A tutorial on NPR rendering using Blender's new Freestyle renderer

Author: T.K.

Last modified: November 11, 2008

Introduction

The present document describes how I created an experimental movie clip (Fig.1) using Blender and its new Freestyle renderer. Blender is a powerful software package for creating 3D computer graphics and video games. Freestyle is a non-photorealistic (NPR) renderer, originally developed by an academic research group independently of Blender but now in the process of full integration into Blender, thanks to a Google Summer of Code 2008 project by Maxime Curioni. I was one of many Blender users eagerly waiting for the Freestyle integration, so the breaking news in early September saying a usable version of Blender with the Freestyle renderer was released was really astonishing. The outcome of my first trial with the new renderer was just amazing. The experimental movie clip was made during a subsequent, in-depth examination of the new renderer. The movie clip is available online at http://vimeo.com/1978456.

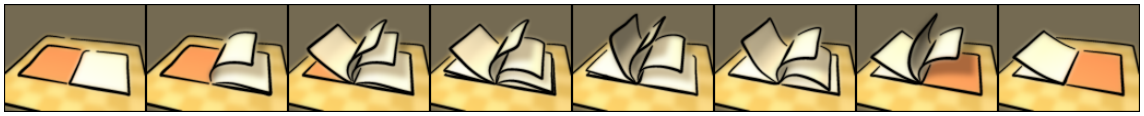
Figure 1: A still image from the experimental movie clip.

The purpose of the document is to present the basic ideas that were necessary to produce the final appearance of the movie clip. To achieve a desired appearance, I used composite nodes and the video sequence editor in Blender, as well as the action editor and NLA editor for basic 3D CG animation. A step-by-step explanation of how to use these features would be too lengthy to put down here, so I will just present a brief summary of what I did using these features. Those readers who are not familiar with these features should consult the Blender User's Documentation available online at http://wiki.blender.org.

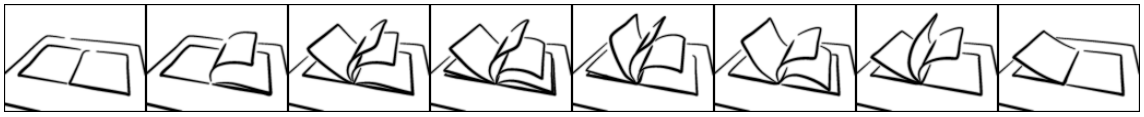
As of this writing, the integration of Freestyle into Blender is in an early stage, which implies the existence of bugs and missing features; you need to avoid them with care. An important missing feature in the current state of the Freestyle integration is a support for render layers that would allow you to composite multiple images (or sequences of images) separately rendered by the Freestyle renderer and/or Blender's internal renderer. To supplement this missing functionality, I used the video sequence editor as described later.

The version of the software I used is Revision 16866 of the Freestyle branch. I made a Win32 build of my own, which is available at GraphicAll.org in the following location: http://www.graphicall.org/builds/builds/showbuild.php?action=show&id=826

The movie clip (the final series of frames)



The drawing part



The painting part

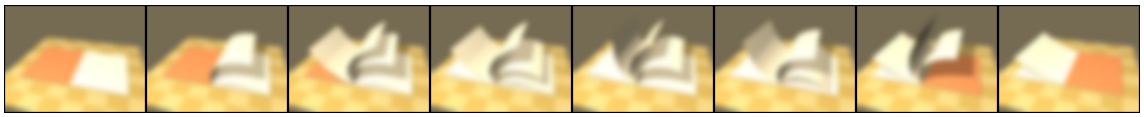


Figure 2: The composition of the movie clip.

The composition of the movie clip is shown in Fig. 2. The movie clip consists of a series of frames, each of which is a composite image made of 2 still images. That is, 2 sequences of still images were separately rendered one after another and composed into the final series of frames to make the movie clip. One image sequence (referred to as the drawing part) shows black and white brush strokes, while the other (referred to as the painting part) gives a colorful tone to the movie clip.

In short, the preparation of the final series of frames takes the following 4 steps:

1. Prepare the 3D scene.
2. Render the drawing part.
3. Render the painting part.
4. Add blur effects to both parts.
5. Composite the two parts.

Let's take a look at these steps one by one.

Preparing the 3D scene

First of all, I prepared a 3D scene by Blender, in which an opened book is put on a table. The book consists of 2 plane objects for the cover and another 10 plane objects for the pages. Each page has an Armature object (consisting of 5 bones) as the parent to define the motion of the page as if it were turned by a wind. Figure 3 shows the 3D scene being edited in Blender, where the bones (pink) attached to the pages are highlighted.

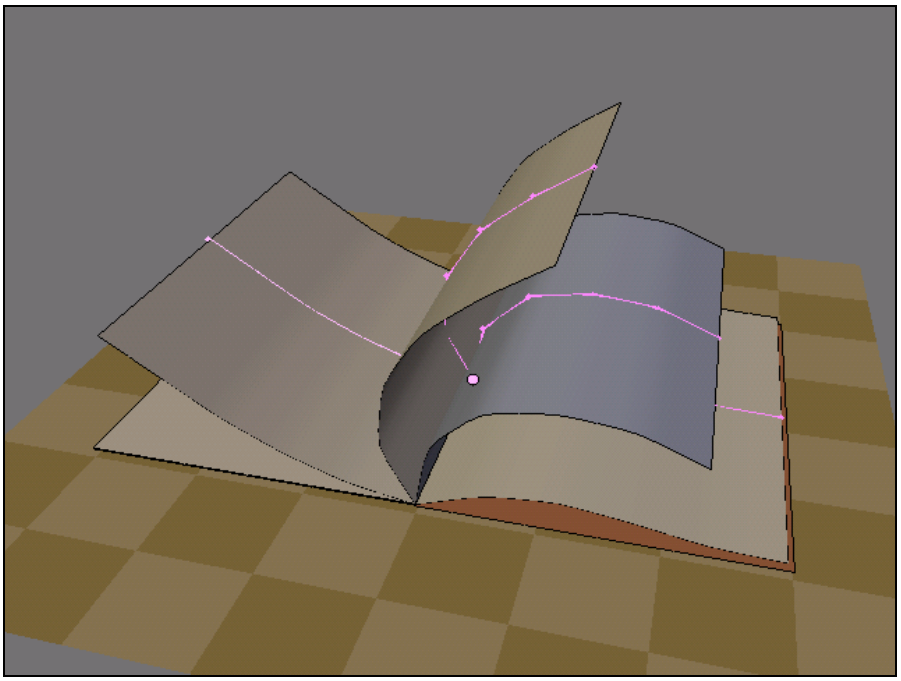


Figure 3: The bones to define the motion of the pages.

All pages have the same motion with different start frame numbers specifying when the motion begins. This is done as follows. I first defined the motion of the first page using the IPO editor and created an action named AC:TurnOver (Fig. 4). Then using the NLA editor, I associated the action with each of the remaining 9 pages with different start frame numbers (Fig. 5).

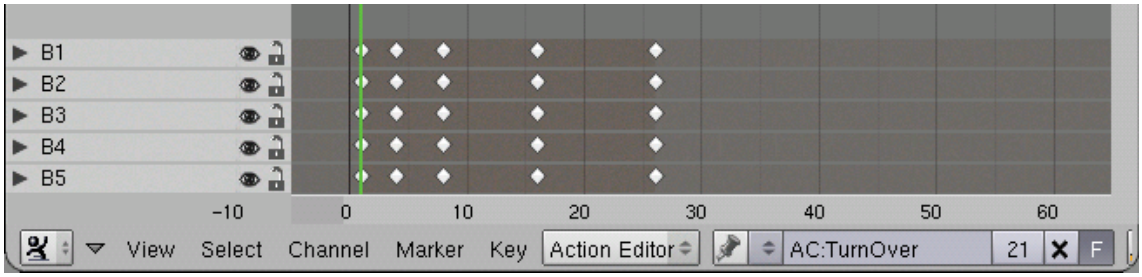


Figure 4: A screenshot of the action editor. B1 through B5 are the names of the 5 bones that comprise the Armature object defining the motion of the first page.

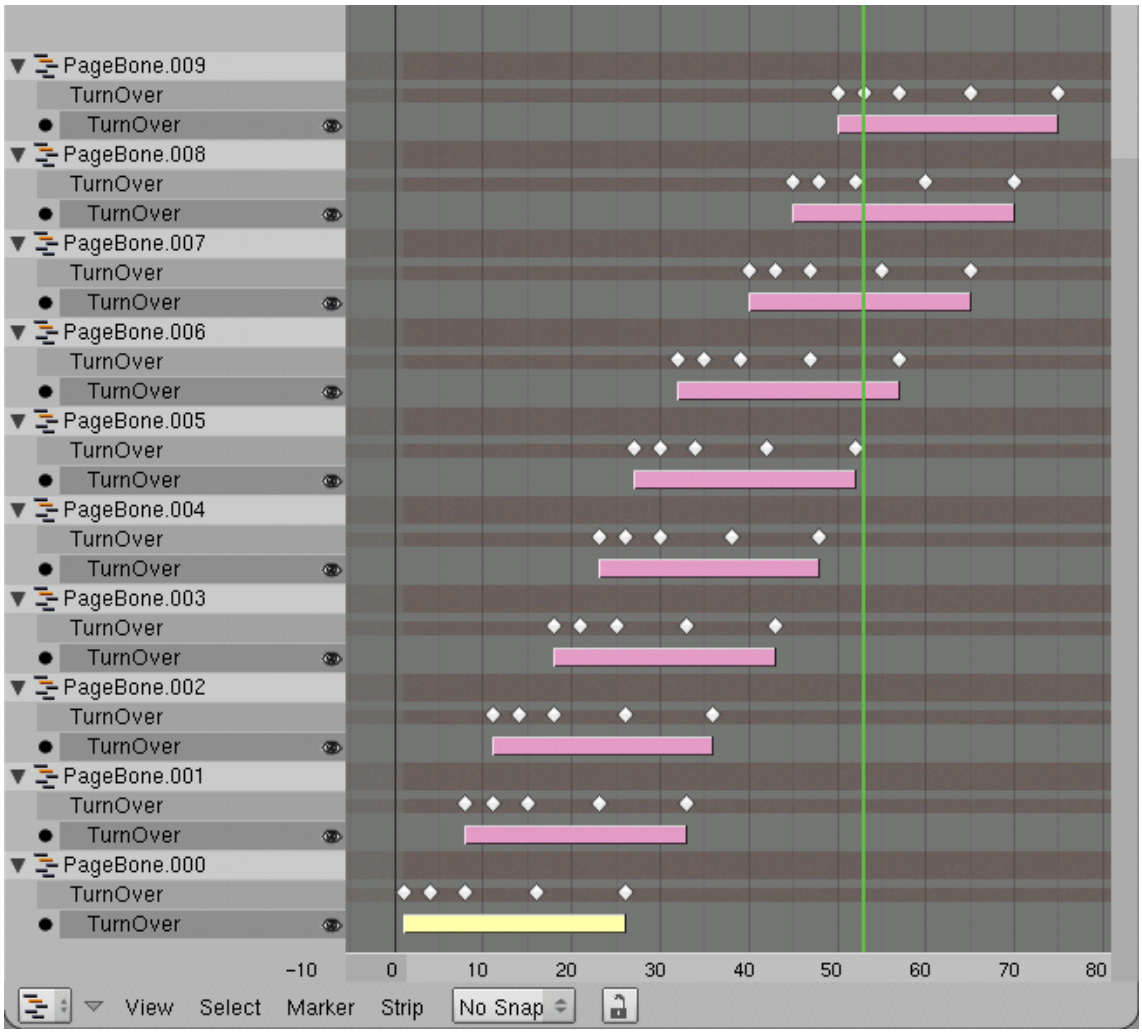


Figure 5: A screenshot of the NLA editor. Armature objects have the names PageBone.000 through PageBone.009. The motion of each Armature object is defined by an action strip named TurnOver, which is in turn defined by the action AC:TurnOver.

Rendering the drawing part

Now let us begin playing with the Freestyle renderer. It is available in the Rendering Engine menu in the Render tab in the Scene panel (Fig. 6). By selecting Freestyle from the menu, the Freestyle tab will show up (Fig. 7), in which you specify a style module that describes a style of strokes in terms of the Python scripting language. Predefined style modules are available in the .blender/scripts/freestyle/style_modules directory (the exact location of the .blender/ directory is platform dependent. In the case of mine, it resides in the same directory with blender.exe).

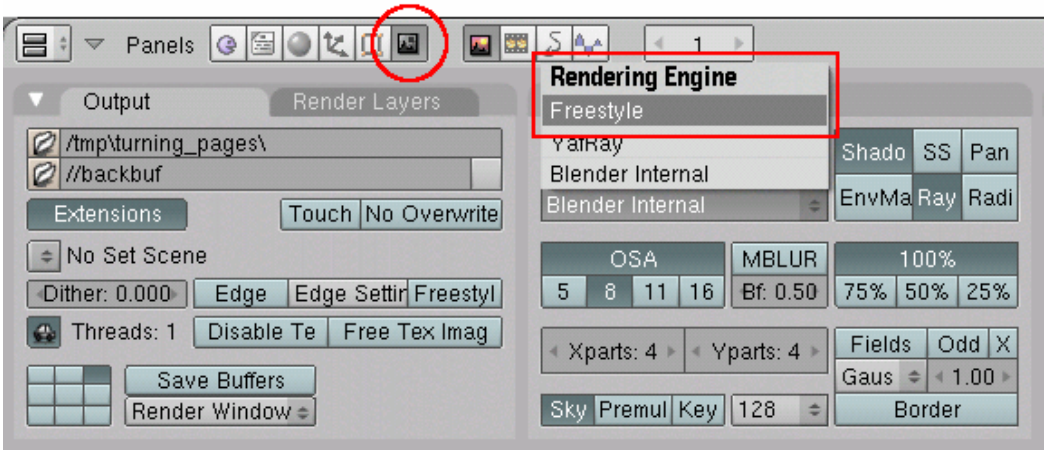


Figure 6: The Rendering Engine menu in the Scene panel.

I gave a try to a predefined style module named japanese_bigbrush.py. You can enable it by pressing the folder-like icon in Fig. 7 and selecting the style module file in the style_modules directory.

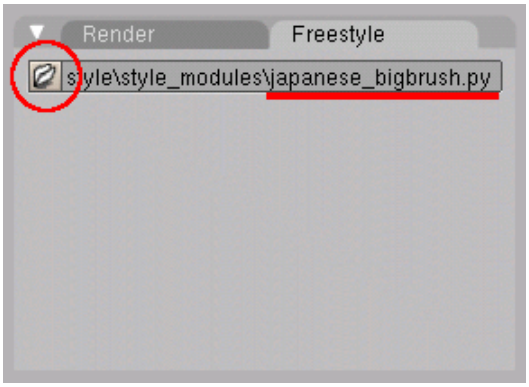


Figure 7: The Freestyle tab in the Scene panel.

Now let's go back to the Render tab and press the RENDER button. Figure 8 shows the output of the Freestyle renderer. The rendered image looks quite interesting.

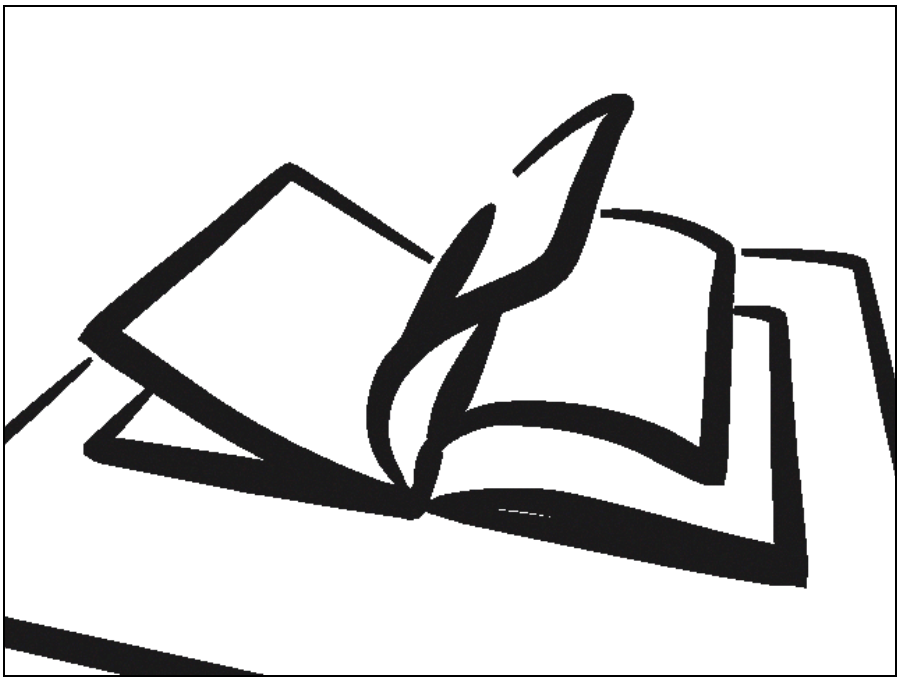


Figure 8: The output of the Freestyle renderer with predefined style module japanese_bigbrush.py (without modifications).

Here I made 2 minor changes to the style module to refine the result. One change is the stroke color. In fact, the stroke color in Fig. 8 is a very dark gray, and I noticed that this color gave an unintended translucent appearance to the strokes in the final movie clip. So I changed the stroke color to black. The other change is the width of the strokes. For me the strokes looked a bit too thick, so I made the stroke width slightly thinner.

Actual modifications of japanese_bigbrush.py were done as follows. The color is specified by ConstantColorShader in line 53. The 4 real numbers specify the stroke color in the RGBA format, so I changed them as shown below:

Original: ConstantColorShader(0.2, 0.2, 0.2, 1.0),
Modified: ConstantColorShader(0.0, 0.0, 0.0, 1.0),

The stroke width is specified by pyNonLinearVaryingThicknessShader in line 51. The stroke width is variable and thus specified by a range; the first 2 numbers specify the minimum and maximum values the stroke width can take. So, I changed them as follows:

Original: pyNonLinearVaryingThicknessShader(4, 25, 0.6),
Modified: pyNonLinearVaryingThicknessShader(2, 12, 0.6),

The result of the refinement is shown in Fig. 9. It appeared satisfactory for me, so I rendered the first image sequence (namely the drawing part) by specifying an output directory to store the image sequence of the drawing part, choosing the PNG image format, and pressing the ANIM button, all in the Scene panel.

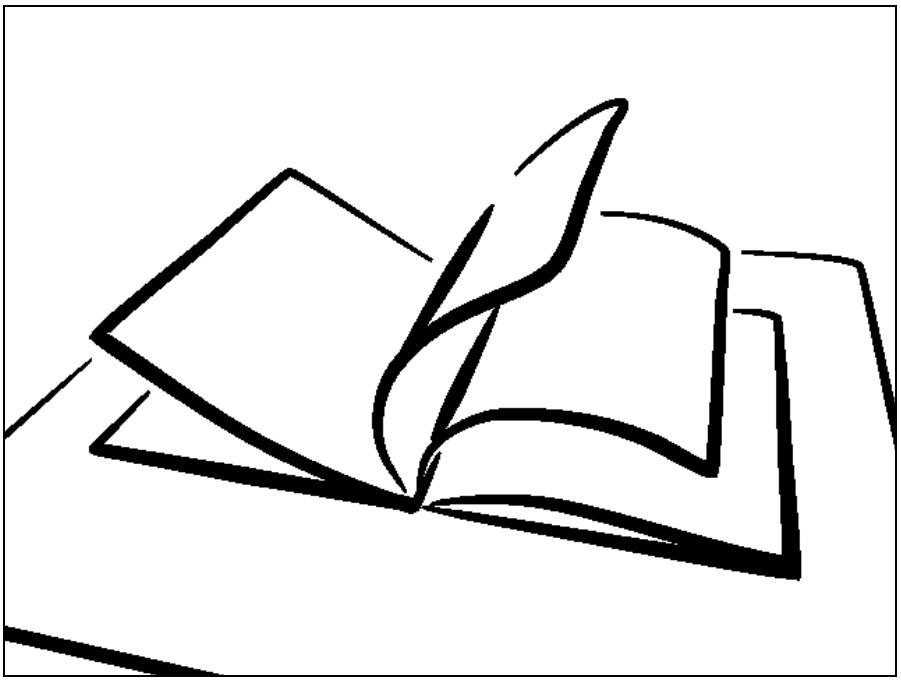


Figure 9: The output with the modified version of japanese_bigbrush.py.

There are 2 pitfalls that may trap new users of the Freestyle renderer, as described below. I believe these are implementation specific problems that will be fixed in a future version of the Freestyle branch.

- You have to choose the Render Window option from the Render Display menu in the Output tab. As far as I tested, the other options (i.e. the Full Screen and Image Editor) do not work with the Freestyle renderer.

- You also have to carefully set the Size X and Size Y in the Format tab. The width and height of a image to be rendered must be smaller than the screen size. In addition, X must be greater than Y; otherwise, the 3D scene will be rendered smaller.

Rendering the painting part

The next step is to prepare the painting part. I also made several trials and errors here to get a desired color tone. Figure 10 shows the output of Blender's internal renderer. The rendered image looked fine for me, so I made a composite image from the 2 images in Figs. 9 and 10 to get a feeling of the final result. Figure 11 shows the composite image, which was also very promising. I rendered the second image sequence (namely the painting part) with a different output directory for storing the image sequence.

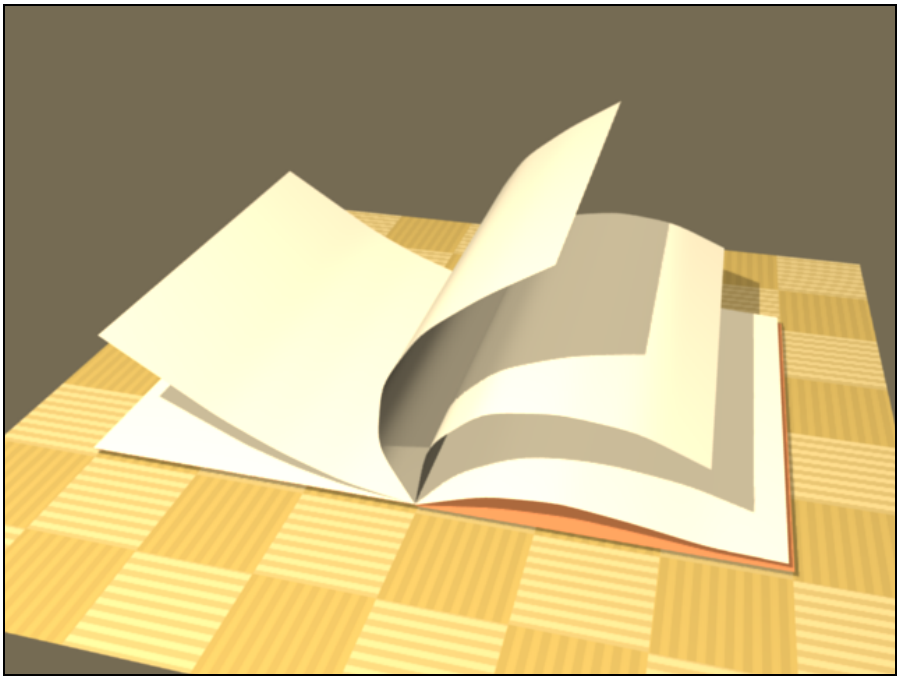


Figure 10: The output of Blender's internal renderer.

Figure 11: A composite image made of the 2 still images in Figs. 9 and 10.

Adding blur effects to both parts

At this point, I generated a complete series of composite images from the 2 parts (I will describe later how to do that) and made a preliminary movie clip to see if the final result is satisfactory. This trial figured out a problem concerning the appearance of the composite images. As you see in the image in Fig. 11, the appearance was quite sharp and strong, which resulted in flicker (i.e. a flashing effect displeasing to the eye) in the motion of the pages. Consequently, I decided to add blur effects to both the drawing and painting part.

Adding blur effects was done by composite nodes as illustrated in Fig. 12. There are 3 nodes in the node editor, namely Image, Blur and Composite nodes from left to right. The Image node supplies an image sequence as the input data to the Blur node which adds a blur effect to each input image. The X and Y parameters of the Blur node were determined through several trials so that it would solve the flicker problem. I repeated this retouch process on both of the drawing and painting parts.

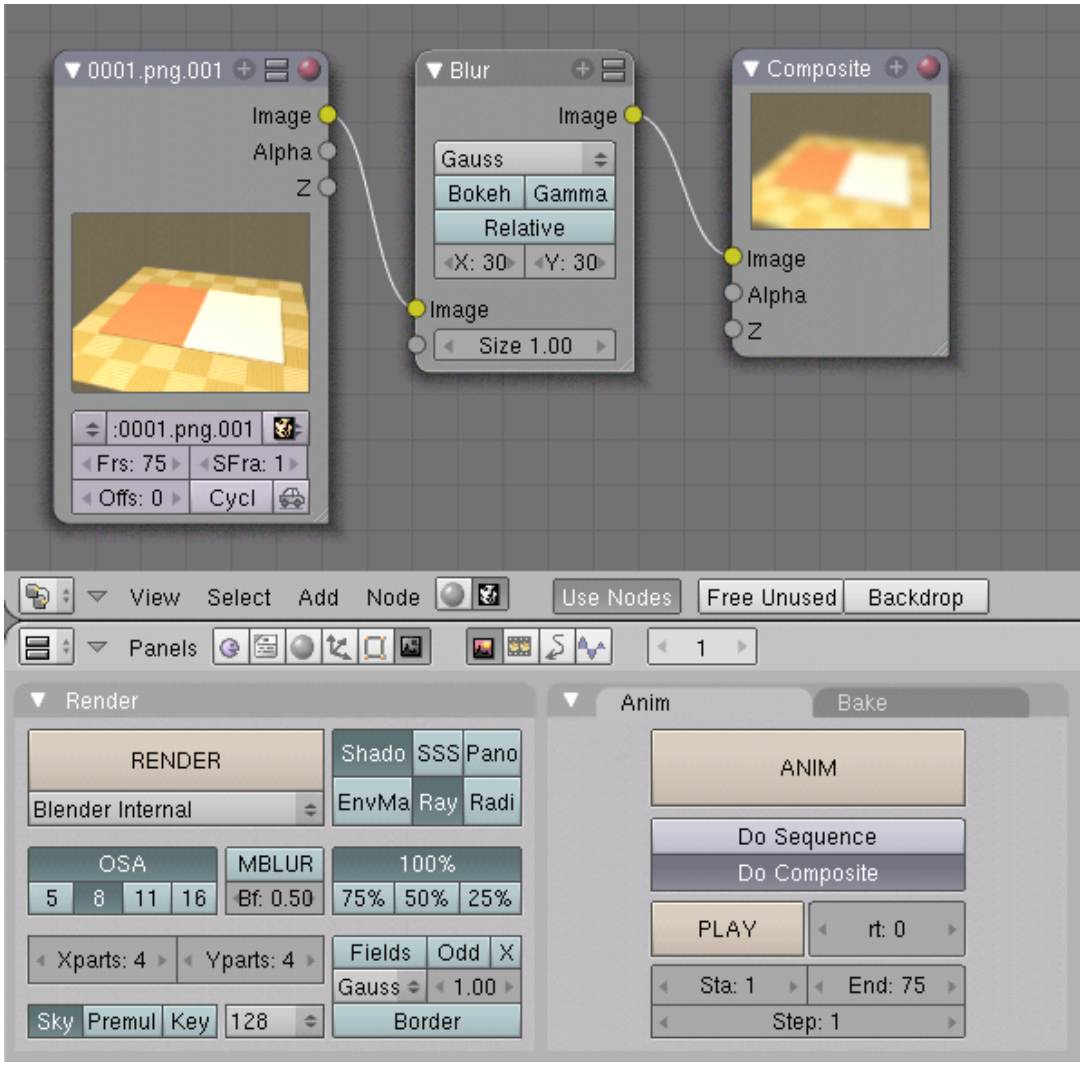


Figure 12: The addition of a blur effect using composite nodes.

Figure 14 shows the blur effects added to the drawing and painting parts using the composite nodes.

Figure 13: The drawing part (top) and the painting part (bottom) with blur effects.

Composing the final series of frames

Now that I have finished the drawing and painting parts with the blur effects, the last step is to generate the final series of composite images based on the 2 image sequences in hand. This can be done by either composite nodes or the video sequence editor. I

used the latter this time.

Figure 14 shows the setup of the video sequence editor, in which there are 3 sequence strips, namely 2 Image Sequence strips and a Mul sequence strip from bottom to top. The Image Sequence strips correspond to the 2 image sequences of the drawing and painting parts, and the Mul sequence strip is used to combine the 2 image sequences by multiplication. I also added a margin of 25 frames at both ends of each Image Sequence strip to make a pause of 1 second at the beginning and the end of the movie clip (the frame rate of the movie clip is 25 frames per second).

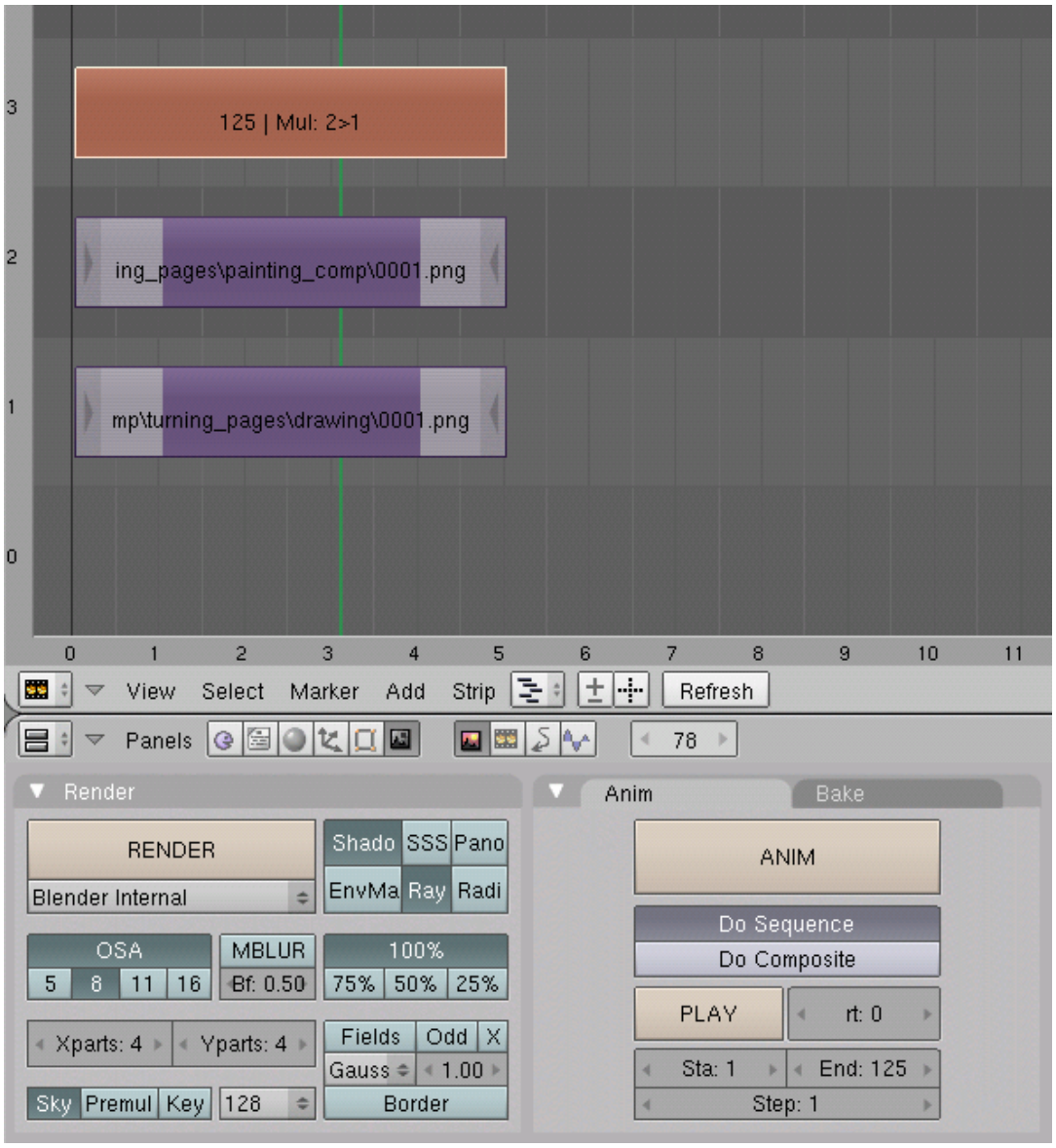


Figure 14: The composition of the final series of frames using the video sequence editor.

Figure 15 shows a frame in the final series of frames that were used to generate the movie clip. Now the appearance of the composite image looks much milder than that in Fig. 11. This appearance resulted in a flicker-free motion of the pages as I intended.

Figure 15: The final composite image done by the video sequence editor.

Summary

The composition of the movie clip is summarized as follows:

- The drawing part: A sequence of still images was rendered by the Freestyle renderer with a modified version of predefined style module japanese_bigbrush.py. A blur effect was later added to the image sequence by composite nodes to avoid flicker.
- The painting part: Another sequence of still images was rendered by Blender's internal renderer. A blur effect was also added to the image sequence using composite nodes.
- The final series of frames: The drawing and painting parts were combined using the video sequence editor to generate the final series of frames from which the movie clip was created. A Mul sequence strip was employed to composite the 2 image

sequences by multiplication.

I hope this tutorial helps new users of the Freestyle renderer. Comments and suggestions are highly appreciated.