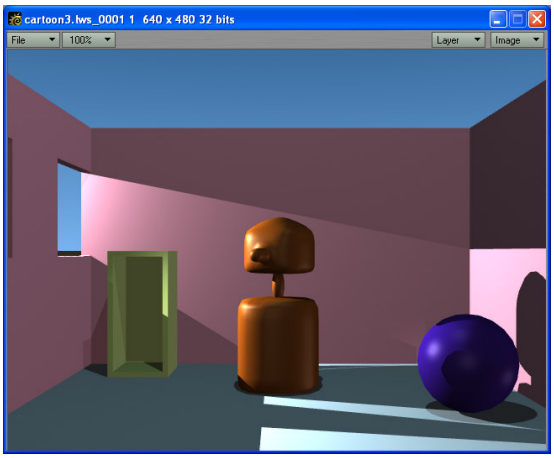
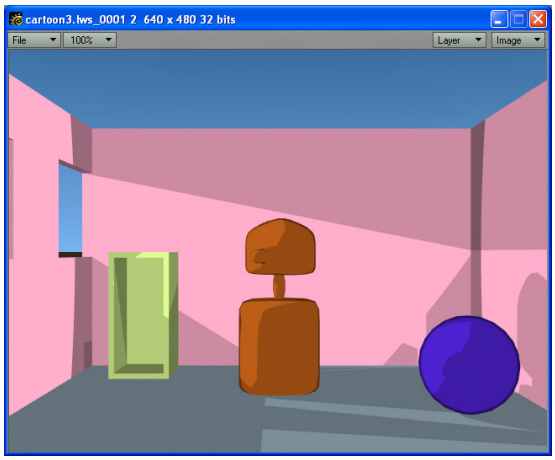
Using Amelie to render as a cartoon filter

In this tutorial, we will explain how to use Amelie in order to give a cartoon apperance to your renders.

On the left side you see the normal LW render and on the right side the result after applying Amelie as a “Pixel Filter”.



Without Amelie



With Amelie

You may apply the same filter to any image you want, with some restrictions. Soft shadows are not well supported, and textures should be as simple as possible (in our example we used only plain colors).

How to start ?

In order to render as a cartoon we need to do three different things:

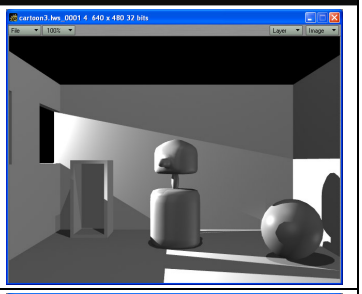
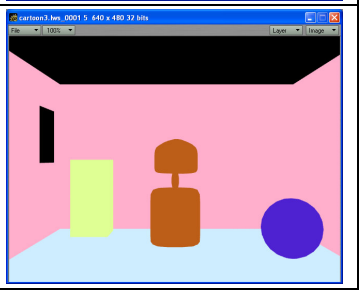
1. Choose the color and intensity of each pixel depending of the illumination and the object color.
2. Draw some sort of border around the object.
3. Display the background without modification.

Any of those three things can be done first as at the end we will merge the three parts in order to get the final result.

Pixel illumination

Let's start with the pixel color and intensity. With this first part, we will get the following result:

We need to check if the pixel gets some light or not and depending of the intensity of the light each pixel receive we will decide which shade of the original color we need to apply.

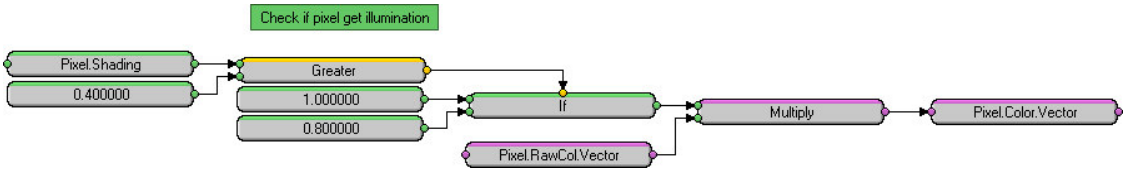
Needed elements:	
The element "Pixel.Shading" allows to get the illumination (with shadows, and highlights) of each pixels of the image.	
The element "Pixel.RawCol.Vector" returns the color of the pixel without any shading.	

With these two elements, we will be able to get the desired effect for the first part of our filter.
What we need to do is to check the intensity of the shading and if the shading is greater than a certain number (in our example 0.4) then we decide that we take bright shading, if not we take a dark shading.

Therefore, if we should write a pseudo code for it would be something like that:

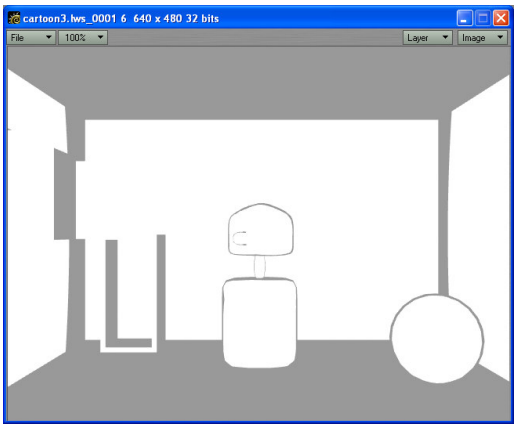
```
If Pixel.Shading > 0.4 then
      Pixel.Color.Vector=Pixel.RawCol.Vector*1
Else
      Pixel.Color.Vector=Pixel.RawCol.Vector*0.8
End if
```

This is how we did it with Amelie.



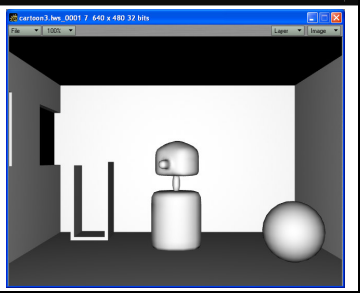
Object border

Now we will try to show object borders. With this first part, we will get the following result:



Needed element:

The element "Pixel.Geometry" allow to see if a pixel is facing the camera or not. A "white" (1.0) pixel means the pixel is in facing the camera.



Just checking the value of this element, we will be able to see some borders (on round objects). Border of cubic objects or planes will not be worked out correctly but it doesn't matter too much.

Again, we will simply check if this element is over or under a certain number and act accordingly to that.

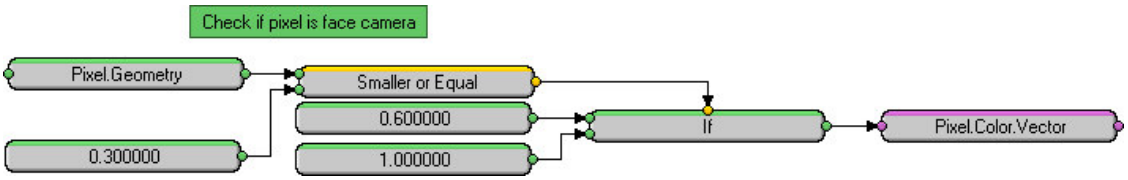
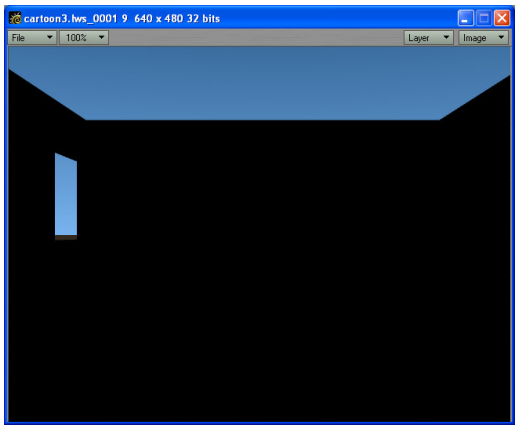


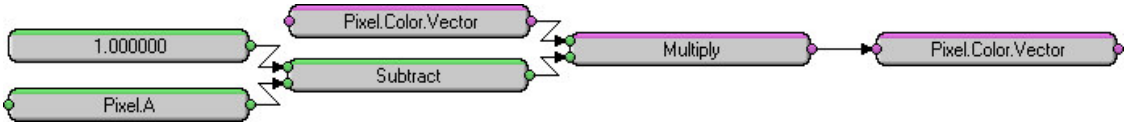
Image background

Now we need to keep the background of our image and display it without modification.



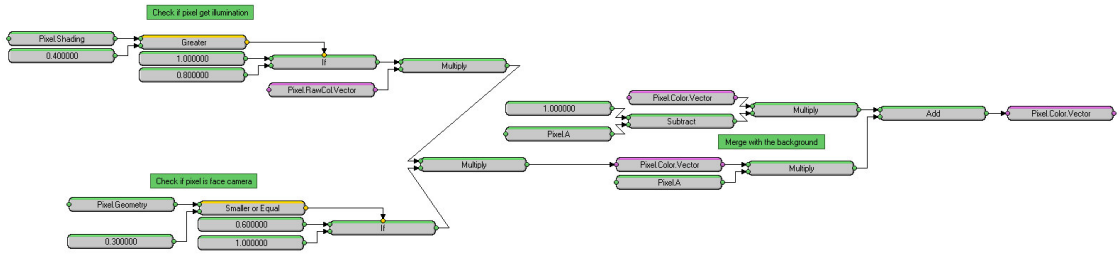
In order to do that, the easiest way is to check/use the Pixel.Alpha value. We have done it with the following formula:

```
Pixel.Color.Vector=Pixel.Color.Vector*(1-Pixel.A)
```



Merge all

Now we need just to merge all those things together and we will reach our goal:



How did we reach this result

In order to solve a problem like the one explained here, you need first to split the goal in sub-goals in order to resolve them one by one and have easier things to solve. As you have seen this cartoon look is composed in this case by using three sub-goals, and therefore we can solve them independently. The next step is to try to solve the sub-goal. In order to do this, you should have a good knowledge of Amelie capabilities, and the best way to check them is to try each element one by one. What I mean is in the pixel filter example; you should check what give you each pixel filter specific elements. As you have seen once you know which elements you may use, it simply matter of mix them.