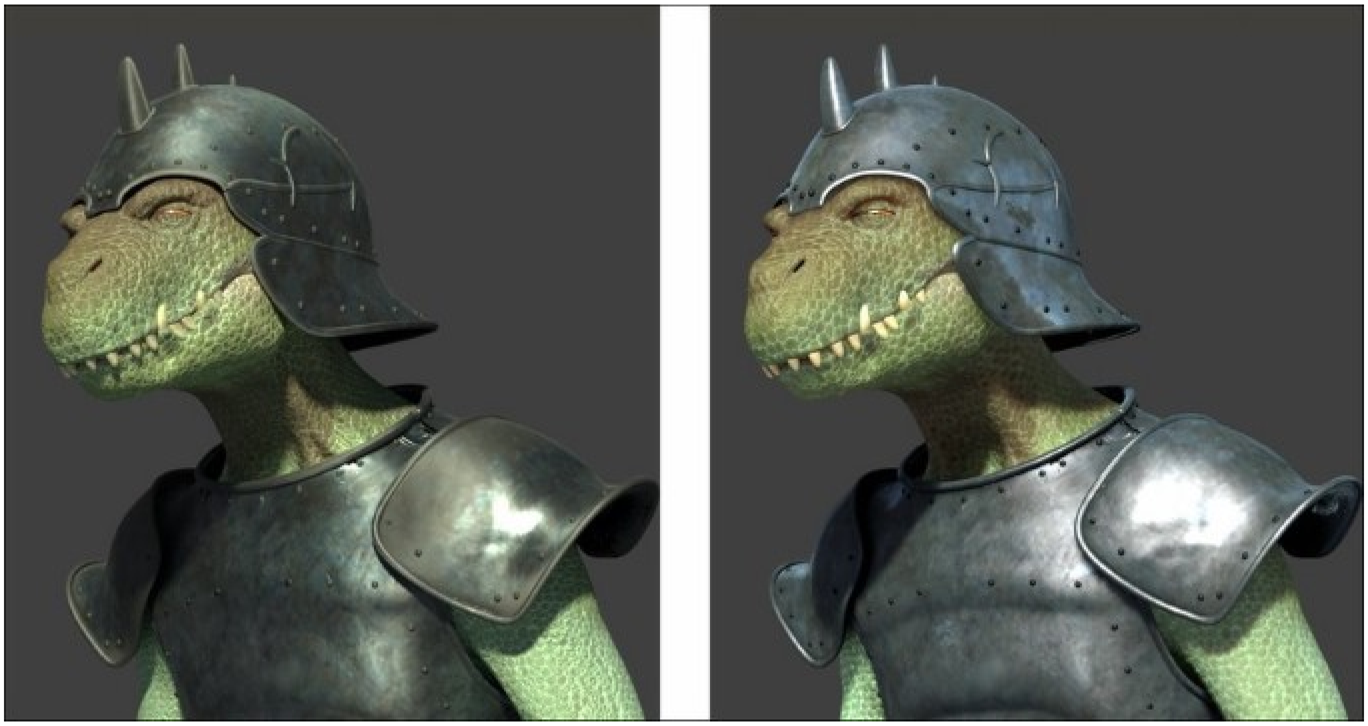
Chapter 13. Creating the Materials in Blender Internal

In this chapter, we will cover the following recipes:

- Building the reptile skin shaders in Blender Internal
- Building the eyes' shaders in Blender Internal
- Building the armor shaders in Blender Internal

Introduction

In this chapter we'll see how to set up the materials for the **Gidiosaurus** and the **Armor** in the **Blender Render** engine; in fact, although not exactly of the same quality as in **Cycles**, it is also possible to obtain quite similar shader results in **Blender Internal**:



Comparison of the Gidiosaurus character rendered in Cycles (left) and Blender Internal (right)

If you are wondering why we should re-do in the **Blender Render** engine, which is quite old and no longer developed and/or supported, the same thing we have already done in **Cycles**, there are several possible reasons: for example, no doubt **Cycles** is superior in quality but, compared with the scanline **BI**, its rendering is (and, being a path-tracer, always will be) slower; even with the aid of a render-farm, rendering times are still a *money* issue in the production of animations.