9
Special Materials

In this chapter, we will cover:

- Creating fur and hair
- Creating a wire-frame shader
- Creating a shadeless material in Cycles
- Creating a fake immersion effect material
- Creating a fake volume light material

Introduction

In this chapter we are going to see some special materials, that is, ones that can be used for particular special effects or for not necessarily realistic objects.

The creation of fur and hair based on the particle system has been included here, mostly because at the moment, the hair rendering feature in Cycles is accessible only by enabling the Experimental feature set in the **Render** window. This is going to change in the future for sure.

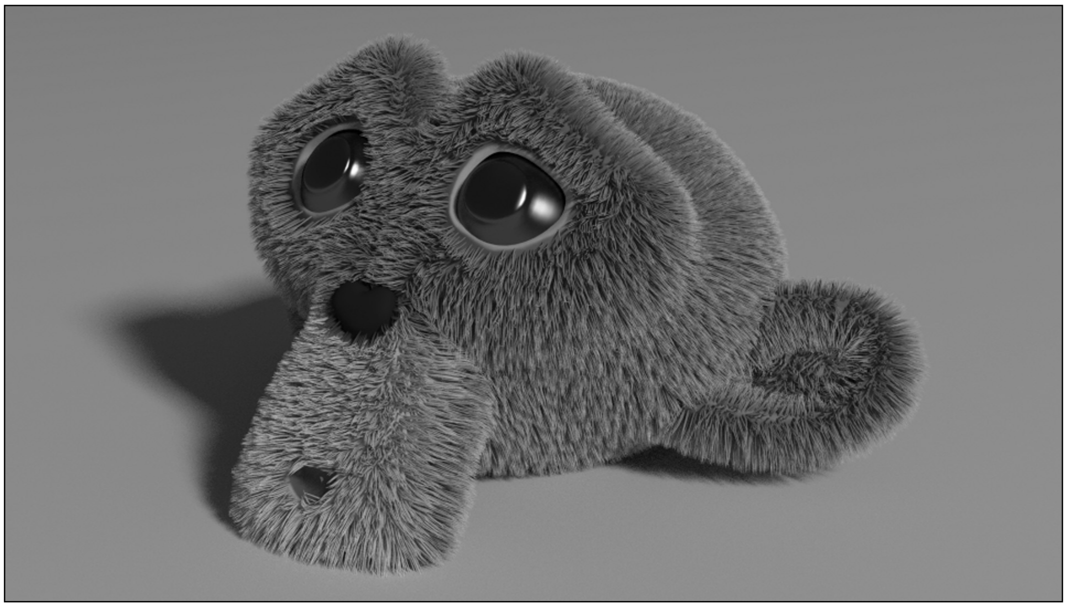
Creating fur and hair

Fur, in the world of computer graphics, is considered amongst the most difficult things to recreate, both because it's generally quite expensive from a memory management point of view (a single character can easily have millions of hair strands) and also because it can be quite a task to make a believable shader that can work under different light conditions.

Blender is not new to fur creation, the open movie Big Buck Bunny had the goal to add tools for fur creation to the **Blender Internal** (**BI**) rendering engine, and it succeeded. In BI, fur is rendered using a new type of primitive, **strand**, instanced on a particle system that can be edited, combed, and tweaked in several ways to obtain the best possible result.

The same applies for the Cycles rendering engine; the particles groundwork is the same but, as already said, at the moment the rendering code is still an experimental feature. It already works pretty well, though.

So, in this recipe, we will create the fur shader of the following image by using a particle system and the Experimental rendering feature set:



Getting ready

Start Blender and open the file `1301OS_09_hair_start.blend`. In the scene there is a Suzanne primitive (**Suzanne_teddybear**) with a hair particle system, named **teddybear**, already set (see it in the **Particles** window) to resemble the fur of a cuddly toy.

The **Suzanne_teddybear** mesh is already unwrapped and has a **Vertex Group** named **density**, used in the particle system panel (**Vertex Groups** tab in the **Particles** window) to establish the density distribution of the fur on the mesh.

If you go to the **Render** window you will find that, under the **Render** tab, the **Feature Set** slot is on **Experimental**; at the moment, this is needed to activate the hair rendering feature in Cycles; and it shows two more tabs in the **Particles** window—**Cycles Hair Rendering** and **Cycles Hair Setting**.

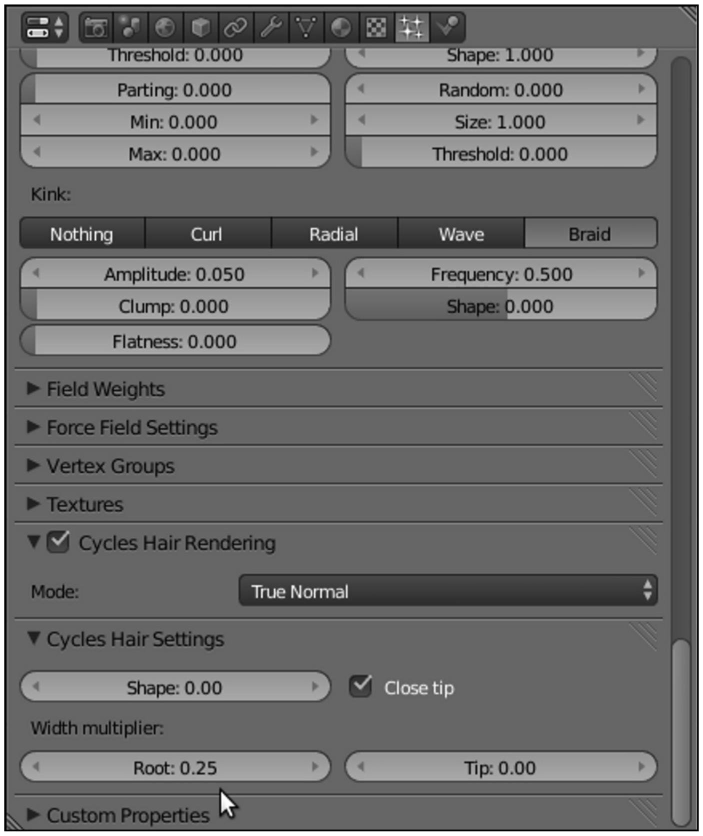
We are going to add three different materials to the Suzanne object; **base_stuff**, that is the basic material for the mesh; **eyes** and **teddybear** for the fur. Note that the order of the materials in the **Material** window is important; we'll see later why.

How to do it...

Let's start by the creation of the materials for the eyes, the Suzanne skin, and the fur:

1. Select the Suzanne mesh and click on the **New** button in the **Node Editor** window header or in the **Material** window to the right; rename the material `base_stuff`.
2. Press *Tab* to go in Edit mode and select the eyes vertexes (put the mouse cursor over the target and press the *L* key to select all the linked vertexes); click on the little **+** icon to the right of the **Material** window (**Add a new material slot**) and add a new material. Click on the **New** button and rename the new material `eyes`, then click on the **Assign** button. Press *Tab* to come out of Edit mode.
3. Click again on the little **+** icon to the right of the **Material** window (**Add a new material slot**) to add a third material (not assigned to any vertex or face, because we are out of Edit mode). Click on the **New** button and rename the new material `teddybear`.
4. Go to the **Particles** window, in the **teddybear** particle system, under the tab **Render**, and make sure that the **Material** number is **3** (it's the **Material used for the particles** button); this means that the third material, starting from the top in the **Material** window, is the one used for the particle system.

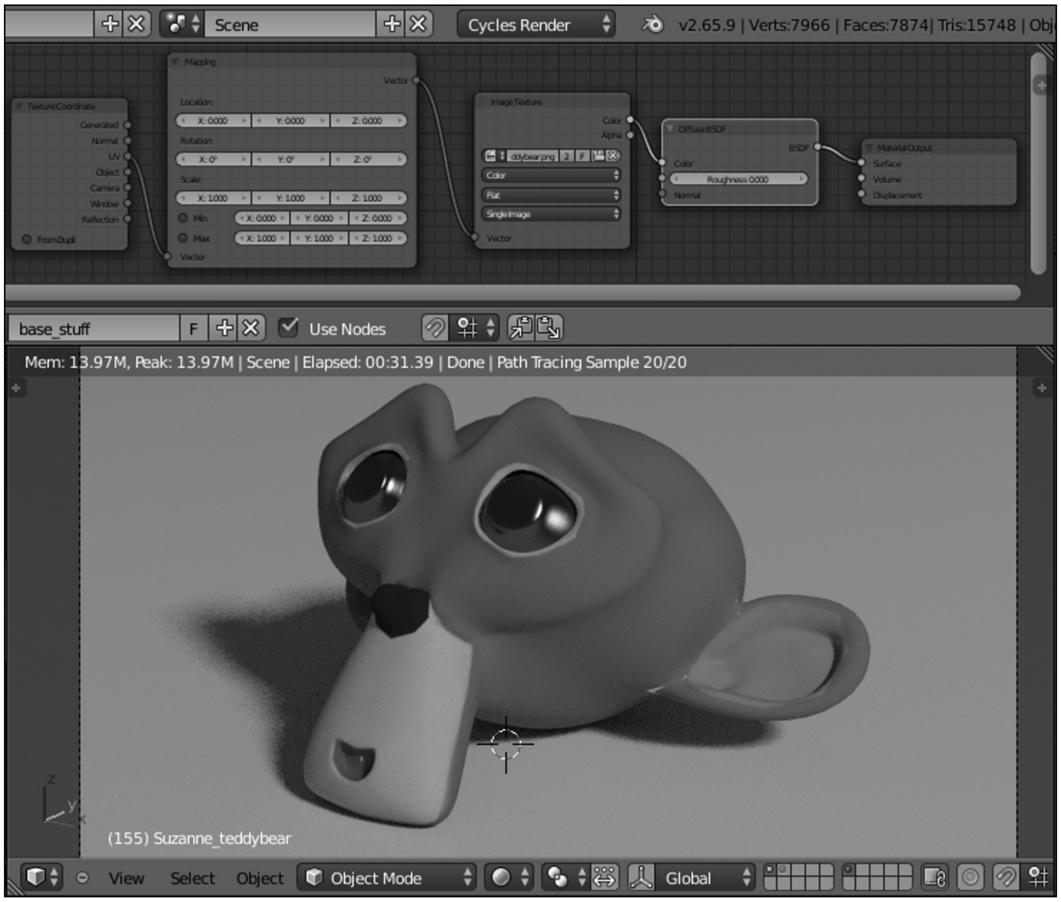
5. Be sure that in the **Cycles Hair Rendering** tab the **Mode** is set to **True Normal** and that in the **Cycles Hair Settings** tab the **Root** is `0.25`, and the **Shape** and **Tip** values are `0.00`.



Threshold: 0.000
Shape: 1.000
Parting: 0.000
Random: 0.000
Min: 0.000
Size: 1.000
Max: 0.000
Threshold: 0.000
Amplitude: 0.050
Frequency: 0.500
Clump: 0.000
Shape: 0.000
Flatness: 0.000
Mode: True Normal
Shape: 0.00
Root: 0.25
Tip: 0.00

6. Now, in the **Material** window, select the **base_stuff** material; in the **Node Editor** window add a **Texture Coordinate** node (press Shift + *A* and go to **Input** | **Texture Coordinate**), a **Mapping** node (press Shift + *A* and go to **Vector** | **Mapping**), and an **Image Texture** node (press Shift + *A* and go to **Texture** | **Image Texture**).
7. Connect the **UV** output of the **Texture Coordinate** node to the **Vector** input socket of the **Mapping** node and the output of this latter to the **Vector** input socket of the **Image Texture** node.
8. Connect the **Color** output of the **Image Texture** node to the **Color** input socket of the **Diffuse BSDF** shader; click on the **Open** button and browse to the **textures** folder to load the image `teddybear.png` (a simple color map painted directly in Blender).
9. Now, in the **Material** window, select the **eyes** material and switch the **Diffuse** shader with a **Mix Shader** node; in the first **Shader** slot select a **Diffuse BSDF** shader and in the second one a **Glossy BSDF** shader node.

10. Set the **Mix Shader** factor to `0.200`, the **Diffuse** node's **Color** to `R 0.010`, `G 0.003`, and `B 0.001` and the **Glossy** node's **Roughness** to `0.100`.

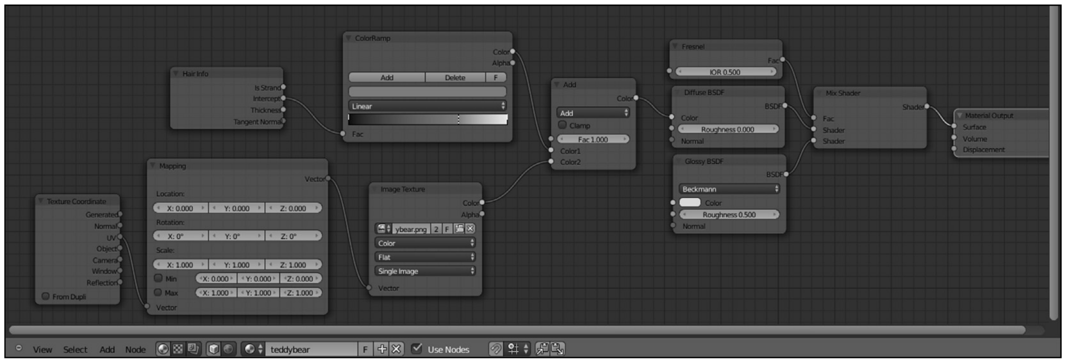


Mem: 13.97M, Peak: 13.97M | Scene | Elapsed: 00:31.39 | Done | Path Tracing Sample 20/20

(155) Suzanne_teddybear

11. In the **Material** window select the **teddybear** material and switch the **Diffuse BSDF** node with a **Mix Shader** node; in the first **Shader** slot select a **Diffuse BSDF** shader node and in the second one a **Glossy BSDF** shader node. Set the **Glossy** node's **Color** to `R 0.800`, `G 0.748`, and `B 0.448` and its **Roughness** to `0.500`.
12. Add a **Fresnel** node (press *Shift* + *A* and go to **Input** | **Fresnel**) and connect it to the **Fac** input socket of the **Mix Shader** node; set the **IOR** to `0.500`.
13. Add a **Hair Info** node (press *Shift* + *A* and go to **Input** | **Hair Info**) and a **ColorRamp** node (press *Shift* + *A* and go to **Convertor** | **ColorRamp**); connect the **Intercept** output of the **Hair Info** node to the **Fac** input socket of the **ColorRamp** node and the **Color** output of the latter to the **Color** input socket of the **Diffuse** shader node.

14. Click on the white color marker of the **ColorRamp** node and set its **Color** to `R 1.000`, `G 0.789`, and `B 0.462`; click on the **Add** button to add a new color marker, move it to three-quarters of the way along the slider and set its **Color** to `R 0.352`, `G 0.226`, and `B 0.093`.
15. Add a **Mix** node (press *Shift* + *A* and go to **Color** | **Mix**), set **Blend Type** to **Add** and the **Fac** value to `1.000`. Paste it between the **ColorRamp** and the **Diffuse** shader.
16. Add an **Image Texture** node (press *Shift* + *A* and go to **Texture** | **Image Texture**), connect its **Color** output to the **Color2** input socket of the **Add** node and click on the little arrows to the left of the **Open** button to select the already loaded `teddybear.png` image map.
17. Optionally, add a **Texture Coordinate** node (press *Shift* + *A* and go to **Input** | **Texture Coordinate**) and a **Mapping** node (press *Shift* + *A* and go to **Vector** | **Mapping**), and connect the **UV** output of the **Texture Coordinate** node to the **Vector** input socket of the **Mapping** node and the output of this latter to the **Vector** input socket of the **Image Texture** node.



How it works...

From step 1 to step 3 we prepared the three materials to be used; we went in Edit mode to assign the second material, **eyes**, to the eyes part of the mesh, then went back to Object mode to add a third material that doesn't need to be assigned to any face of the mesh, because it is "on the Suzanne mesh", only used for the hair rendering.

In steps 4 and 5 we made sure we had the right particle system settings.

From step 6 to step 8 we built the **base_stuff** material, a simple **Diffuse** shader colored by the UV mapped **teddybear** image texture; note that the texture we used in this first material is used also to give the right color to the hair; it is also useful to have it on the underlying mesh, to cover any hole or missing part in the particle system.

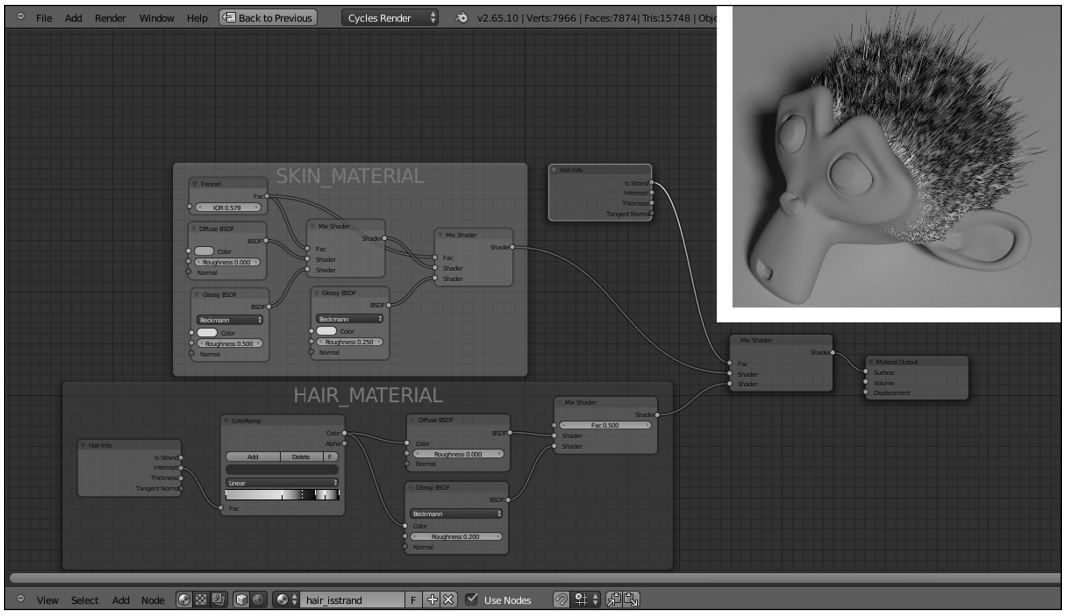
In steps 9 and 10 we built the **eyes** shader; again, a very basic material made of the dark Diffuse color and the light gray Glossy components, mixed by the **Mix Shader** node.

From step 11 to step 17 we built the shader to be used by the particle system for the fur, mixing the already used `teddybear.png` image map, mapped on the UV coordinates, with a **ColorRamp** brown gradient mapped, by the **Intercept** option of the **Hair Info** node, on the length of each hair particle.

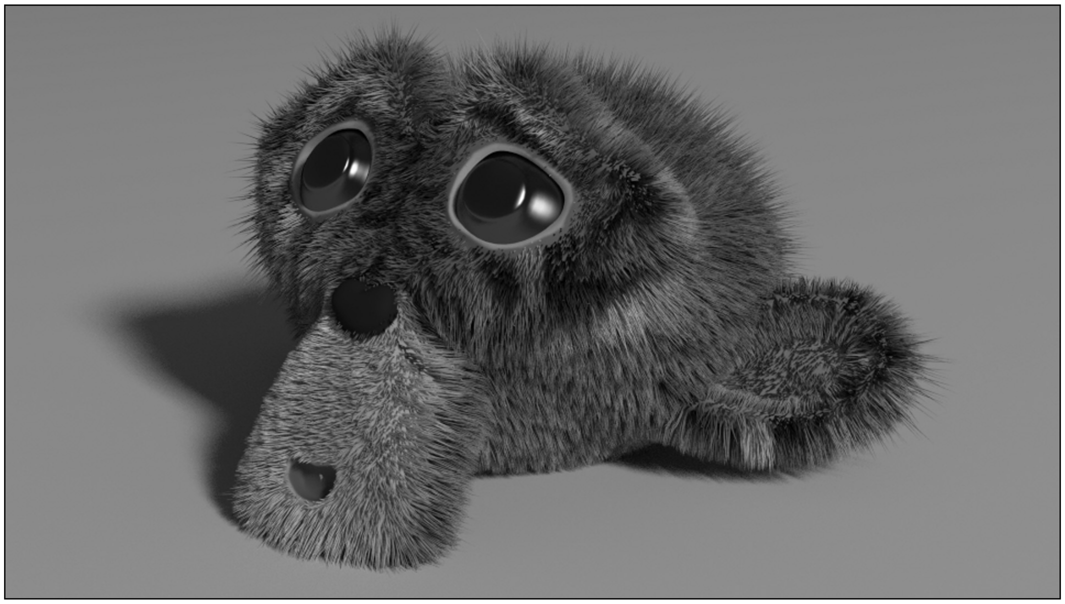
There's more...

The `teddybear.png` image texture has been used both in the **base_stuff** and the **teddybear** materials. Note that this is usually not necessary, because in Blender the particle system hairs get the textures from the surface they are emitted from, so it would have been enough to use the **base_stuff** material for the fur (by setting, in the **Material** button under the tab **Render** in the **Particles** window, the number to **1**). However, because we wanted to add the **ColorRamp** gradient, mapped on the **Intercept** output, to the shader we had to make a new material.

Note that in the **Hair Info** node there is also the Boolean **Is Strand** output which, similar to the outputs of the **Light Path** node (**Is Camera**, **Is Shadow**, and so on) can be used as an alternative to the **Material** button in the **Particles** panel to assign 0 to the emitter mesh and 1 to the fur strands:



This also means that we can use different image textures to obtain fur materials that are different from the material of the particle emitter. For example, in the following image, the `tiger.png` image texture has been used only for the fur, whereas the **base_stuff** material still uses the `teddybear.png` texture.

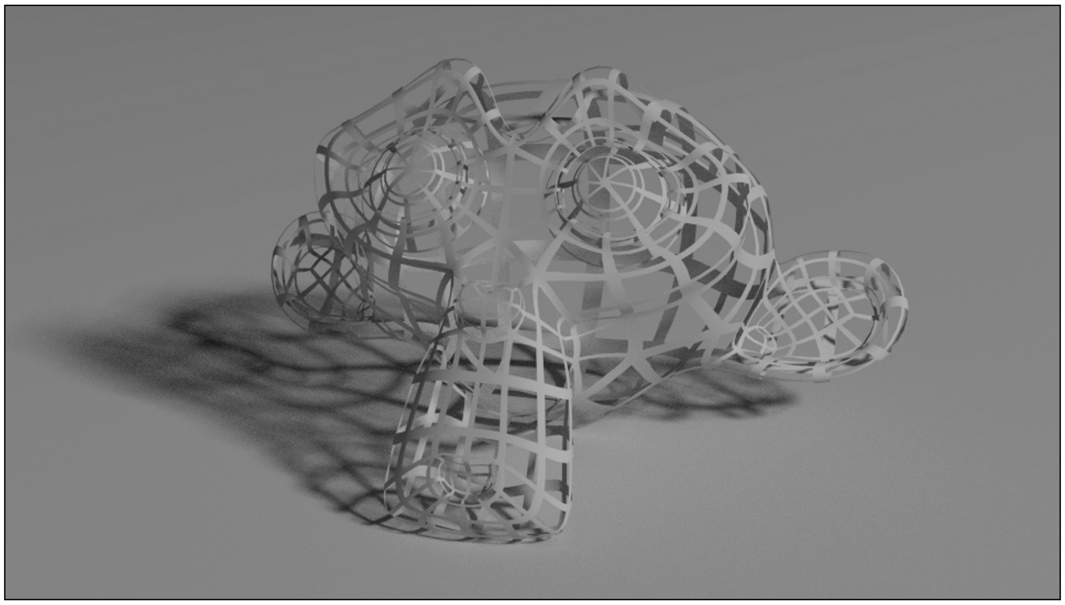


The **Suzanne_tiger** in the previous image also has two different particle systems to create the fur—**tigerfur_long** and **tigerfur_short**— and therefore has three **Vertex Groups** to modulate the fur appearance, **density_long**, **density_short**, and **length**.

To have a look at the **Suzanne_tiger** object open the file `1301OS_09_tiger.blend`.

Creating a wire-frame shader

In this recipe we will create a wire-frame material like this:



Getting ready

Start Blender and open the file `1301OS_09_start.blend`, which contains a preset scene with an unwrapped Suzanne primitive object.

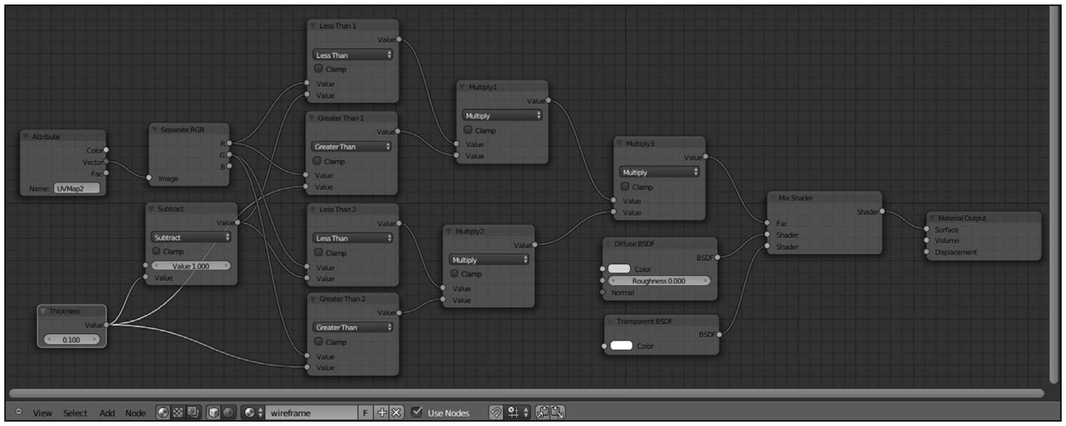
1. Select the Suzanne mesh and press *Tab* to go in Edit mode; go to the **Object Data** window and click on the little **+** icon on a side of the **UV Maps** tab to add a new UV coordinates layer.
2. Put the mouse in the 3D view and press *U*; in the **UV Mapping** pop-up menu select **Reset**.
3. Go out of Edit mode and rename the second UV coordinate set **UVMap2**.

How to do it...

Let's go with the wire-frame material creation:

1. Click on the **New** button in the **Node Editor** window header or in the **Material** window under the **Properties** panel; rename the material **wireframe**.
2. Add a **Separate RGB** node (press *Shift* + *A* and go to **Convertor | Separate RGB**) and a **Math** node (press *Shift* + *A* and go to **Convertor** | **Math**); set the **Math** node operation to **Subtract** and the first **Value** to `1.000`.
3. Press *Shift* + *D* to duplicate the **Math** node and set the operation to **Less Than**; rename it `Less Than 1` and connect the **R** output of the **Separate RGB** node to its first **Value** input socket and the **Value** output of the **Subtract** math node to its second **Value** input socket.
4. Press *Shift* + *D* to duplicate the **Less Than 1** node and set the operation to **Greater Than**; rename it as `Greater Than 1` and move it under the **Less Than 1** node. Connect the **R** output of the **Separate RGB** node to its first **Value** input socket.
5. Add a **Value** node (press *Shift* + *A* and go to **Input** | **Value**), rename it `Thickness` and connect it to the second **Value** input socket of the **Subtract** node, then to the second **Value** input socket of the **Greater Than 1** node; set the value to `0.100`.
6. Press *Shift* + *D* to duplicate the **Less Than 1** node and rename it `Less Than 2`; move it under the **Greater Than 1** node. Connect the **G** output of the **Separate RGB** node to its first **Value** input socket and the **Value** output of the **Subtract** math node to its second **Value** input socket.
7. Press *Shift* + *D* to duplicate the **Greater Than 1** node and rename it **Greater Than 2**; move it under the **Less Than 2** node. Connect the **G** output of the **Separate RGB** node to its first **Value** input socket and the **Thickness** value node output to its second **Value** input socket.
8. Press *Shift* + *D* to duplicate one of the **Math** nodes (it doesn't matter which one), set the operation to **Multiply**, rename it `Multiply1` and connect the output of the **Less Than 1** node to the first **Value** input socket and the output of the **Greater Than 1** node to the second **Value** input socket.
9. Press *Shift* + *D* to duplicate the **Multiply1** node and rename it `Multiply2`; connect the output of the **Less Than 2** node to the first **Value** input socket and the output of the **Greater Than 2** node to the second **Value** input socket.

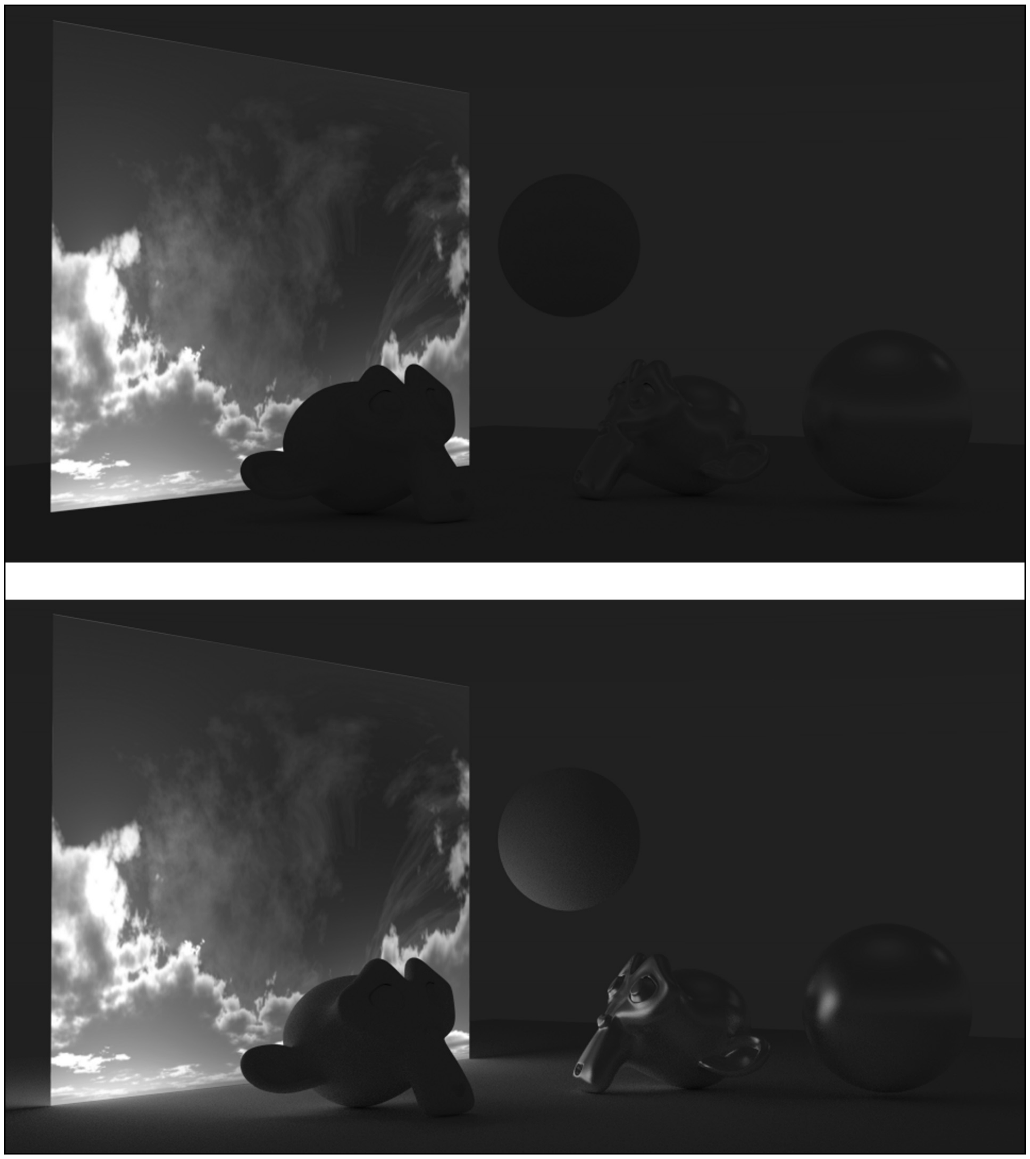
10. Press *Shift* + *D* to duplicate the **Multiply1** node and rename it `Multiply3`; connect the output of the **Multiply1** node to the first **Value** input socket and the output of the **Multiply2** node to the second **Value** input socket.
11. Add an **Attribute** node (press *Shift* + *A* and go to **Input** | **Attribute**); in the **Name** slot write the name of the second set of UV coordinates, that is **UVMap2** and connect its **Vector** output to the **Image** input socket of the **Separate RGB** node.
12. Add a **Mix Shader** node (press *Shift* + *A* and go to **Shader** | **Mix Shader**) and paste it between the **Diffuse** shader and the **Material Output** node. Connect the output of the **Multiply3** node to the **Fac** input socket of the **Mix Shader** node.
13. Add a **Transparent BSDF** node (press *Shift* + *A* and go to **Shader** | **Transparent BSDF**) and connect it to the second **Shader** input socket of the **Mix Shader** node.
14. Change the color of the **Diffuse** shader node to something more amusing than a simple white. And, for better results, don't forget to deselect the **Subdivide Uvs** option in the **Subdivision Surface** modifier. Use the **Thickness** node value to establish the size of the wires.



Creating a shadeless material in Cycles

In this recipe we will create a shadeless material.

A shadeless material is simply a material that behave as if self-illuminated but not actually emitting any light on the nearby objects;



In the previous screenshot you can see a vertical plane, mapped with a cloudy sky texture, with the shadeless material (top) and with a normal emission shader (bottom); in the top case, the sky texture is perfectly visible but it's not affecting the spheres, Suzannes, or the floor plane. In fact, a shadeless material is perfect for backdrop elements mapped on planes (or, more often, on domes; that is, unwrapped half-spheres) to simulate the sky, the clouds, even distant trees, forests, and mountains in the background of your scene.

In Blender Internal, obtaining a shadeless material is very simple; just enable the appropriate option in the material panel. In Cycles it's a different thing, and actually there are two ways to obtain the effect of a shadeless object; one by the material and one with the aid of the **Ray Visibility** tab in the **Object** window under the **Properties** panel.

Getting ready

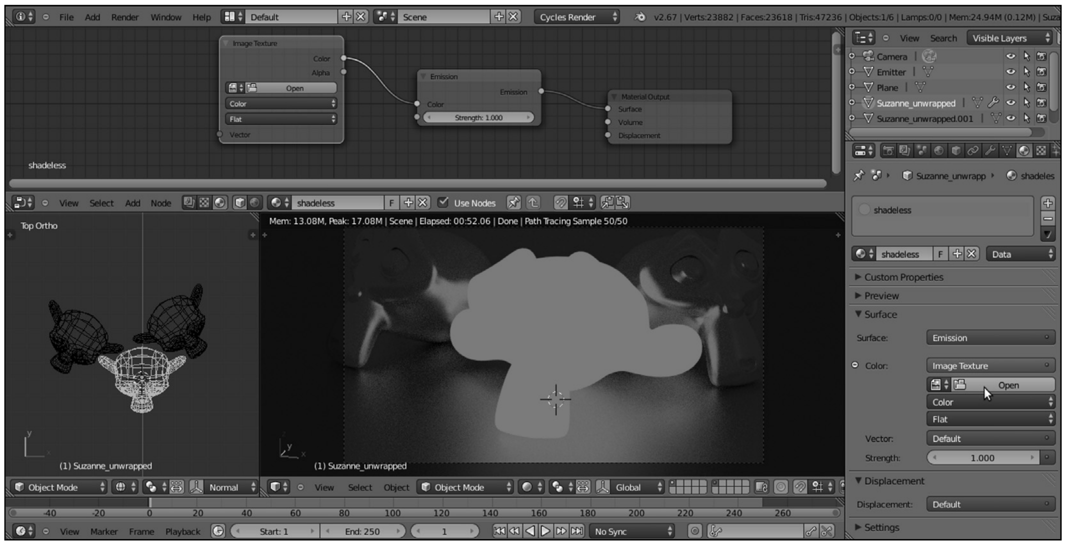
Start Blender and open the file `1301OS_09_start.blend`.

1. Go to the **World** window and set the **Background** strength to `0.200`.
2. Select the **Emitter** Plane and set the **Strength** to `0.100`.
3. Select the **Plane** object, and in the **Material** window under the **Properties** panel to the right switch the **Diffuse BSDF** shader with a **Glossy BSDF** shader node.
4. To make the next image more readable, set the glossy **Roughness** value to `0.200`.
5. Select the Suzanne mesh and click on the **New** button in the **Node Editor** window header or in the **Material** window under the **Properties** panel to the right.
6. Set the **Camera** view in **Rendered** mode to have an immediate visual feedback.
7. Press *Shift* + *D* to duplicate the Suzanne mesh and move it to the left of the scene, rotating it to accommodate it close to the original one. Click on the **2** number on the side of the **Material** name in the **Node Editor** window header to make it **single-user**.
8. In the **Material** window switch the **Diffuse BSDF** node with a **Mix Shader** node; in the first **Shader** slot select a **Diffuse BSDF** node and in the second one a **Glossy BSDF** shader. Set the **Glossy** node's **Roughness** to `0.100` and the **Fac** value of the **Mix Shader** node to `0.800`. Rename the material `mirror`.
9. Press *Shift* + *D* to duplicate the **mirror** Suzanne mesh and move it to the right of the scene, close to the original one.

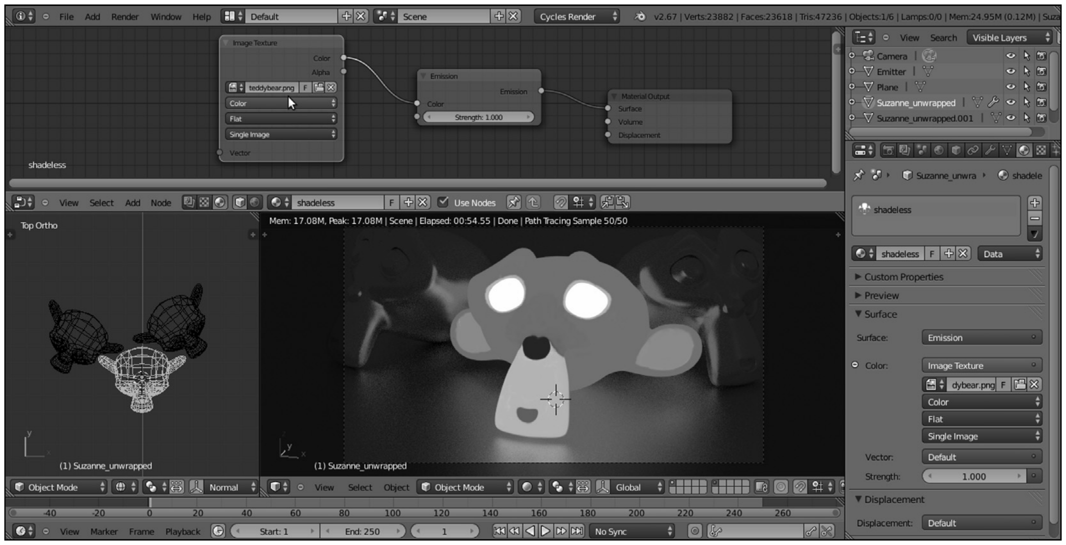
How to do it...

Let's start with the first material method:

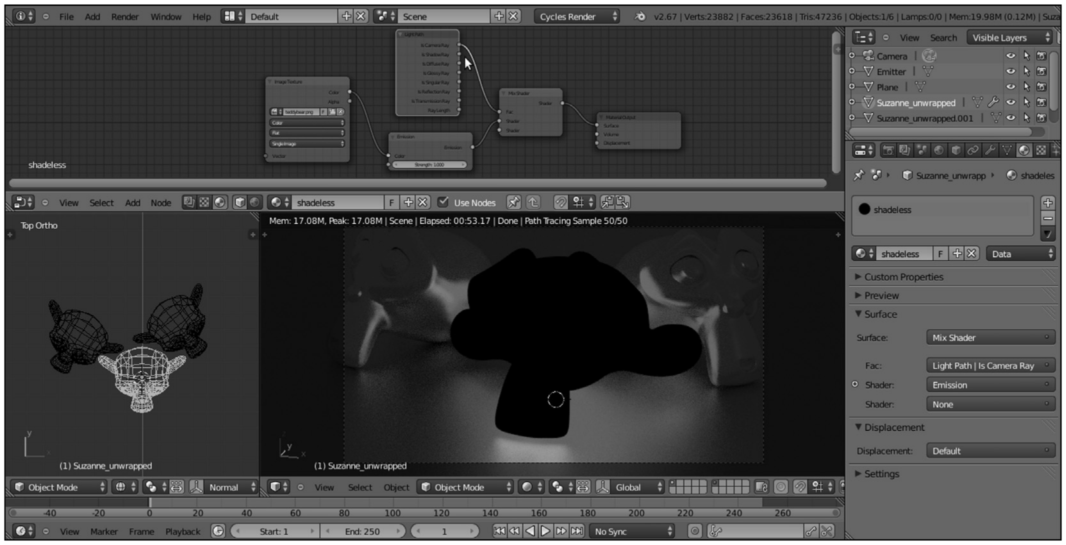
1. Select the original Suzanne mesh and, in the **Material** window, switch the **Diffuse BSDF** shader with an **Emission** shader. Add an **Image Texture** node (press *Shift* + *A* and go to **Texture** | **Image Texture**) and connect its **Color** output to the **Color** input socket of the **Emission** node; rename the material `shadeless`.
2. In the **Rendered Camera** view the original Suzanne mesh is actually emitting a pink light on the scene, because there is no image texture loaded at the moment:



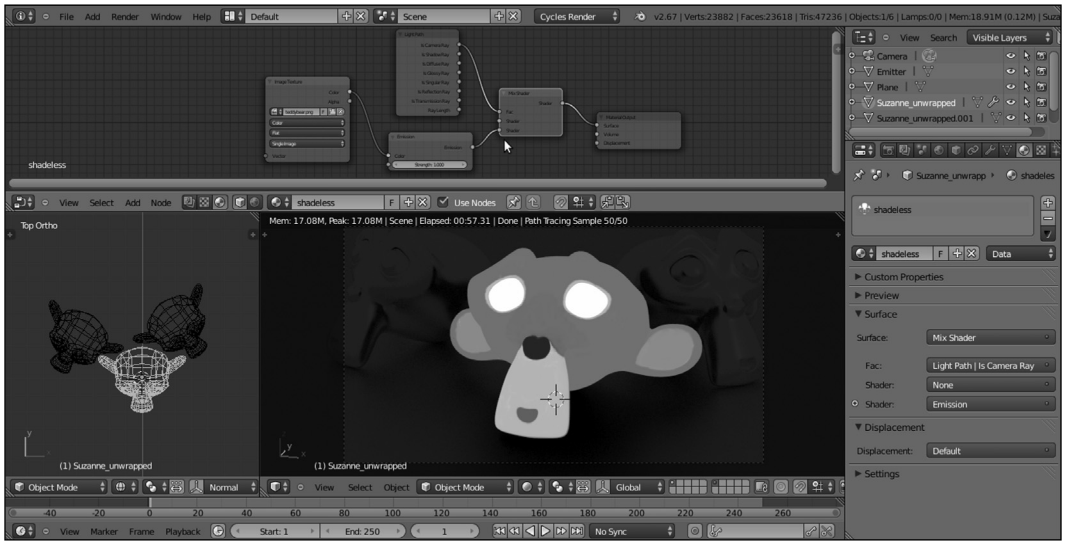
3. Click on the **Open** button of the **Image Texture** node, browse to the `textures` folder and load the image `teddybear.png`.



4. In the **Node Editor** window add a **Mix Shader** node (press *Shift* + *A* and go to **Shader** | **Mix Shader**) and paste it between the **Emission** node and the **Material Output** node.
5. Add a **Light Path** node (press *Shift* + *A* and go to **Input** | **Light Path**) and connect the **Is Camera Ray** output to the **Fac** input socket of the **Mix Shader** node. The Suzanne mesh turns totally black (equals no material) but is still lighting the scene:

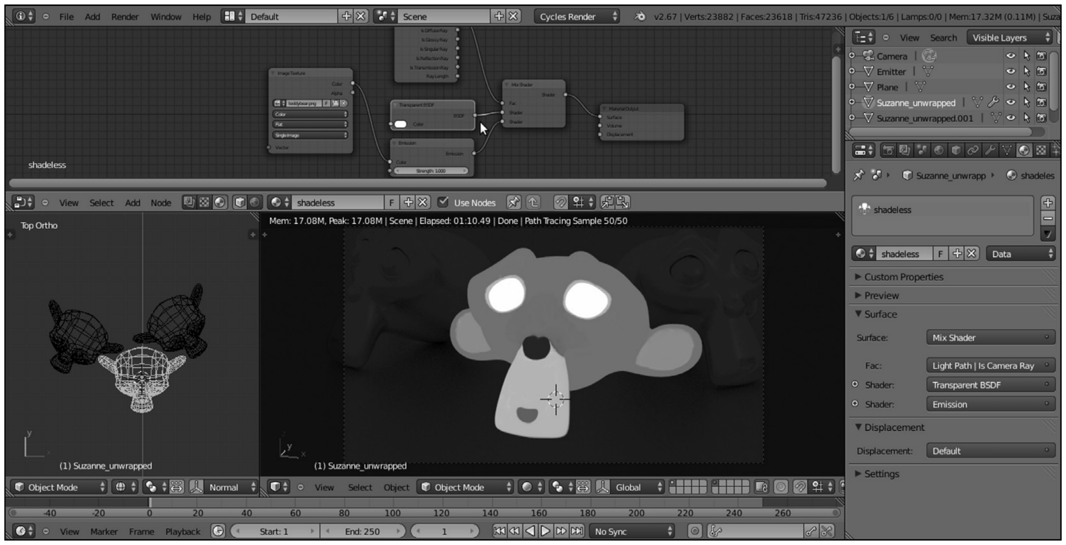


6. Switch the connection of the **Emission** node from the first **Shader** input socket of the **Mix Shader** node to the second **Shader** input socket:



At this point, the shadeless Suzanne is still affecting the surrounding objects; for example, it's reflected as a black object by the floor and by the mirror Suzannes; in fact, the output of the first empty input socket of the **Mix Shader** node is a black color, because there is no material at all. To make it not reflectable:

7. Add a **Transparent** shader node (press *Shift* + *A* and go to **Shader** | **Transparent BSDF**) and connect it to the first **Shader** input socket of the **Mix Shader** node.



How it works...

Thanks to the **Is Camera Ray** output of the **Light Path** node, all the light rays emitted by the **Camera** and directly hitting the Suzanne mesh are rendered with the **Emission** material brightness (because this material has a value equal to 1, being connected to the second **Shader** socket of the **Mix Shader** node), while for the other kind of rays (reflected, transmitted,..., where the first socket value equals 0) there is no emitting material coming from the Suzanne mesh. Actually, at first there is no material at all and this gives us a black reflected Suzanne; to avoid the black reflections, a **Transparent BSDF** shader has been connected to first socket of the **Mix Shader** node.

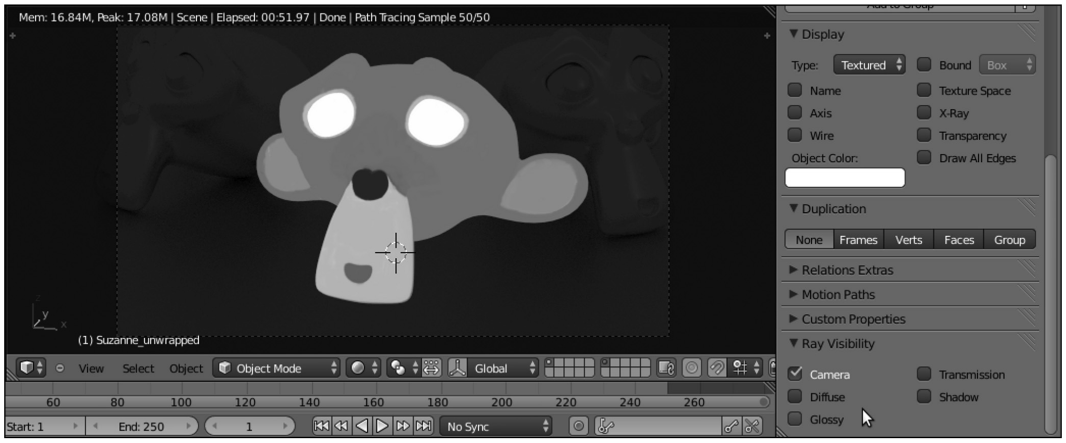
There's more...

The second method to obtain a shadeless object is:

1. Starting from the preceding file, select the Suzanne mesh and in the header of the **Node Editor** window click on the **F** button on the right side of the material name **datablock** to assign a **Fake User** (this is to keep the material saved in the blend file even if not assigned to anything). Then click on the **X** icon (**unlink datablock**).
2. Now click on the **New** button to create a new material. In the **Material** window under the **Properties** panel switch the **Diffuse BSDF** with an **Emission** shader node.
3. In the **Node Editor** window add an **Image Texture** node (press *Shift* + *A* and go to **Texture** | **Image Texture**) and connect its **Color** output to the **Color** input socket of the **Emission** shader.
4. Click on the **Open** button on the **Image Texture** node to load the image texture `teddybear.png`.

   We are now at the same point as at step 3 of the first method; we have created a light emission material based on the texture mapped on the Suzanne mesh.

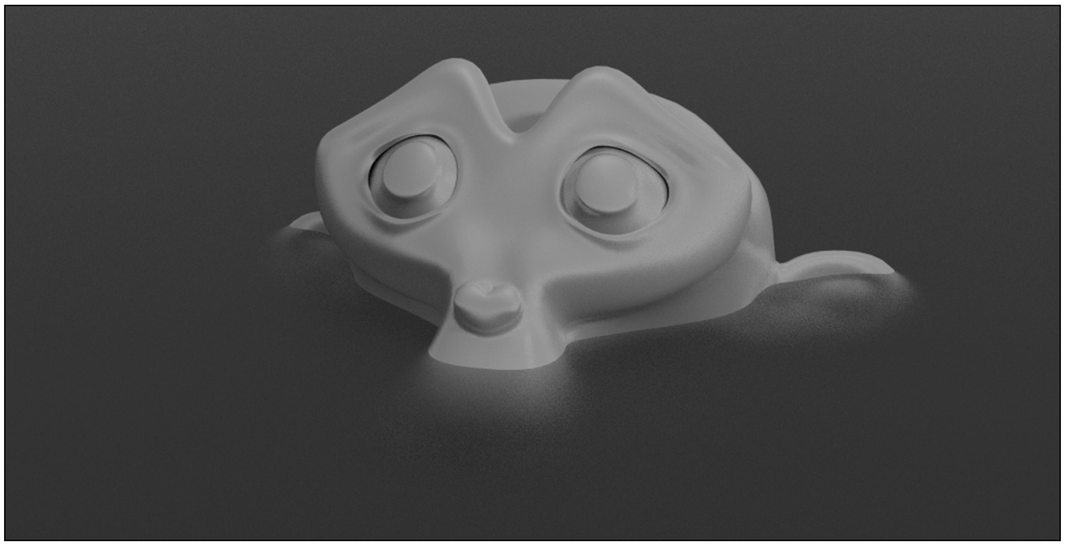
5. Now go to the **Object** window under the **Properties** panel in the **Ray Visibility** tab (usually the last at the bottom) and uncheck the **Diffuse**, **Glossy**, **Transmission**, and **Shadow** items.



So, basically, only the **Camera** item is active now. Simple, quick, and effective!

Creating a fake immersion effect material

In this recipe we will create a material that gives the effect of an object immersed into a substance that becomes more and more opaque as the depth increases, as for example murky water:



Getting ready

Start Blender and open the file `1301OS_09_start.blend`.

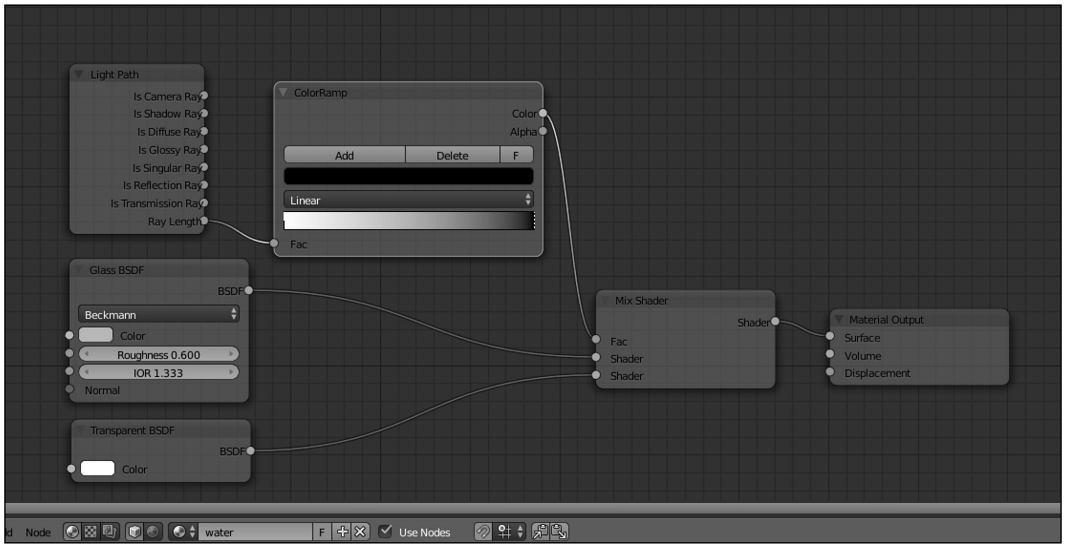
1. Go to the **World** window and click on the little square with a dot on the right side of the **Color** slot. From the menu select **Sky Texture**. Set the **Strength** value to `0.500`.
2. Select the plane, rename it **water** and move it to **Location Z: 1.17000**; then press *Shift* + *D* to duplicate it, rename it `bed` and move it to **Location Z: -2.00000**.

How to do it...

Let's go with the creation of the different materials:

1. Go to the **Material** window and select the Suzanne mesh; in the **Node Editor** window header click on the **New** button and rename the material `Suzanne`. In the **Material** window switch the **Diffuse BSDF** shader with a **Mix Shader** node; in the first **Shader** slot select a **Diffuse BSDF** shader node and in the second one a **Glossy BSDF** shader node.
2. Set the **Glossy** node's **Roughness** to `0.100` and the **Fac** value of the **Mix Shader** node to `0.600`.
3. Select the **bed** plane and click on **New** in the **Node Editor** window header; rename the material `bed`.
4. In the **Material** window switch the **Diffuse BSDF** shader with an **Emission** shader node; set the **Color** to `R 0.800`, `G 0.659`, and `B 0.264` and the **Strength** to `0.100`.
5. Select the **water** plane and click on the **New** button in the **Node Editor** window header; rename the material `water`.
6. In the **Material** window switch the **Diffuse BSDF** shader with a **Mix Shader** node; in the first **Shader** slot select a **Glass BSDF** shader node and in the second one a **Transparent BSDF** node.
7. Set the **Glass BSDF** node's **Roughness** to `0.600`, the **IOR** value to **1.333** and the **Color** to `R 0.185`, `G 0.611`, and `B 0.800`.
8. Add a **Light Path** node (press *Shift* + *A* and go to **Input** | **Light Path**) and a **ColorRamp** node (press *Shift* + *A* and go to **Convertor** | **ColorRamp**); connect the **Ray Length** output to the **Fac** input socket of the **ColorRamp** node and invert the position of the black and white color markers (that is, move the black slider all the way to the right and the white one to full left).

9. Connect the **ColorRamp** output to the **Fac** input socket of the **Mix Shader** node:



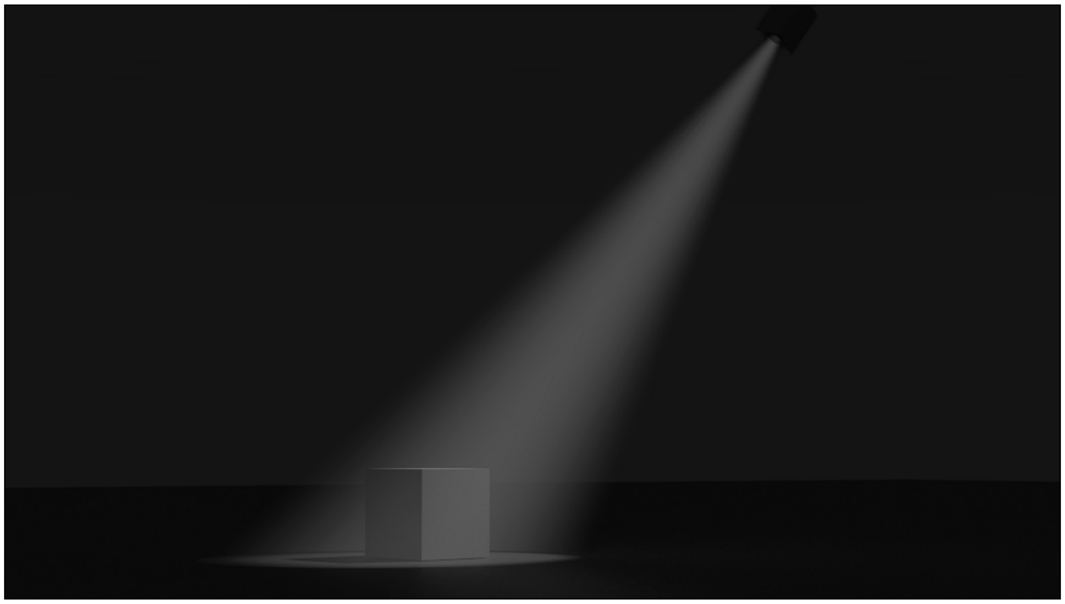
How it works...

The effect being applied to the **water** material is due to the **Ray Length** output, which returns the length of the light rays passing through an object, basically giving the thickness of that object. In our case this is, the distance from the water mesh surface to the far distance (from the **Camera** point of view, because in a path tracer the light rays are shot from the **Camera**). The gradient of the **ColorRamp** node is mapped on the length of this **Ray Length** output (which is also clamped and inverted by the same **ColorRamp** node), connected to the **Fac** input socket of the **Mix Shader** node so to work as a stencil map that smoothly blends the effects of the **Glass** and **Transparent** shaders.

The transition from the **Transparent** shader to the **Glass** shader gives the impression of a volume of water becoming more and more murky as the distance from the surface increases.

Creating a fake volume light material

In this recipe we will create a material that fakes the typical effect of a cone of light visible because of passing through the dust suspended in the air, or coming from the sky in a cloudy day (the so-called God's rays) We will use a mesh simulating a volumetric cone of light, rather than a real light to be used to light the scene. Instead, as in the already made blend file that we start from, a matching Lamp must be set for the real lighting.



Getting ready

Start Blender and open the file `1301OS_09_volumelight_start.blend`, which contains a preset scene with a cone mesh, a spot mesh, and a spot lamp parented to the cone mesh. The spot lamp cone follows the shape of the cone mesh and its purpose is to light the cube leaning on the ground Plane.

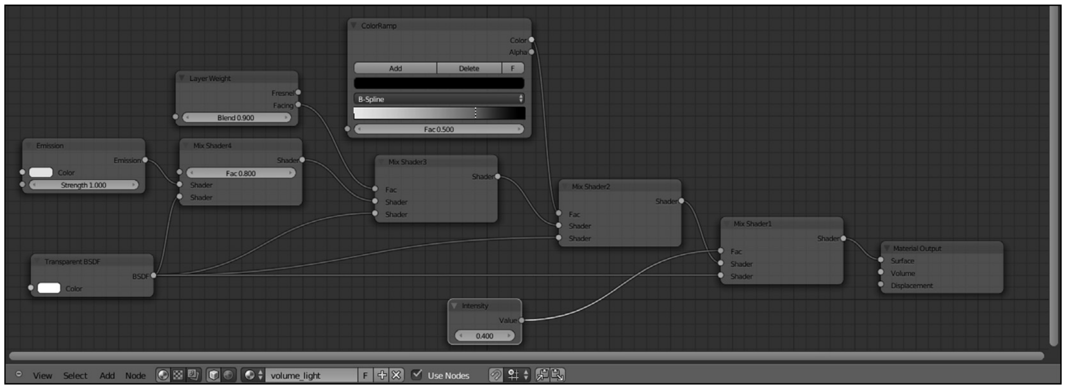
The cone and the spot meshes also have a brief animation to point the cone of light to the cube.

How to do it...

Let's go with the light cone material:

1. Select the **volume_light** mesh and click on the **New** button in the **Node Editor** window header or in the **Material** window under the **Properties** panel to the right; rename the material `volume_light`.
2. In the **Material** window switch the **Diffuse BSDF** shader node with a **Mix Shader** node; go to the **Active Node** panel to the right of the **Node Editor** window (if not already present press *N* to make it appear) and in the **Label** slot rename the **Mix Shader** as `Mix Shader1`.
3. In the first **Shader** slot of the **Mix Shader1** node select a new **Mix Shader** node, renaming it `Mix Shader2`; in the second **Shader** slot select a **Transparent BSDF** shader node.

4. Go to the **Mix Shader2** node and in the first **Shader** slot select again a **Mix Shader** node, renaming it `Mix Shader3`; connect the output of the **Transparent BSDF** node to the second **Shader** slot.
5. Go to the **Mix Shader3** node and in the first **Shader** slot select a last **Mix Shader** node, renaming it `Mix Shader4`; connect the output of the **Transparent BSDF** node to the second **Shader** slot.
6. Go to the **Mix Shader4** node and in the first **Shader** slot select an **Emission** node; connect the output of the **Transparent BSDF** node to the second **Shader** slot of the **Mix Shader4** node.
7. Set the **Color** of the **Emission** node to `R 0.769`, `G 0.800`, and `B 0.592` and the **Fac** value of the **Mix Shader4** to `0.800`.
8. Add a **Layer Weight** node (press *Shift* + *A* and go to **Input** | **Layer Weight**) and connect its **Facing** output to the **Fac** input socket of the **Mix Shader3** node; set the blend value to `0.900`.
9. Add a **ColorRamp** node (press *Shift* + *A* and go to **Convertor** | **ColorRamp**) and connect its **Color** output to the **Fac** input socket of the **Mix Shader2** node. Set the interpolation to **B-Spline** and move the black color marker three-quarters to the right of the total length of the slider; move the white color marker to the full left.
10. Add a **Value** node (press *Shift* + *A* and go to **Input** | **Value**), rename it **Intensity** and connect it to the **Fac** input socket of the **Mix Shader1** node. Set the value to `0.400`.



How it works...

The effect of the light blending into the night is obtained by the various blending factors of the **Mix Shader** nodes, that drive the mixing of the **Emission** shader with the **Transparent** one. The **Value** node connected to the **Fac** input of the **Mix Shader1** node establishes the intensity of the fake volumetric light. A value of 1.000 will turn it off completely (be careful not to go beyond 1.000, otherwise the cone mesh will show up as a dark silhouette), and values towards 0.000 or even negative ones make it appear more and more intense.