A Comparative Look At Compositor Systems In Blender

Omar Emara

May 2, 2023

Abstract

Blender currently has three compositor systems: The Tile-Based Compositor, The Full-Frame Compositor, and The Real-Time Compositor. The consolidation of the three compositors is currently being planned, and the main concern is the differences in how they operate and which design should be adopted for the future. This document demonstrates the differences between the three compositors through a series of examples to hopefully aid in making that decision.

1 Introduction

All three compositors have nearly identical operations and node implementations, where the few differences can be attributed to performance constraints or concious decisions to deviate from existing implementation for one reason or another. So the differences in the implementations of individual operations is of no concern to us in the context of this comparison. Moreover, we have no interest in performance evaluation because such evaluation was either already demonstrated or is irrelevant due to the use of different accelerators (e.g. GPU acceleration).

The main differences between the three compositors, however, is the way they deal with image transformations and combined processing of images of different resolutions and densities. And this will be the main topic of comparison and discussion.

Each of the following sections demonstrates one of the differences using a simple example node tree, and a brief discussion and reasoning behind the differences is presented.

2 Simple Scale

This example demonstrates a simple scale operation by a factor of two on a 512×512 image where the scene size and viewport camera region size are also 512×512.

As can be seen, the Tile-based and Realtime compositors produced similar results, in which the center half of the image was interpolated to cover the output size of 512×512 due to it doubling in size.

The Full-frame compositor on the other hand produced an image of size 768×768, where $768 = \frac{3}{2}512$, and the result is the center two thirds of the image, that is $\frac{768}{512 \cdot 2}$. That's because the Full-frame compositor has the limitation where scaling is clipped past the 1.5 factor.

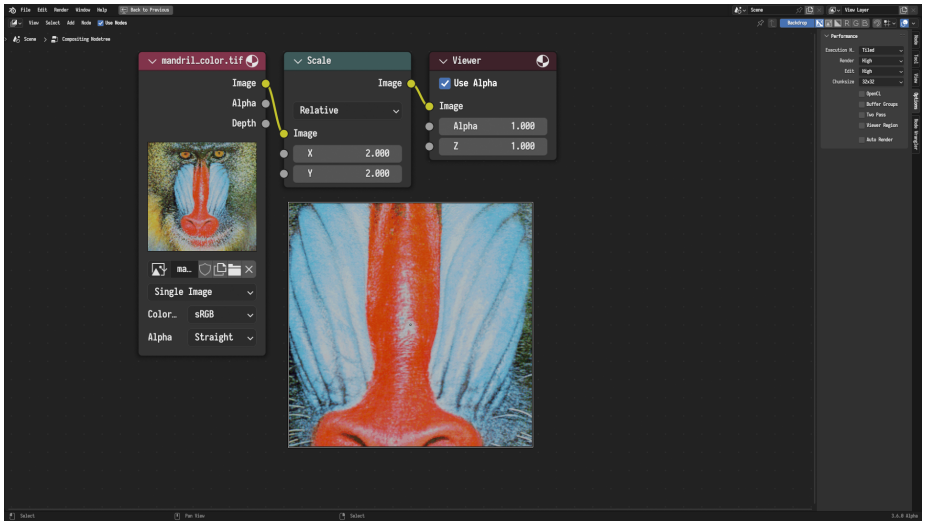

Figure 1: Tile-based compositor with simple scale.

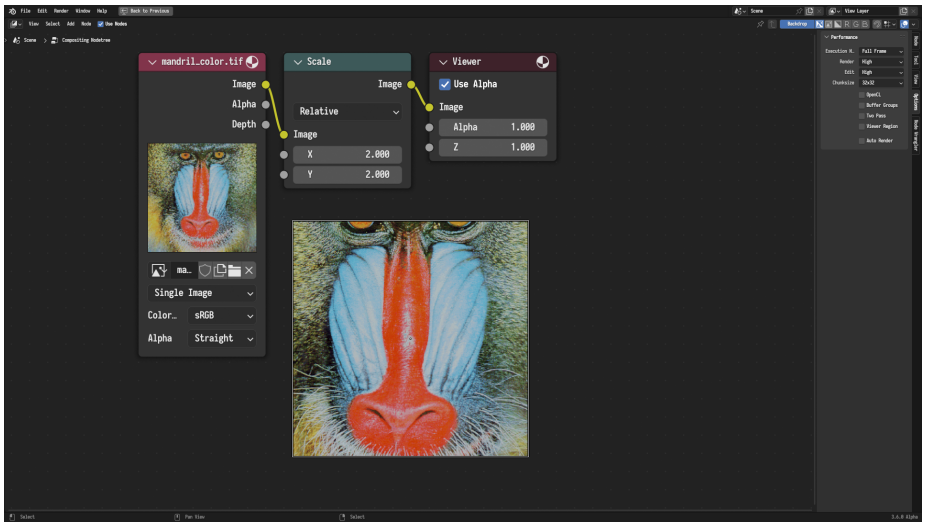

Figure 2: Full frame compositor with simple scale.

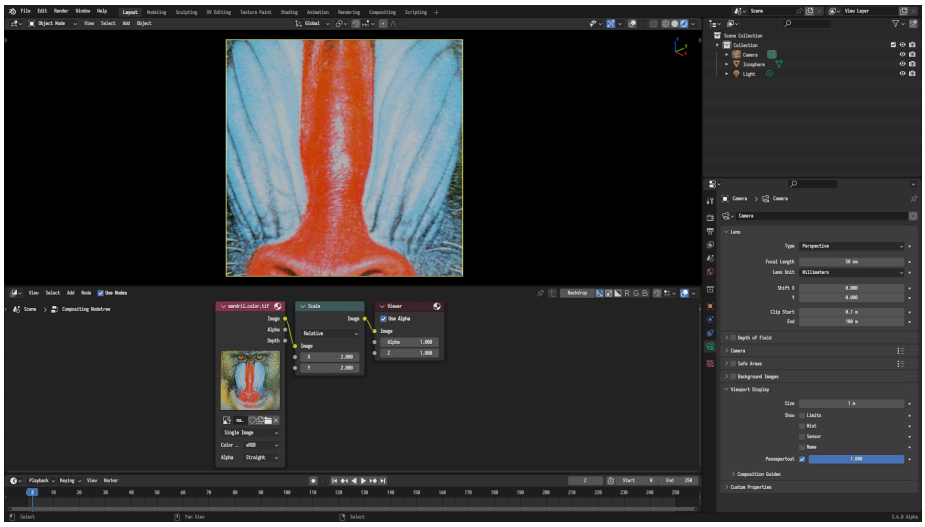

Figure 3: Realtime compositor with simple scale.

3 Simple Scale Undo

This example demonstrates the undo of a transformation, like the previously demonstrated simple scale.

As can be seen, the Tile-based and Realtime compositors produce similar results, where the image is intact after the undo, that's because both don't realize the transformation at the point of evaluation of the transform nodes, and thus their transformations are non-destructive.

The Full-frame compositor on the other hand produces an image of size 384×384 where $384 = \frac{768}{2}$, where the out-of-bound areas weren't restored after the undo, that's because it realizes its transformations at the point of evaluation of the transform nodes, and thus its transformation is destructive.

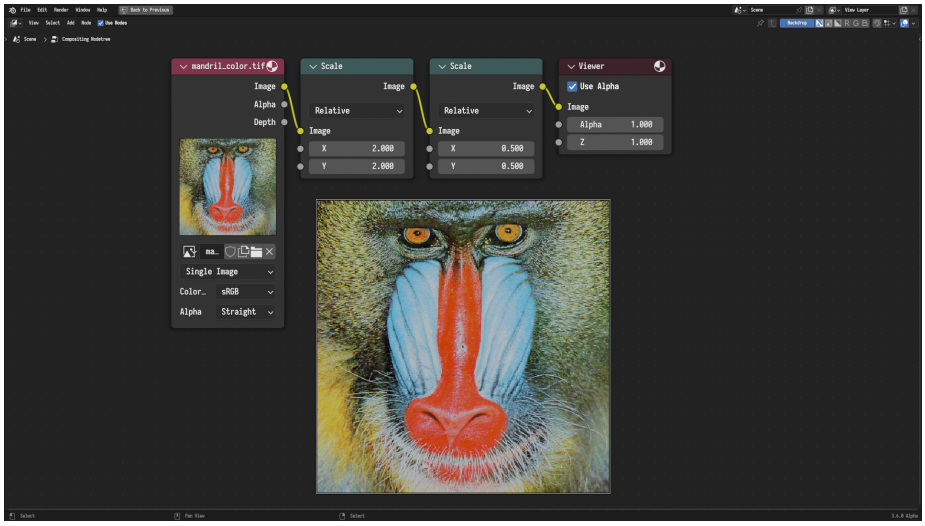

Figure 4: Tile-based compositor with simple scale undo.

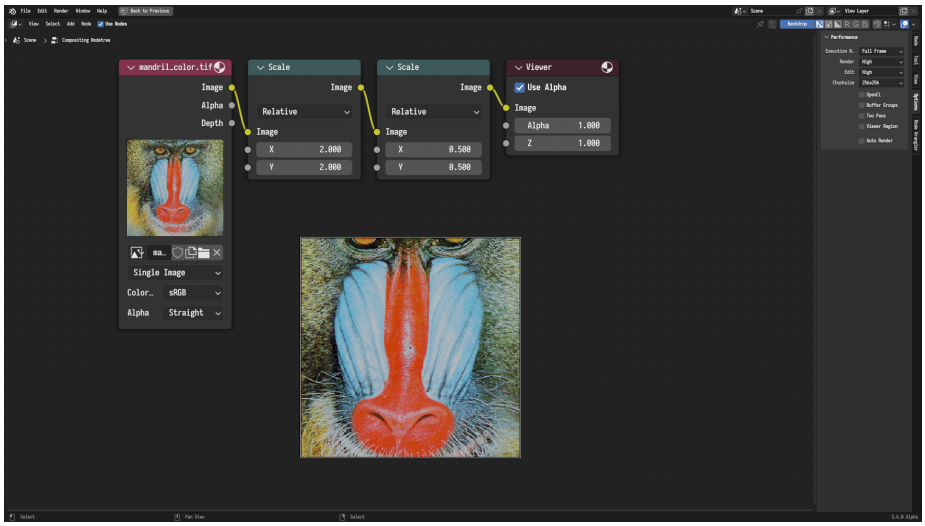

Figure 5: Full frame compositor with simple scale undo.

Figure 6: Realtime compositor with simple scale undo.

4 Image Sampling

This example demonstrates image sampling where the previously demonstrated scaled image is sampled in the entire normalized $[0, 1]$ range at a certain size.

As can be seen, the Tile-based and Full-frame compositors produces cropped results as if they were not sampled.

The Realtime compositor on the other hand produces the full non-cropped image, that's because it assumes that sampling happens in the local space of the image, so any transformations are ignored.

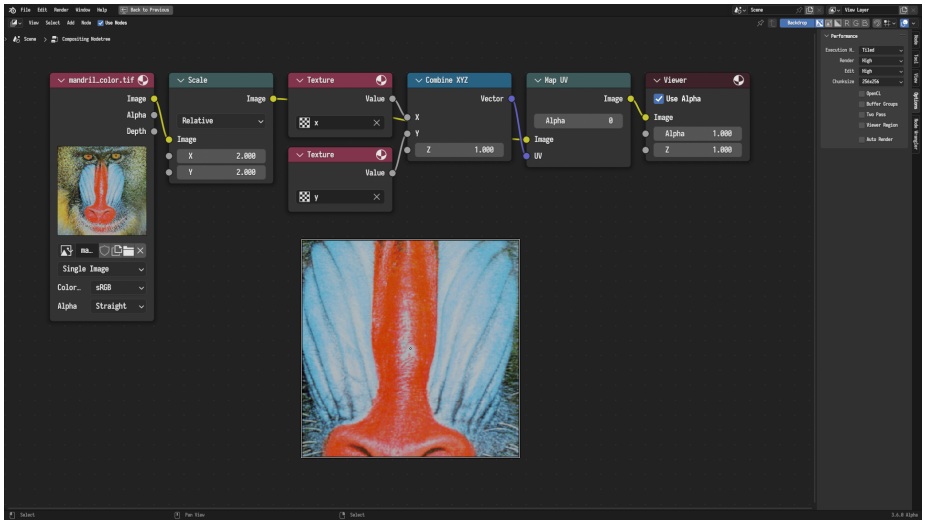

Figure 7: Tile-based compositor with image sampling.

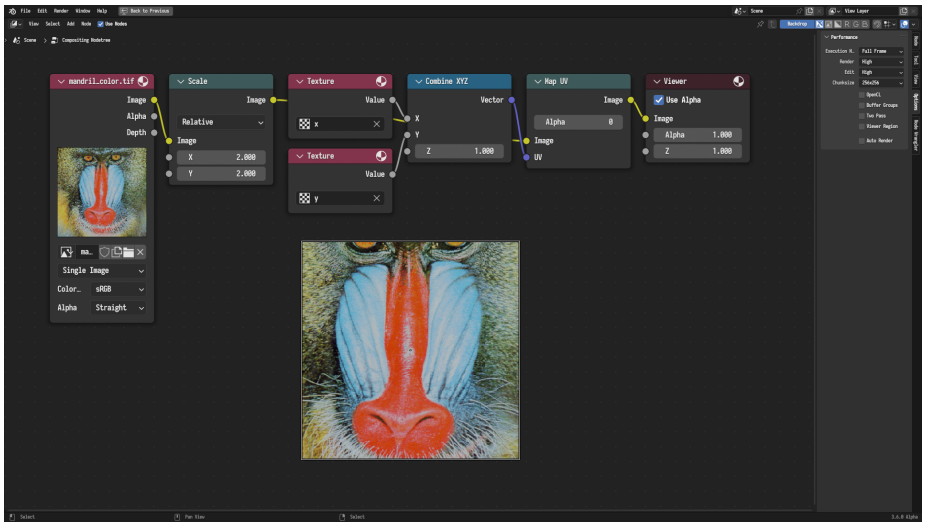

Figure 8: Full frame compositor with image sampling.

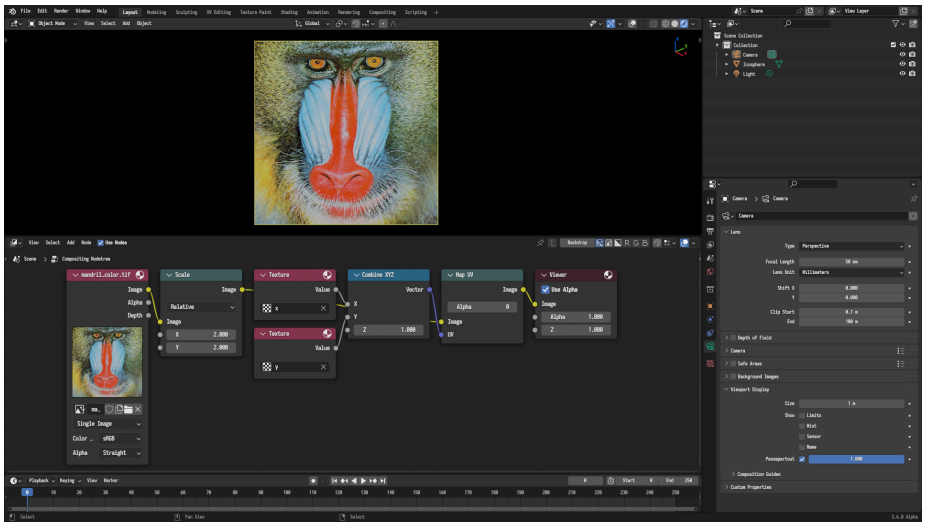

Figure 9: Realtime compositor with image sampling.

5 Automatic Realization

This example demonstrates the flipping of a rotated image along the X axis.

As can be seen, the Tile-based compositor automatically realized the rotated image on a new rectangular region of an inferred size, producing a new image, which is then flipped. That's because it automatically realizes the transformation of images upon meeting a buffered operation.

The Full-frame compositor produced the same result, the only difference is that it realized the image at the point of the transformation, not at the Flip node.

The Realtime compositor on the other hand flipped the image in its local space, that's because it doesn't do any automatic realization itself, and the user would have to explicitly tell it to do that if the intention was to flip in the global space.

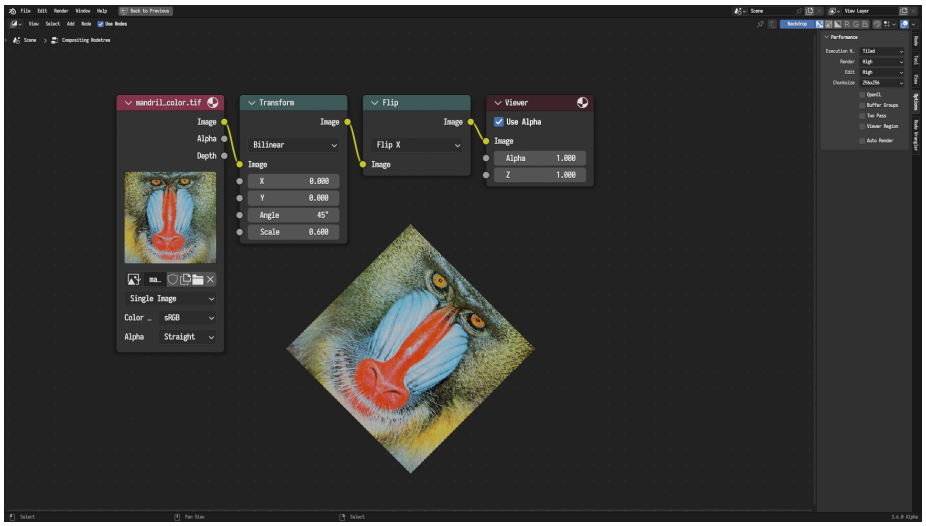

Figure 10: Tile-based compositor with automatic realization.

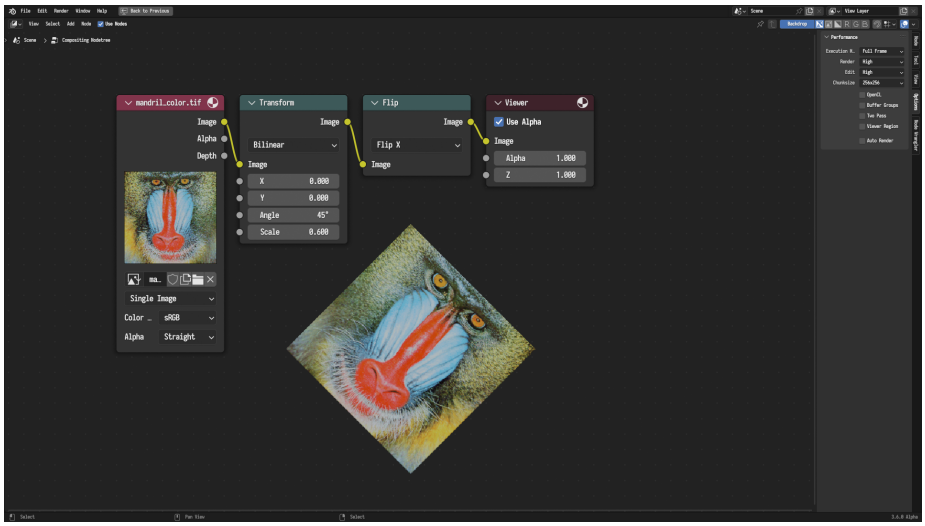

Figure 11: Full frame compositor with automatic realization.

Figure 12: Realtime compositor with automatic realization.

6 Size Inference

This example demonstrates size inference where a Translate node is given a single color, which is then blurred then mixed with an image.

As can be seen, the Tile-based compositor assumes that its input is not a single value, but rather a full image with a uniform color, having a size of 512×512, which is inferred from the size of the "upstream" Mix node, which in turn, assumes the size of the input image. In effect, since the translate node has a Y translation of $\frac{512}{2}$, the blur node realizes an image that is half transparent and half colored, which when blurred creates a gradient.

The Full-frame compositor similarly infers the same size, but it does not realize translations, so the half-and-half image is left non-blurred.

The Realtime compositor on the other hand evaluates the node tree from inputs to outputs, and does not attempt to infer anything from upstream nodes, in effect, it considers the input to the Translate node a single value, which is passed through untouched in the Translate and Blur nodes, so it is as if the Mix node has received a standard single value.

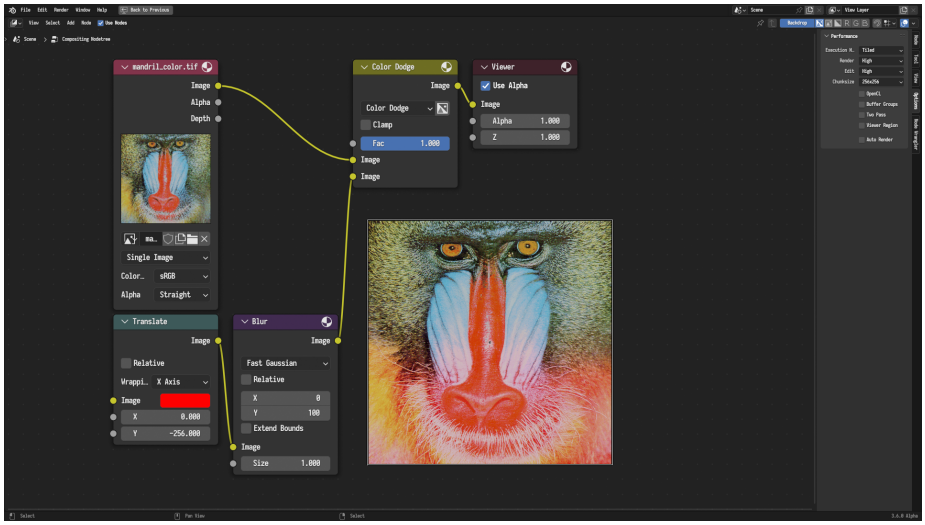

Figure 13: Tile-based compositor with size inference.

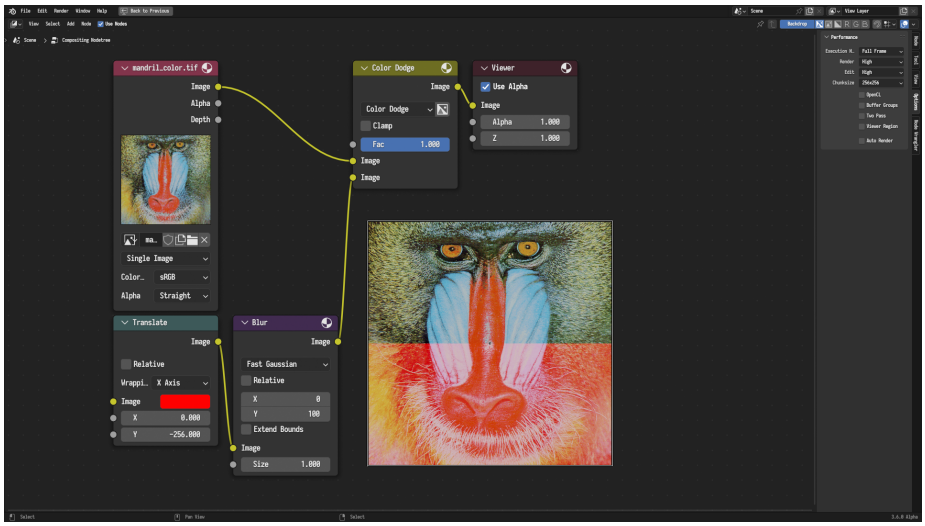

Figure 14: Full frame compositor with size inference.

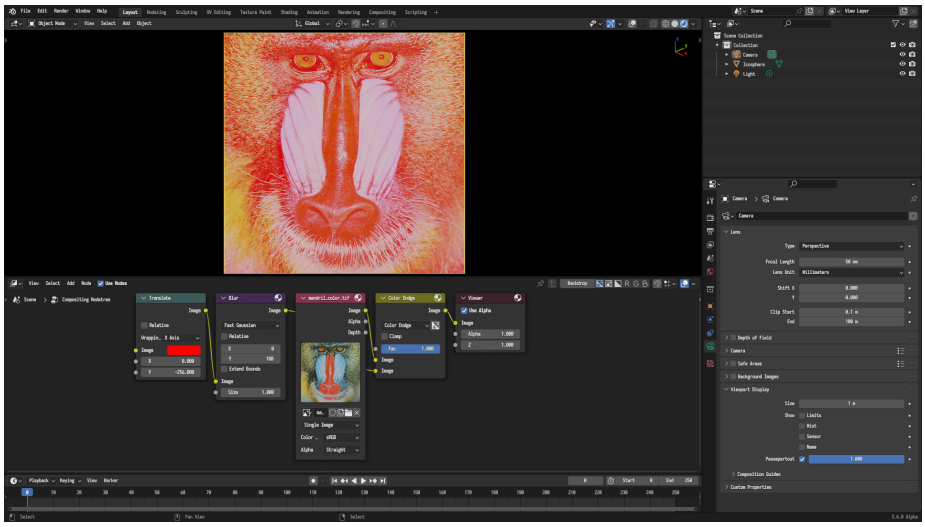

Figure 15: Realtime compositor with size inference.

7 Wrapping

This example demonstrates wrapping where an image is repeated uniformly to cover another image.

As can be seen, the Tile-based compositor repeated the badge image to cover the entire background image, however, it was clipped due to realization on the size inferred from the background image.

The Full-frame compositor doesn't repeat the badge image because it is immediately realized, in fact, the wrapping option in that case just switches between realizing or not realizing immediately.

The Realtime compositor on the other hand just repeats the image on the entire background, because it stores the preferred wrapping method in images until they are realized later.

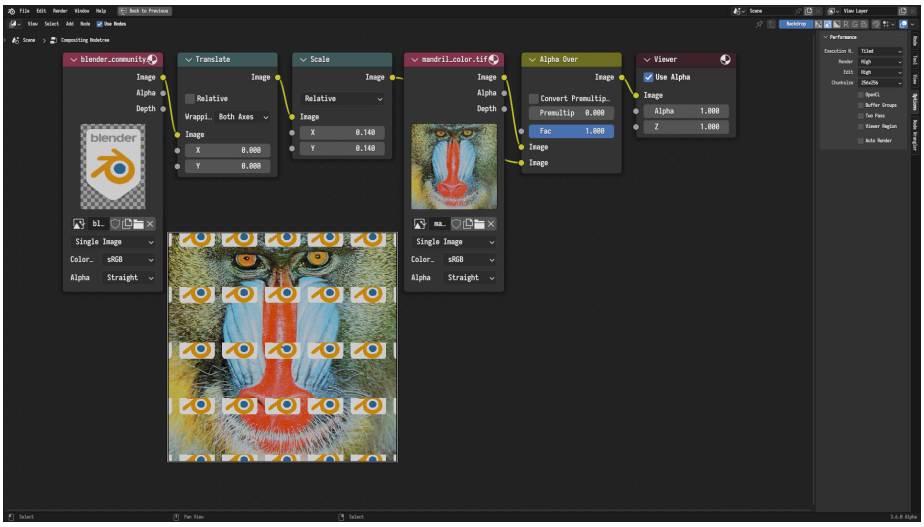

Figure 16: Tile-based compositor with wrapping.

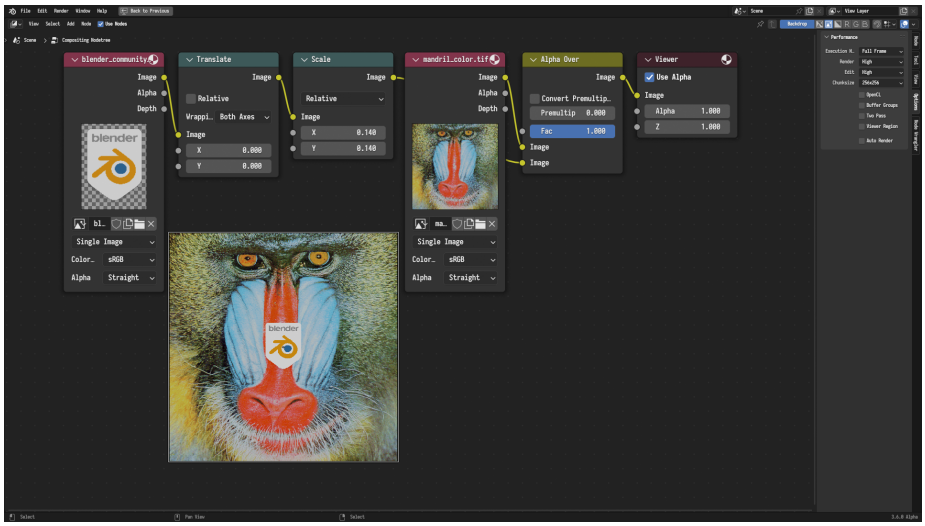

Figure 17: Full frame compositor with wrapping.

Figure 18: Realtime compositor with wrapping.

8 Pixel Density

This example demonstrates difference in pixel density where an image is overlaid on another scaled image.

As can be seen, in the Tile-based and Full-frame compositors, the badge image has a pixel density that matches that of the output, that's because the scaled background image was realized on a rectangular region that matches the pixel density of the output.

For the Realtime compositor on the other hand, the badge image has a pixel density that is half of the pixel density of the output, that's because it doesn't do any automatic realization, and the background input covers double the area while maintaining the same size.

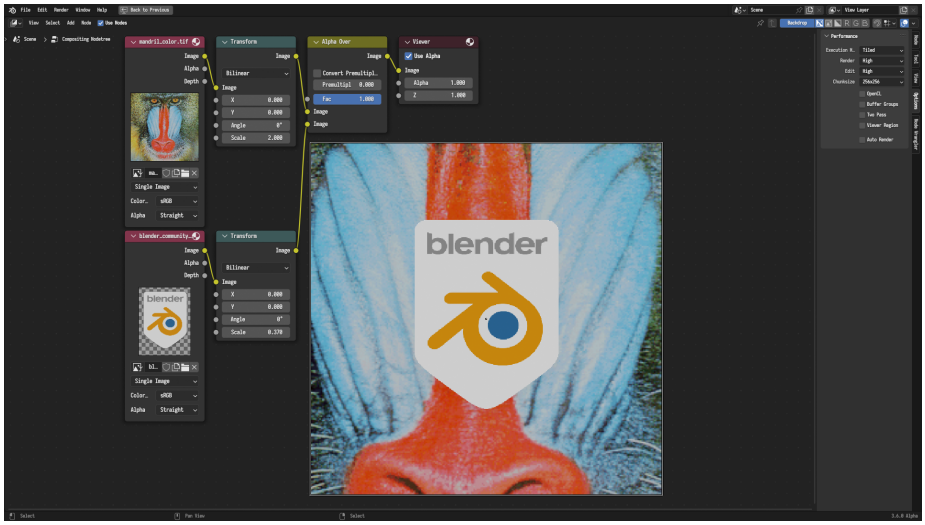

Figure 19: Tile-based compositor with pixel density.

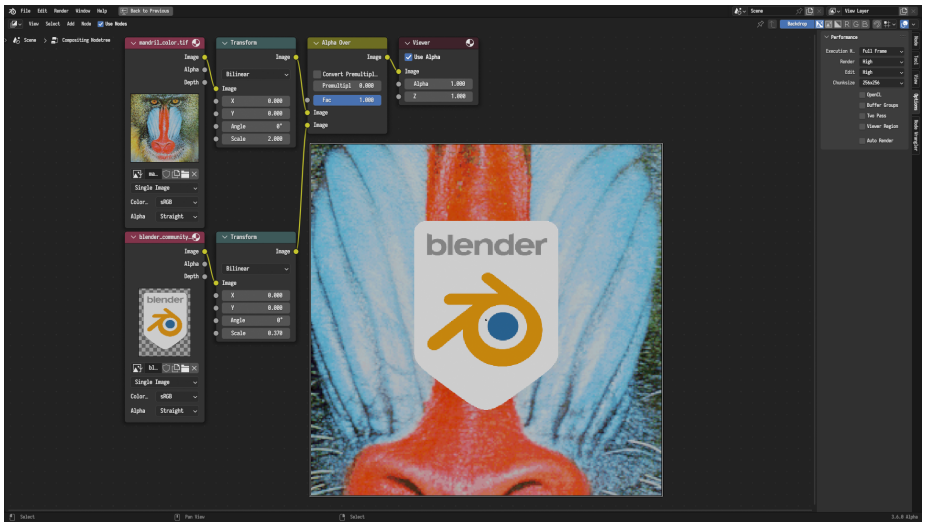

Figure 20: Full frame compositor with pixel density.

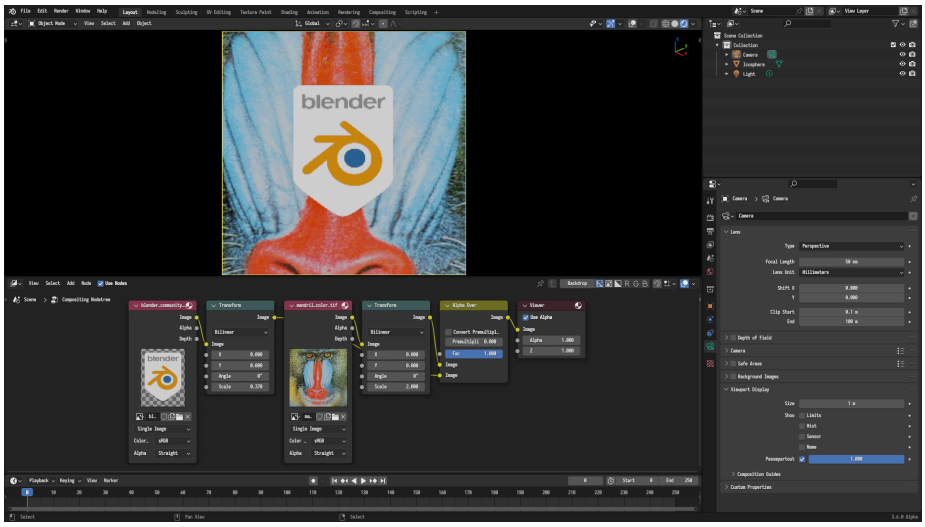

Figure 21: Realtime compositor with pixel density.