Nuke Tutorial

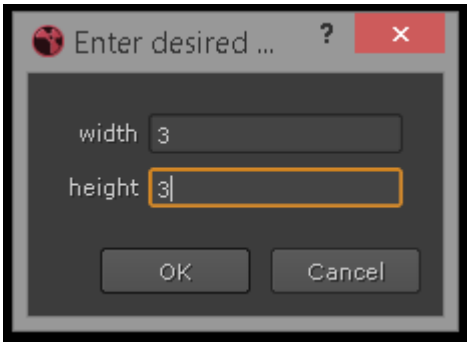
Understanding the matrix node

for Beginners

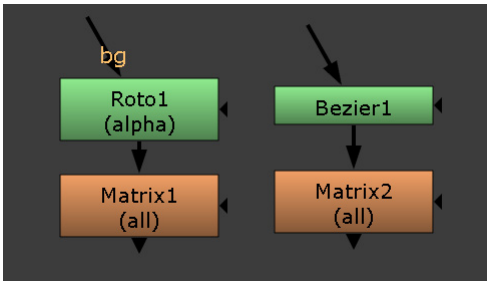
Matthias Eckhardt

This Tutorial is aimed at Nuke Newcomers that try to get a basic understanding of how the matrix node works. Some of the Values are simplified to make it easier to follow. If you have any questions feel free to aks at:

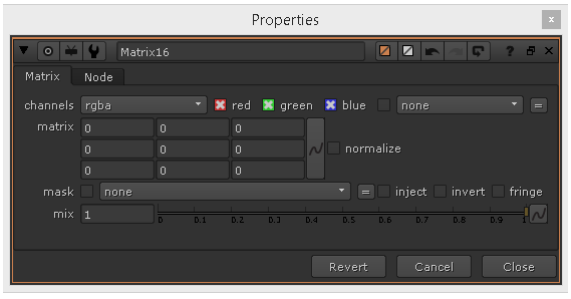
matt.eckhardt@hotmail.com
http://www.poly-by-poly.com



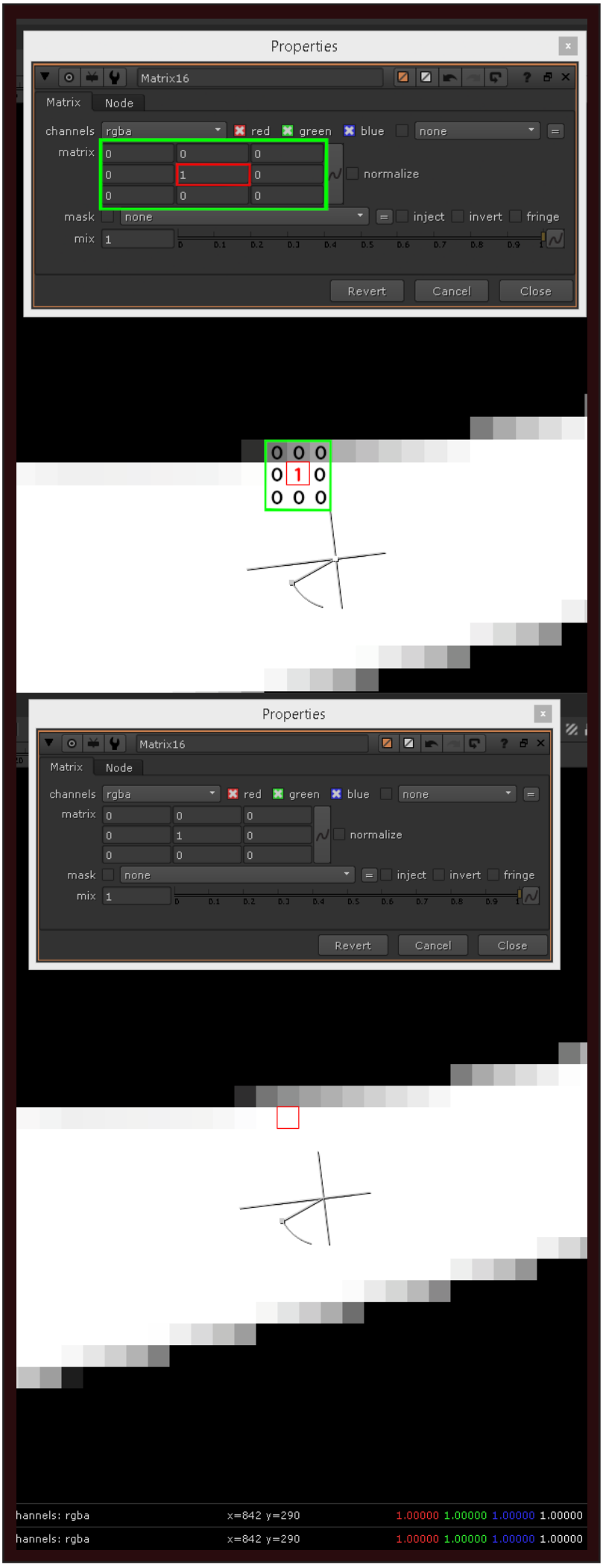
The matrix node is a representation of a pixel (the center value) and his surrounding neighbour pixels (the values around the center). By changing the values you start a multiplication process for each number which in the end changes the value of the center pixel. This process is repeated for each pixel of the image.



After the creation of a random shape with a roto or bezier node, the input for the matrix is ready. You can use other inputs like Read Nodes, Checkersboards, noise,... When you create a new matrix node, a popup appears where you can type in the desired size of the matrix. To keep it simple a 3x3 matrix is used in this tutorial.



A matrix is a very versatile mathematical method, and is used in many other applications, even though the way it works is different, This Tutorial focuses entirely on Nuke. If you dont want to do/follow the math, you can focus on the pictures as they show the outputs. It is highly recommended to try to follow the calculations though so a real understanding of the node is built.

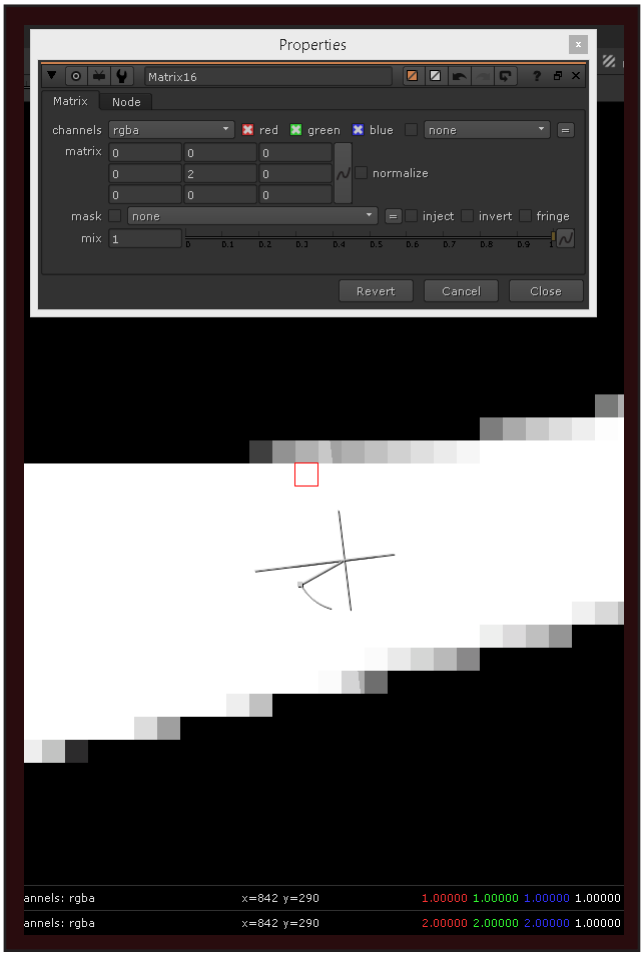


As you can see in the picture, we have a random bezier shape and a matrix node. The red marked pixel is represented by the center of the (also red marked) matrix. The upper left value in the matrix represents the upper left pixel in the green area, the value above the center represents the pixel above the red pixel and so on.

The matrix node works through every single pixel of the image in the above described way. As we can't calculate all of them, we will focus on this specific pixel.

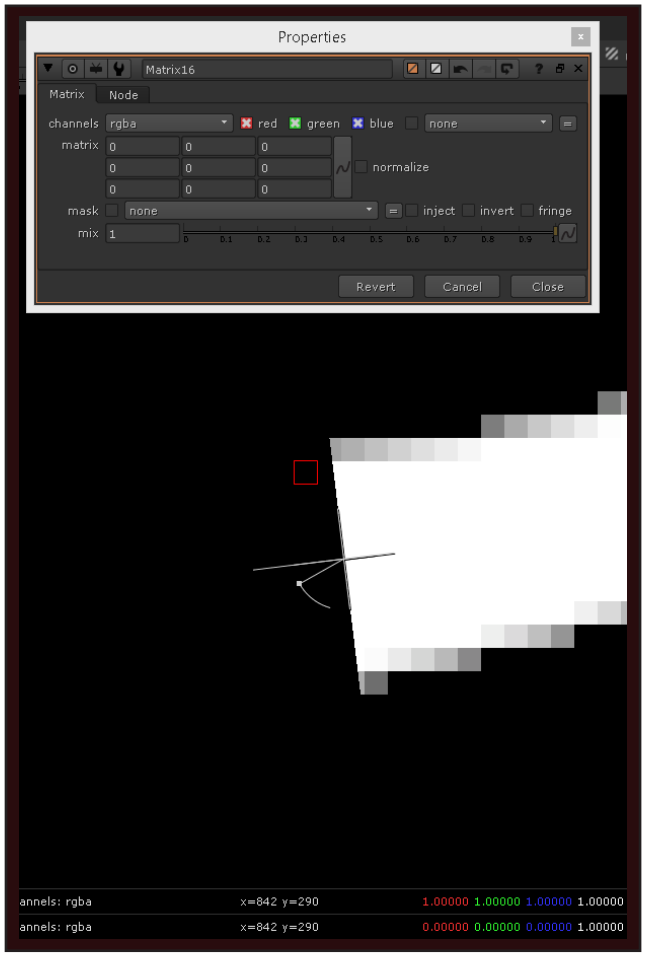
In this example, the red marked pixel from the input has a rgb value of 1 / 1 / 1. We will use this to further explain how the node works. To make it easier to track the process, the colorvalues will be marked pink. As you can see only the center input of the node has a value of 1, so the rgb pixel values will be multiplied by 1 each:
1*1 / 1*1 / 1*1.

Since the surrounding values in the matrix node are 0, the surrounding pixels rgb values will be multiplied with 0, so they all result in 0 and have no effect on the center value.



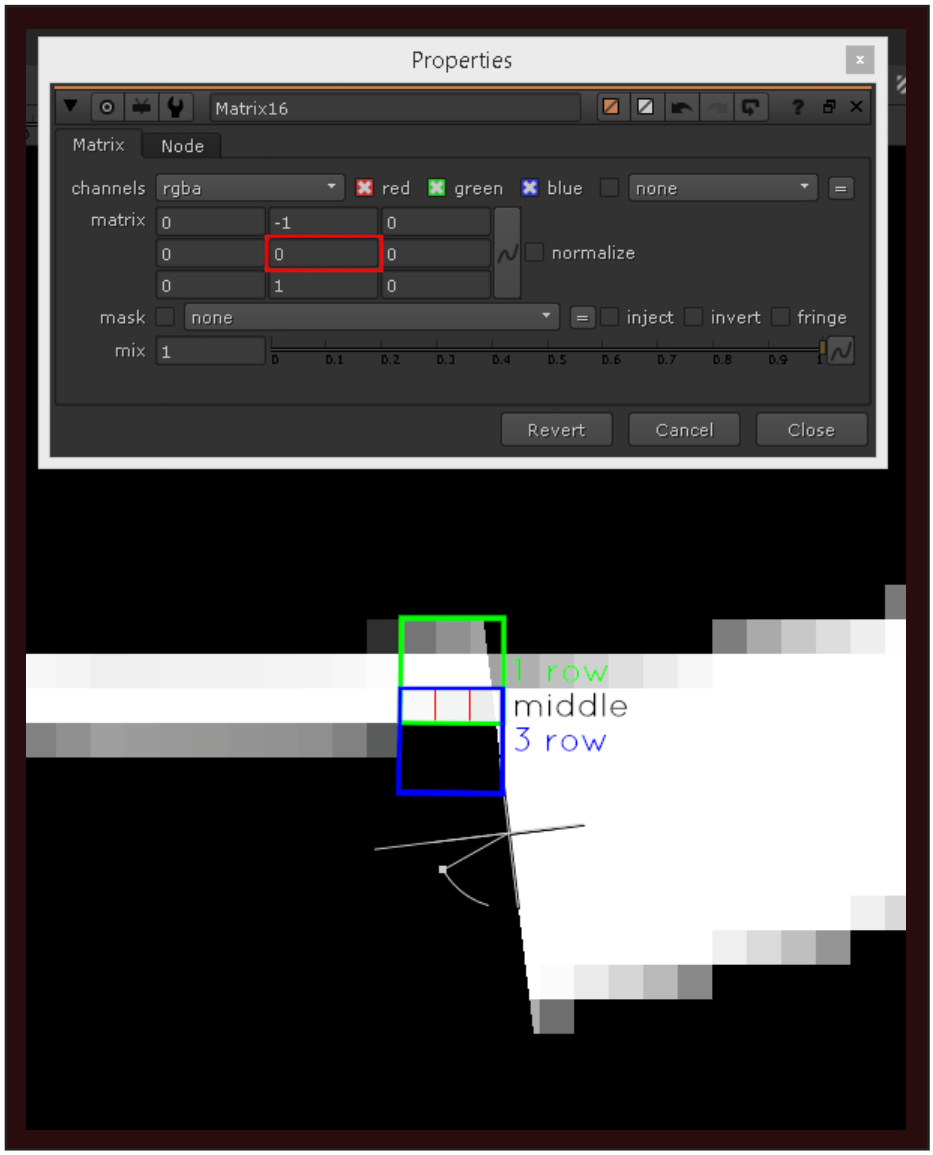
As we changed the center value in the node to 2, the multiplication changes the rgb values to 1*2 / 1*2 / 1*2. You can also see this on the bottom with the rgb values from both inputs of the wipe (w). As the footage is white, the change is not quite visible, only in the half transparent parts you can see a bit of change on the edge of the wipe and on the half transparent pixels.

A wipe (w) is a possibility to overlay two viewer-inputs to compare them without having to merge the footage. It also shows the input values on the bottom of both of the viewerinputs on the marked pixel. In this example you can see the value of 1 on the original output and the value of 2 on the matrix output.



In this picture we changed the center value to 0 and now you can clearly see the left side of the wipe is black. Each pixel gets multiplied by 0, which means you now get rgb values of 0 / 0 / 0. This process also works with negative numbers.

Until now we only worked with the center pixel value which multiplies with the rgb-inputs. As the surrounding values of the matrix always were 0 they had no impact.



As you can see, we have a completely different picture now. Here we included the surrounding pixels above and underneath our center. In our example we have the following node process:

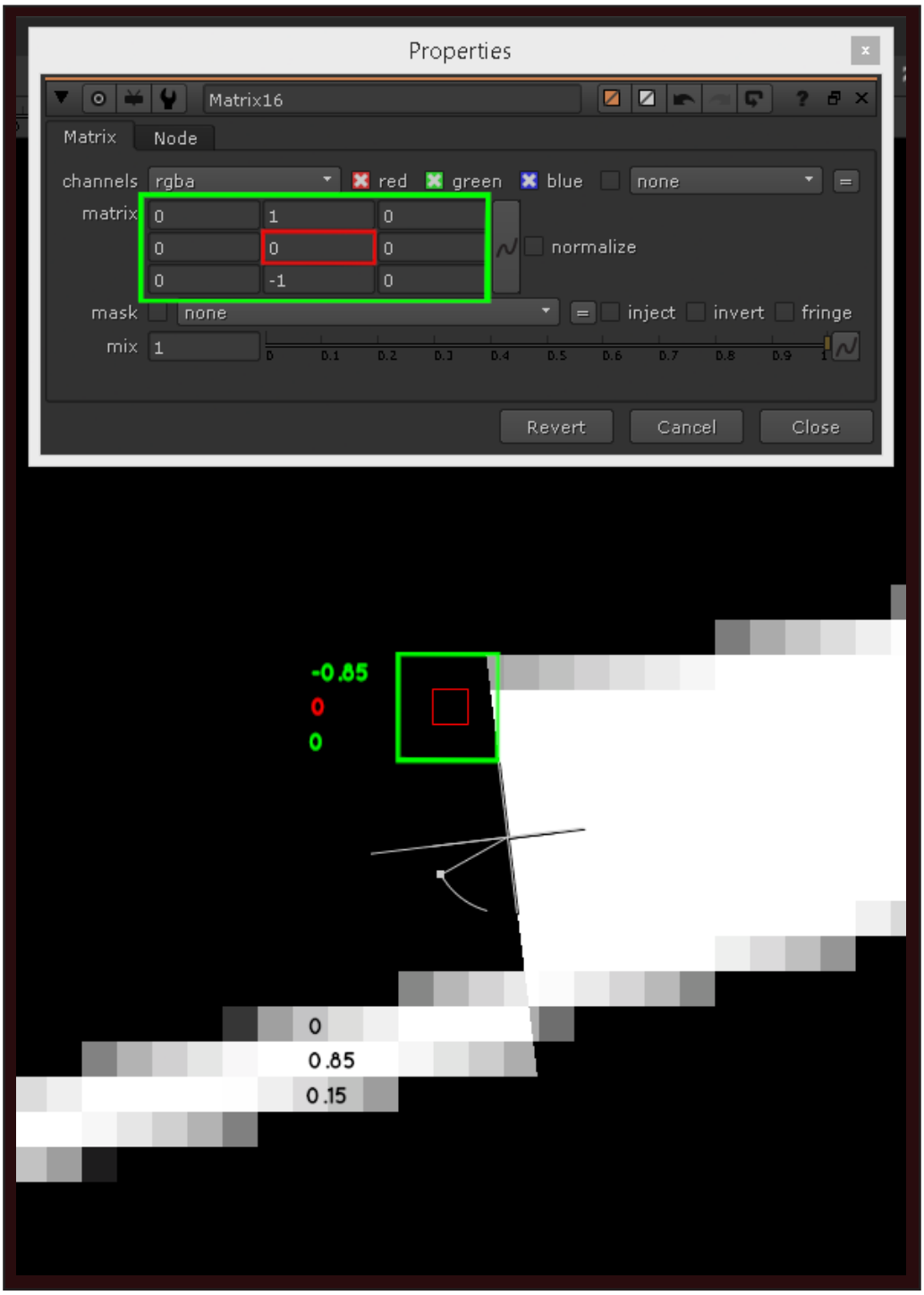
The pixel above our red marked one has the original (simplified) value of 0.15 / 0.15 / 0.15, the center pixel has 1 / 1 / 1 and the pixel below has also a value of 1 / 1 / 1 too. This means in our overall calculation we have (0.15 * -1) + (1 * 0) + (1 * 1). With that we have a new value for our middle pixel: 0.85 / 0.85 / 0.85. If you now wonder why there is only the line on the upper side in our output, think about what is happening to the other pixels. The pixel above our marked one is the last input pixel with color information before the shape end is reached and the footage becomes black. When it goes goes through the matrix the 1st row looks like:

(0 * -1) + (0.15 * 0) + (1 * 1) = 1
(black footagepart * -1) + (edge of mask * 0) + (white footagepart * 1) = 1

The pixel underneath our marked one follows the same principle and our 3rd row pixel has this calculation:

(1 * -1) + (1 * 0) + (1 * 1) = 0

Since all the pixels below that have the same calculation, they all stay at 0. If we visualise that, we get that 3 pixel top-edge like in the picture.

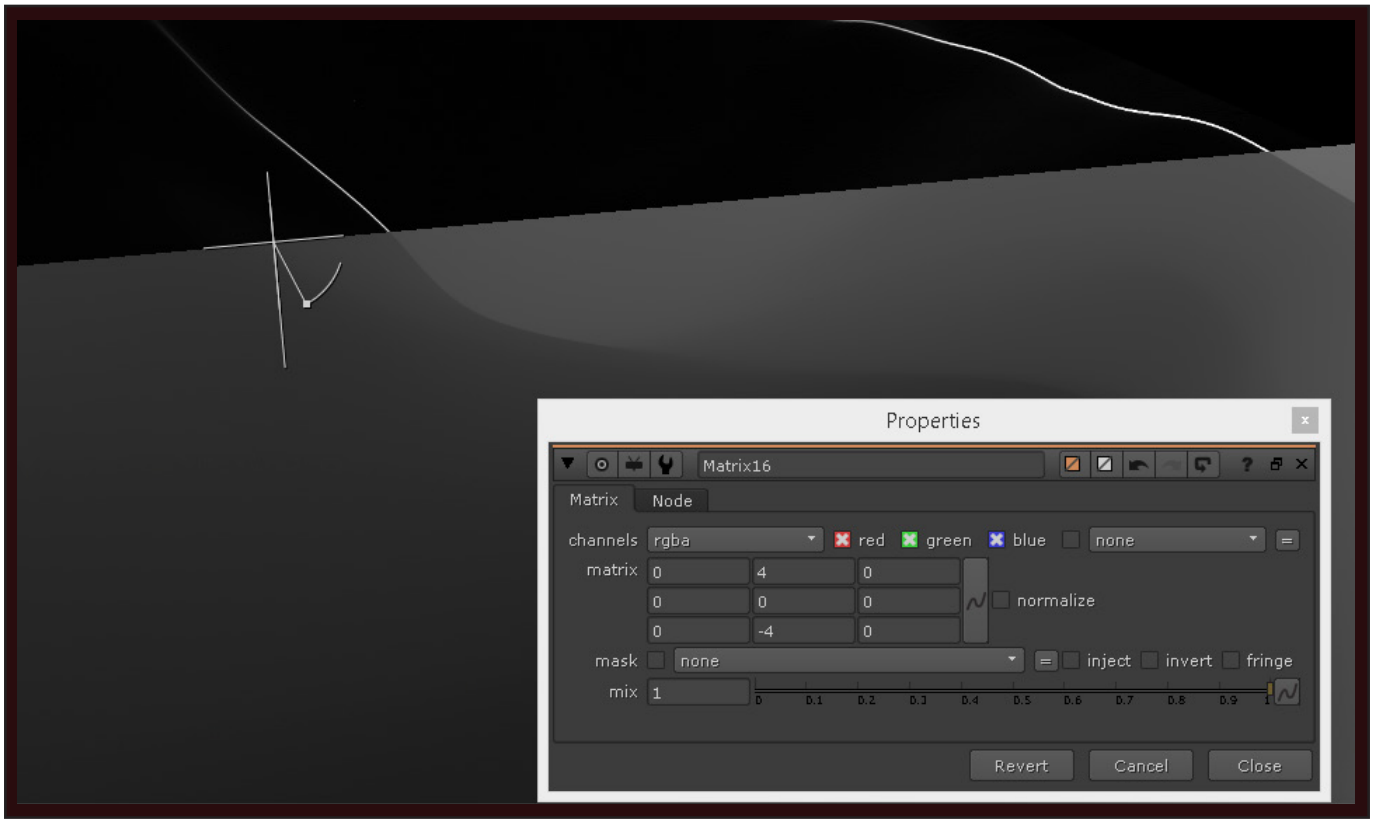


Here we switched the negative and positive number around to get the outline on the bottom instead of the top. This is simply the same calculation, only reversed. If we look at the three marked pixels and compare them to the calculations before, we get the following results:

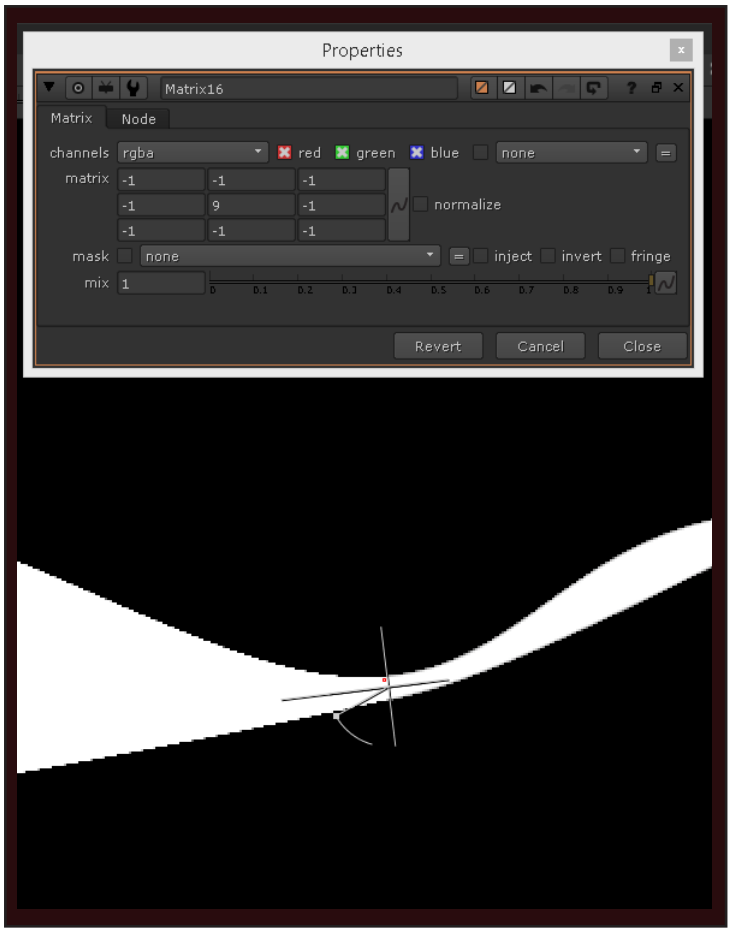
(0.15 * 1) + (1 * 0) + (1 * -1) = -0.85
(1 * 1) + (1 * 0) + (1 * -1) = 0
(1 * 1) + (1 * 0) + (1 * -1) = 0

With these three calculations we can already see why the line is created in the bottom of the shape. For a better understanding the results of the top and the results of the bottom are visualised in the picture above.

(again: these numbers are simplified for a better understanding)



As we only used a very high contrast black/white input before, you can also see that its coming in quite handy with other footage. Here it is used to extract edges from a depth-pass. You dont need to only do this with black and white images, but you always got to consider that the values in the matrix node are multipliers.



You can use the matrix node for many different things like the edgedetects we did the first few pages, blurs, emboss, sharpen (like in the picture on the left), ... You can achieve quite fancy effects but always be careful as the values often exceed 1 and go beyond -1. You may want to consider to clamp them. Have fun.