

# MattelID in Maya



This workflow is deprecated.

Have you tried [Cryptomatte](#) instead of MattelIDs for automatic ID generation?

Assigning [Matte IDs](#) to rendered object and outputting them as part of the AOVs can be useful for later compositing because this allows compositors to isolate objects.

In the most basic way, you can do this in a few steps. We describe this below for material IDs but the process is similar for shapes.

Below we have a simple example with three shapes. This can be used for MattelID0 since a color output has three easy possibilities -- Red, Green, and Blue. (We will cover using patterns after this.)

? Unknown Attachment

First we'll assign a Green MattelID to the sphere. We can do this by selecting the transform and adding the RenderMan user attribute for MattelID0:

? Unknown Attachment

If you were to render at this point, even if you added the correct render pass, it would not work. You need to connect a PxrMattelID to the material. You can do this using the Node Editor and connecting to the Utility Pattern Connection of the PxrSurface material:

? Unknown Attachment

Now add the appropriate pass to the render outputs:

? Unknown Attachment

The resulting MattelID0 pass looks like the one below:

? Unknown Attachment

You can repeat this process for the other materials, adding one Blue and a Red. You can connect the same PxrMattelID to the other PxrSurface Utility Pattern connections as shown below:

? Unknown Attachment

? Unknown Attachment

This works fine, but what if you needed to pass through a pattern as your result?

If you recall, we didn't make any changes to the PxrMattelID node. It was left at default. We will leave this one alone but disconnect it from PxrSurface1 assigned to the sphere. We're doing this so we can affect just one object and not all three.

Create a new PxrMattelID and connect it to the PxrSurface1 for the sphere. In the MattelID0 connection connect a PxrChecker and a manifold as shown below and then render.

? Unknown Attachment

? Unknown Attachment

Ok, that sort-of worked, but it's green.

This is because in the material we still have the color set to green in the PxrSurfaceMaterial.

This color is multiplied against the pattern we connect to the PxrMattelID. This can be useful, but maybe we want a black and white checker this time. Set the color to white (since it's being multiplied, this should pass the pattern through unchanged.)

? Unknown Attachment

Now you can add additional PxrMaterialDs and chain together more patterns for different results in your renders. Your compositor will thank you.