

This PDF is available for free; feel free to share it!

[Subscribe to my YT channel](#) & turn on notifications to get updates on new videos!

Hello everyone!

In this tutorial, I will show you how to use the **Pixelate Compositing node** in Blender to give your renderings a pixelated look, similar to the **retro charm of old video games**.

Although this tutorial was created with Blender 3.3, the **Pixelate node** has been available since at least version 2.79, making this tutorial valid for intermediate versions as well. The rendering engine used for the examples shown here is Cycles.

This tutorial applies to **both entire scenes and single objects**.

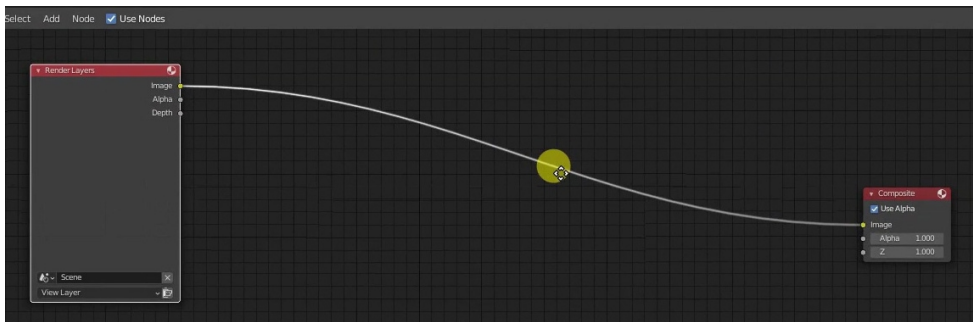
In the case of single objects, you can even create icons or sprites (sequences of images for animations) on a transparent background.

Regardless, the operations required are the same and occur during the Nodes Compositing process in Post-Production, which takes place after the rendering.

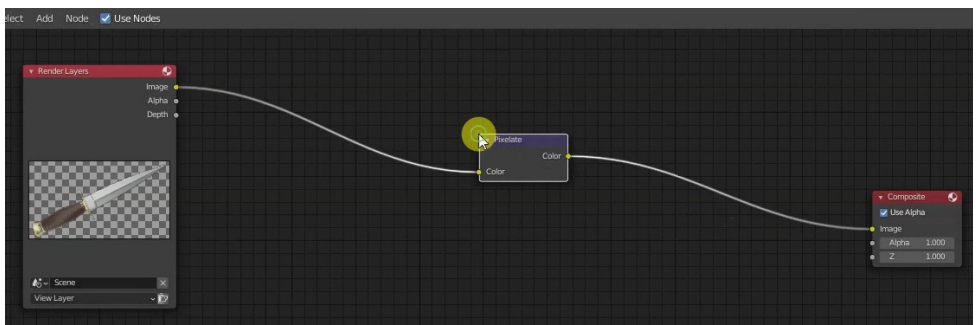


The scene doesn't have any standard light sources, but it does include **an HDRI image in the World background**. With everything ready for rendering, we can proceed to create the pixelated look.

After rendering, open the *Nodes* editor from the **General - Compositor** section and click on the "*Use Nodes*" checkbox to activate Post-Production. You'll see the *Render Layers* output linked to the *Composite* input.



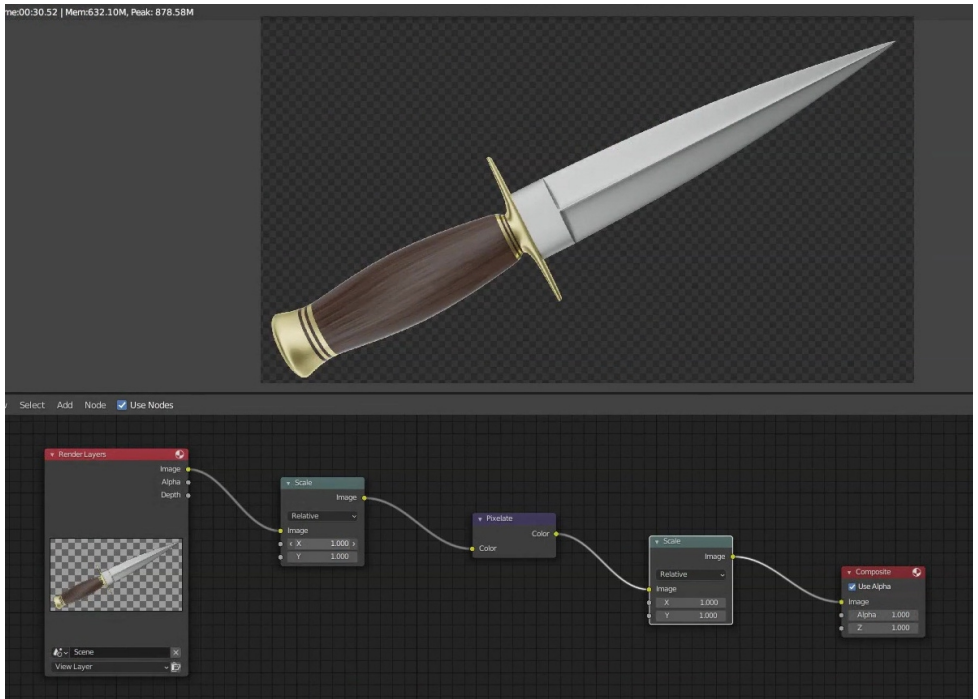
Add the **Pixelate** node by choosing it from *Add - Filter* and placing it between the two existing nodes. In the **Image Editor** window displaying the rendering, you won't notice any difference.



According to Blender's online documentation, you need to place *Pixelate* between two ***Distort - Scale*** nodes, with the first node having a value **less than 1** and the second node being **the reciprocal value** of the first one.

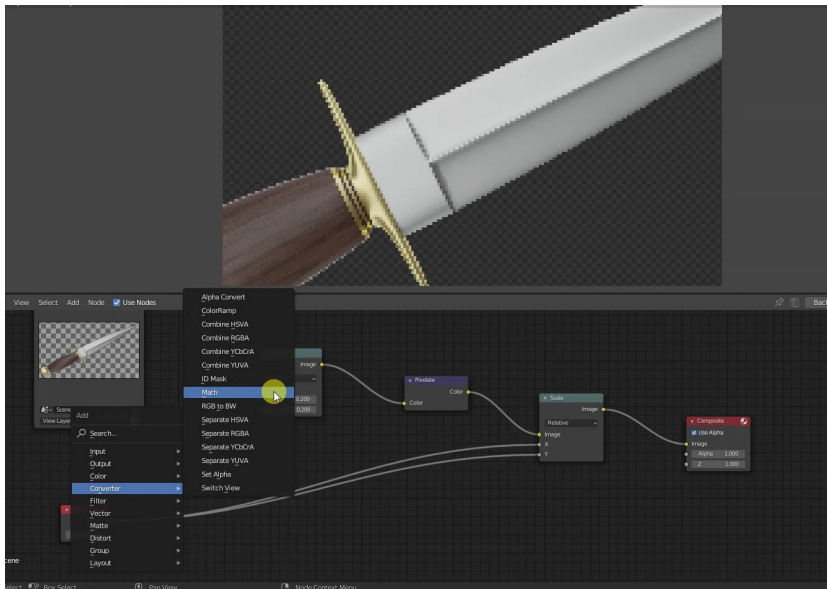
As an example, the documentation uses **0.2 and 5**.

Using these two values produces an interesting result, but feel free to experiment with different values to increase or decrease the pixel size and modify the result in real-time.



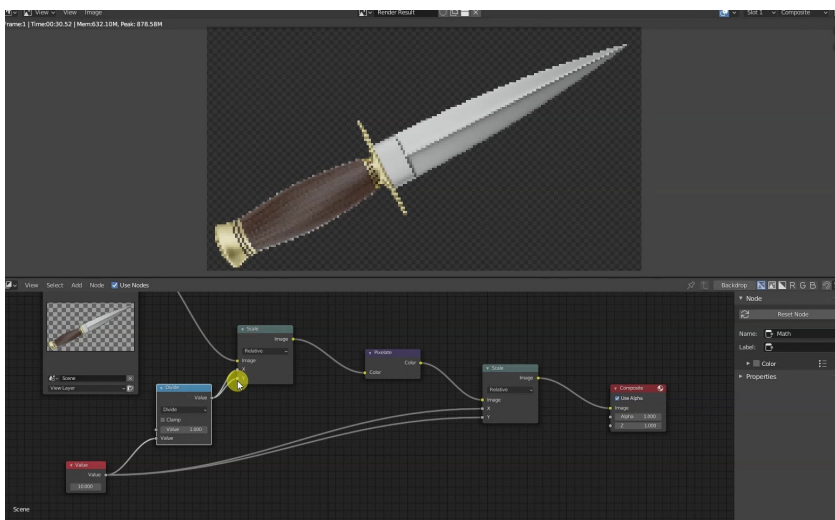
To interactively modify the value of the pixel size, insert an ***Input - Value*** node in the scheme. This node provides a **numeric value**, which we can use in the second ***Scale*** node.

To use ***Value*** in the first ***Scale*** node too, first, calculate its **reciprocal** by inserting a ***Converter - Math*** node.

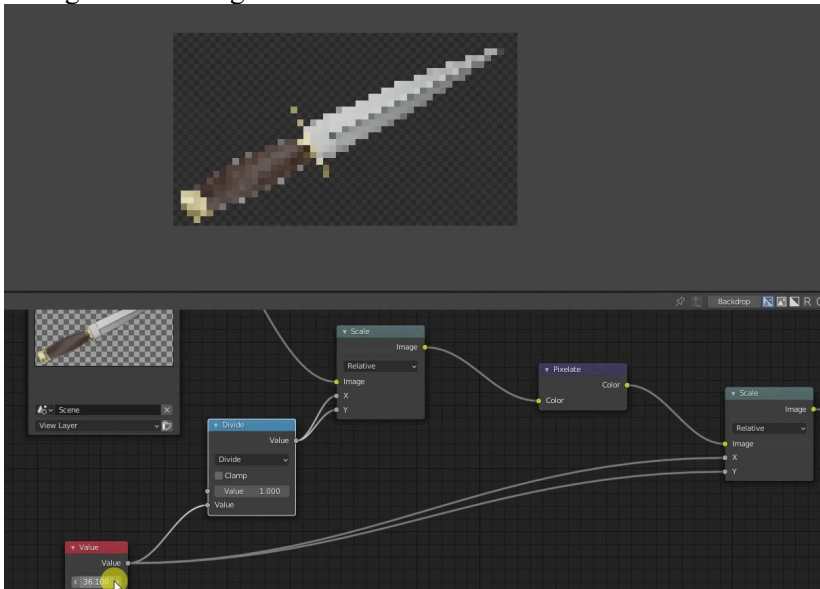


Set **Divide** as the **Math** mode, type 1 in the first **Value** field, and connect the **Value** node to the second **Value** field of Math.

Finally, connect the output of **Math** to the **X and Y** inputs of the first **Scale** node.



Now you can **modify the value of the *Value* node** and observe how the rendered image changes in the Image Editor window in real-time.



Before closing this tutorial, I want to give you a suggestion: instead of using a PBR Material with realistic Texture, **try using materials with a *TOON Shader*** for flatter, less nuanced colors. Also, perform multiple tests with **traditional light sources** (avoiding global lighting) to obtain more interesting shaded areas.

I'm showing an example of the same object as before, with the changes I just described to the Materials and lighting of the scene.



That's all for this tutorial. See you soon!