Using PxrMattelD

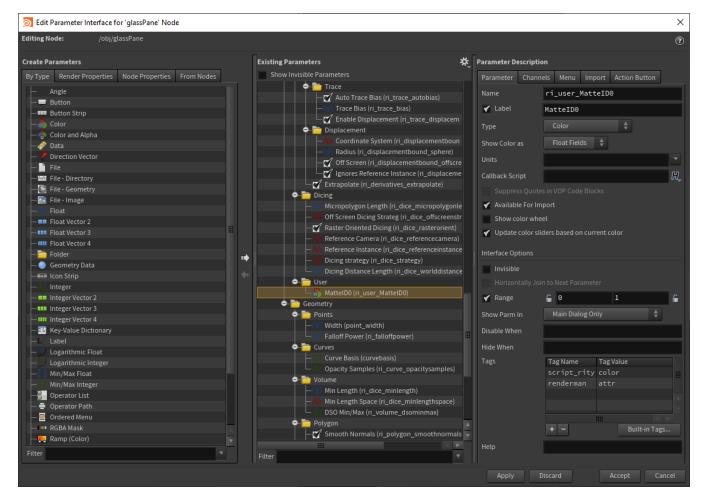
PxrMatteID outputs matte AOVs for compositing. This requires you add a User Attribute, explained in more detail here.

A PxrCryptomatte has since superseded this workflow as a useful and nearly automatic alternative.

Add a user attribute for MattelD0.

Duplicate the fields for:

- Name
- Label (this is the UI name)
- Tags (necessary to see RenderMan attributes and edit them during interactive rendering) script_ritype color renderman attr

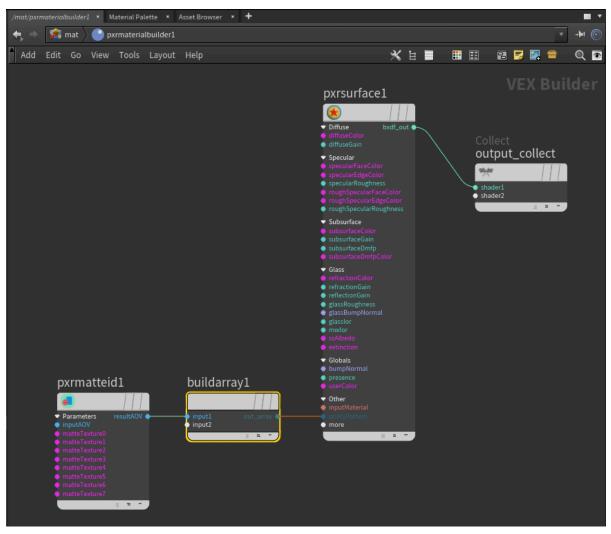


Set its type and color value.

sphere × Take List × Parameter	Spreadsheet × 🕂				
🚓 🌩 🔛 obj					* 🖄 💿
Geometry sphere				* H 🛈 ?	
Transform Material Render Misc					
User Defined Attribu 1		+ – Clear			
× + Attrib Name 1	MatteID0				
Туре	Color value 🛔				
String Value					
Real Value					I
Color Value			1	θ	
Vector Value					

Add PxrMattelD

Create a Houdini buildarray node to connect to the Utility Pattern on the PxrSurface material. Then connect a PxrMattelD Pattern node to the buildarray as shown below.



pxrmatteidI × Take List	× Parameter Spreadshe	et × +			
🚓 🔿 🤌 shop 🔪 💦 risnet1					- 🗄 💿
R Pxr Matte ID pxrmatteidl					₩, 0 0
Parameters					
Input AOV	0				
	🖌 Enable				
Matte Texture 0	0				1
Matte Texture 1	0		0	0	*
Matte Texture 2			0	0	*
Matte Texture 3			0	0	*
Matte Texture 4			0	0	*
Matte Texture 5	0		0	0	*
Matte Texture 6	0		0	0	*
Matte Texture 7	0		0	0	*

Connect PxrMatteID's resultAOV to PxrSurface's Utility Pattern.

Set up AOV for Output

Choose the correct/corresponding MatteID AOV output from the RenderMan ROP node Displays Tab

risl ×	Take List 🛛 × Parameter Spreadsheet × 🕂				Ŧ
\Leftarrow, \Rightarrow					
) Ŷ Rende		*, ∭, (?
Render					
Valid					
:					
Reno					
	Camera /obj/cam1		a	₹	
Options					
0 × +					
× +	Display \$HIP/render/\$HIPNAME.\$OS.\$F4.exr Create Intermediate Directories			ß	
	E Mattes				
	✓ MattelD0				
	MattelD1				
	MattelD2				
	MattelD3				
	MattelD4 MattelD5				
	MattelD6				
	MattelD7				
	🖼 Integrator				

You will notice we set a color on the OBJ with an attribute and there's also a color from the MattelD pattern itself. This is so you can multiply the color of the attribute by the pattern node. For example, you can supply a texture mask to the MattelD pattern node and have it multiplied against the color chosen in the MattelD Attribute