Josh Johnson

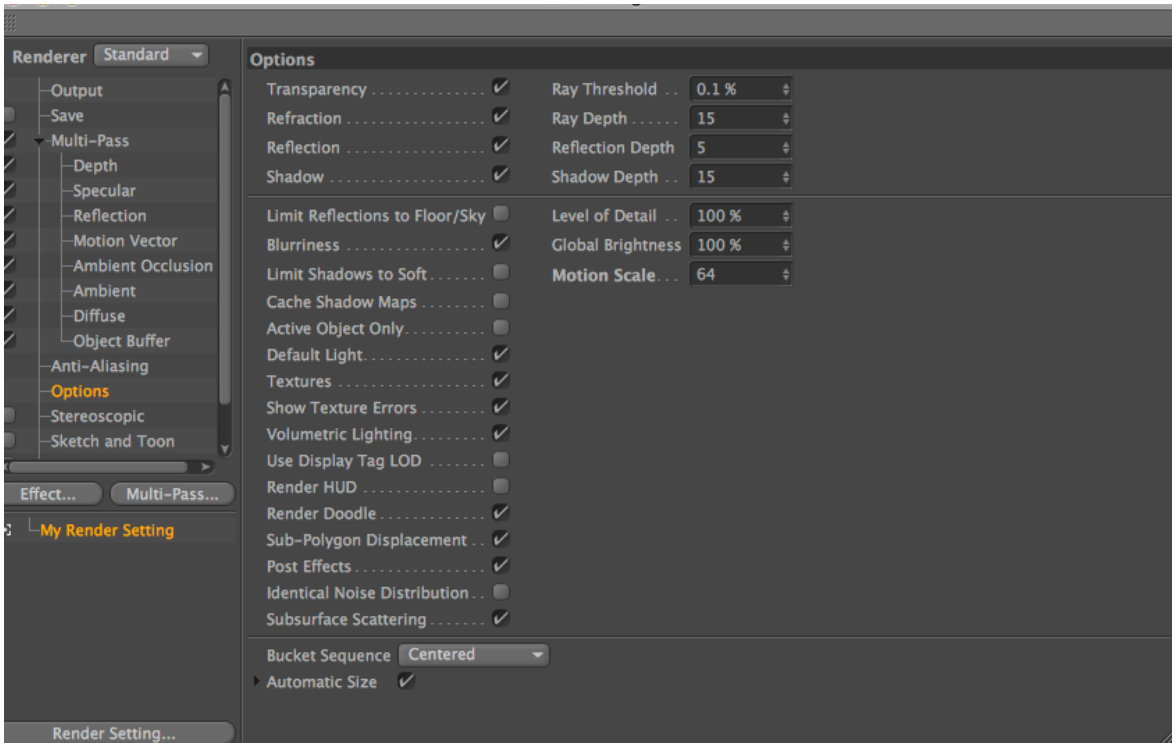
www.vfxdaily.com
josh@vfxdaily.com

C4D Motion Vectors in NUKE

So after struggling for a long time with trying to make a motion vector pass in Cinema 4D work in NUKE I think I found a workflow that seems to work. First off, I want to thank the blog: http://inpyo3d.blogspot.com/. It had info on Maya Motion Vectors to NUKE workflows and I tried to translate their methods to a C4D/NUKE workflow.

Cinema 4D

In C4D add a motion vector pass as you normally would and setup your project to render in 32bit. Use OpenEXR. Now go to your Render Options and take note of the Motion Scale. In this screenshot mine is set to 64.



Renderer Standard

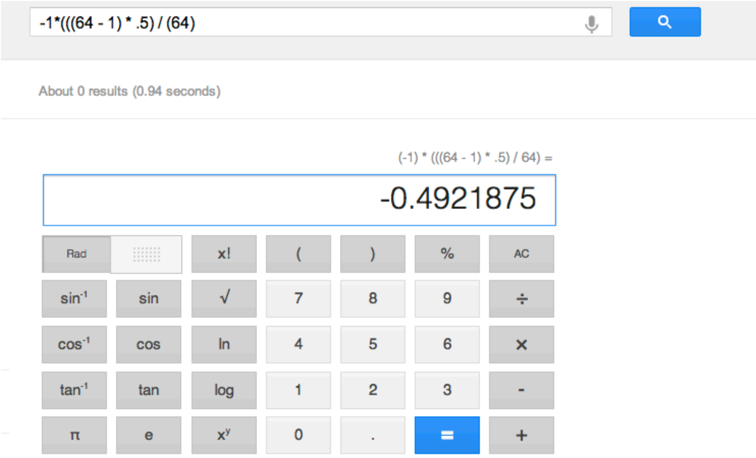
Options

Transparency ☑
Refraction ☑
Reflection ☑
Shadow ☑
Ray Threshold 0.1 %
Ray Depth 15
Reflection Depth 5
Shadow Depth 15
Limit Reflections to Floor/Sky □
Blurriness ☑
Limit Shadows to Soft □
Cache Shadow Maps □
Active Object Only □
Default Light ☑
Textures ☑
Show Texture Errors ☑
Volumetric Lighting ☑
Use Display Tag LOD □
Render HUD □
Render Doodle ☑
Sub-Polygon Displacement ☑
Post Effects ☑
Identical Noise Distribution □
Subsurface Scattering ☑
Level of Detail 100 %
Global Brightness 100 %
Motion Scale 64
Bucket Sequence Centered
Automatic Size ☑

NUKE

In NUKE setup your tree as you would then at the bottom of it pipe in a VectorBlur node. In the VectorBlur dialogue add your MV pass from C4D to the UV Channel. Here's where the tricky part comes into play. Well, the math part and math is hard for me... Thankfully some of the info I found online helped me with this part. So I found an equation online -1*(((Motionscale - 1) * .5) / (Motionscale)). This is the equation you must enter into the add "u" and add "v" fields of the Vectorblur node. In C4D we set are

Motion Scale (Maximum Vector Length) to 64 so the equation I entered into the u and v fields looked like this: -1*(((64 - 1) * .5) / (64)) which equals -0.496094. Now you can just enter the equation into the field and NUKE will solve it or you can enter it into Chrome or a calculator and enter the result into NUKE. Something I just wanted to point out the NUKE manual states if your MV's have been normalized to be between 0 and 1, you can set the u and v values in the add control to -.5. I found using -.5 instead of -1*(((64 - 1) * .5) / (64)) doesn't show a difference at least to my eyes.



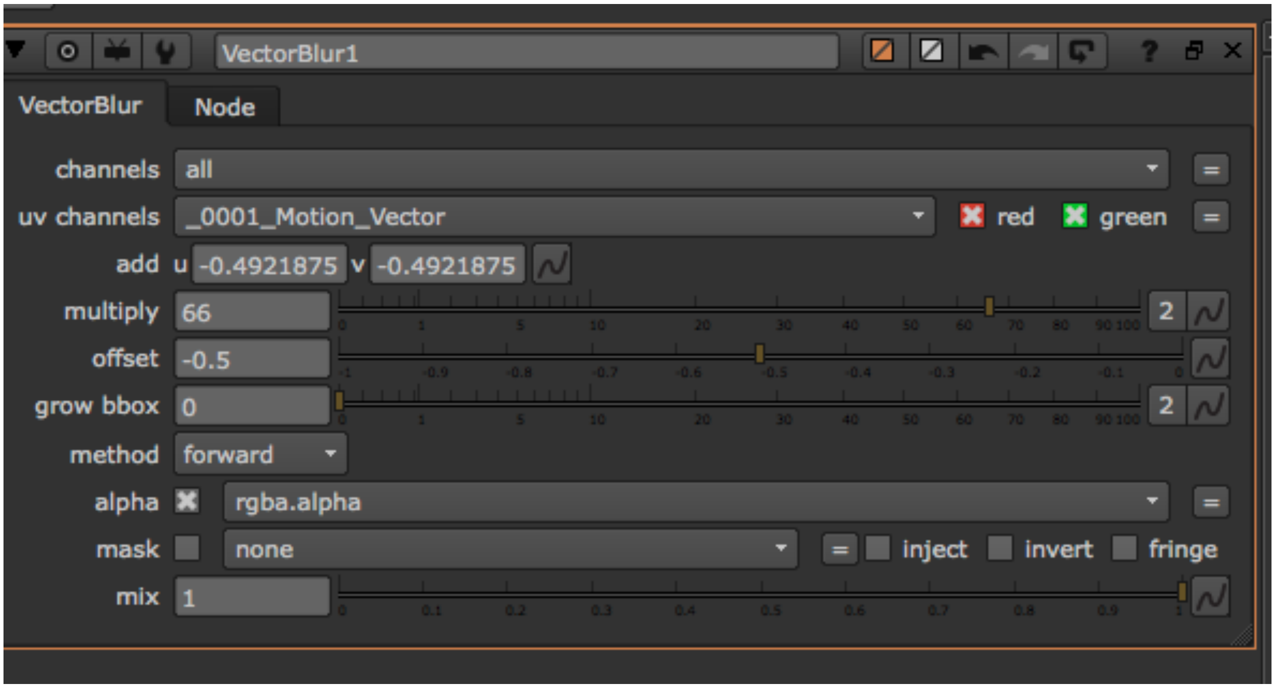
-1*(((64 - 1) * .5) / (64)

About 0 results (0.94 seconds)

(-1) * (((64 - 1) * .5) / 64) =

-0.4921875

Next is the multiply field. Set this to a number that matches the MB you want. I left my offset at -.5 but you can adjust that if you are getting errors etc. but -.5 will be consistent with C4D's motion blur. For the method section of the Vector Blur node I recommend forward it will give more accurate results. Also, be sure to check alpha and use the appropriate channel. For this scene I rendered out an object buffer of my boxes and simply shuffled that to the alpha channel for the stream.



VectorBlur1

VectorBlur | Node

channels: all

uv channels: _0001_Motion_Vector ☒ red ☒ green

add u -0.4921875 v -0.4921875

multiply: 66

offset: -0.5

grow bbox: 0

method: forward

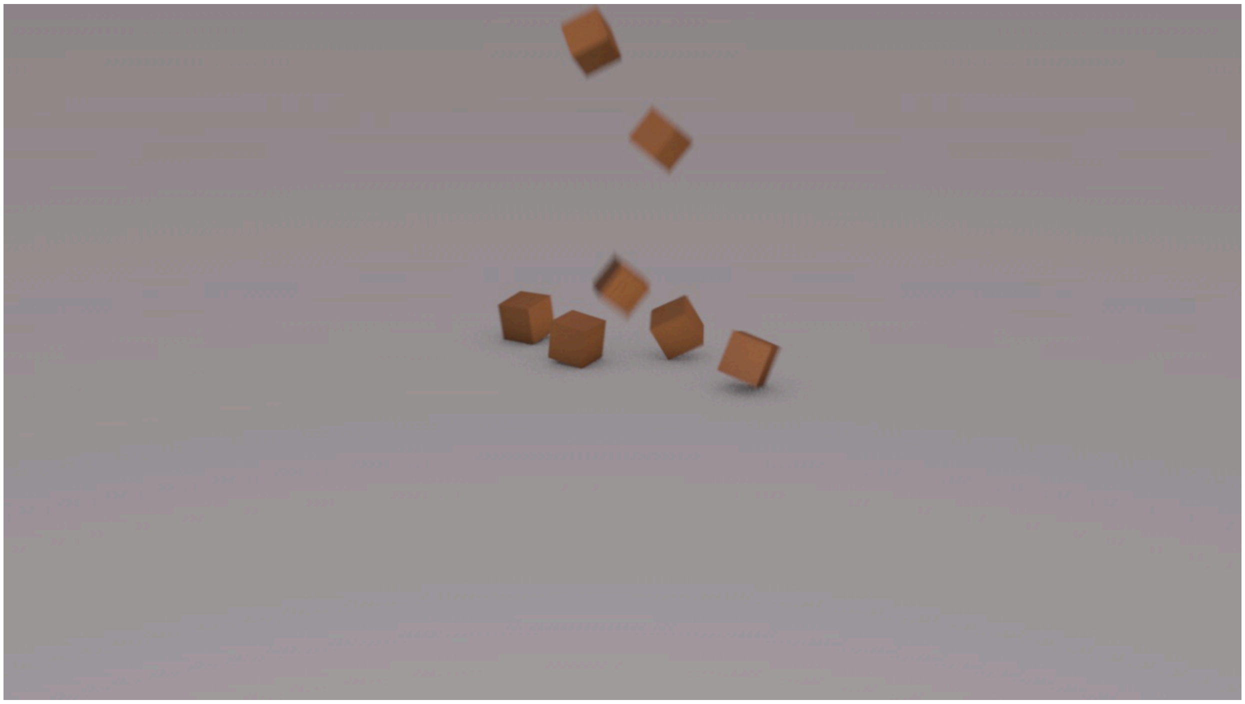
alpha ☒ rgba.alpha

mask □ none □ inject □ invert □ fringe

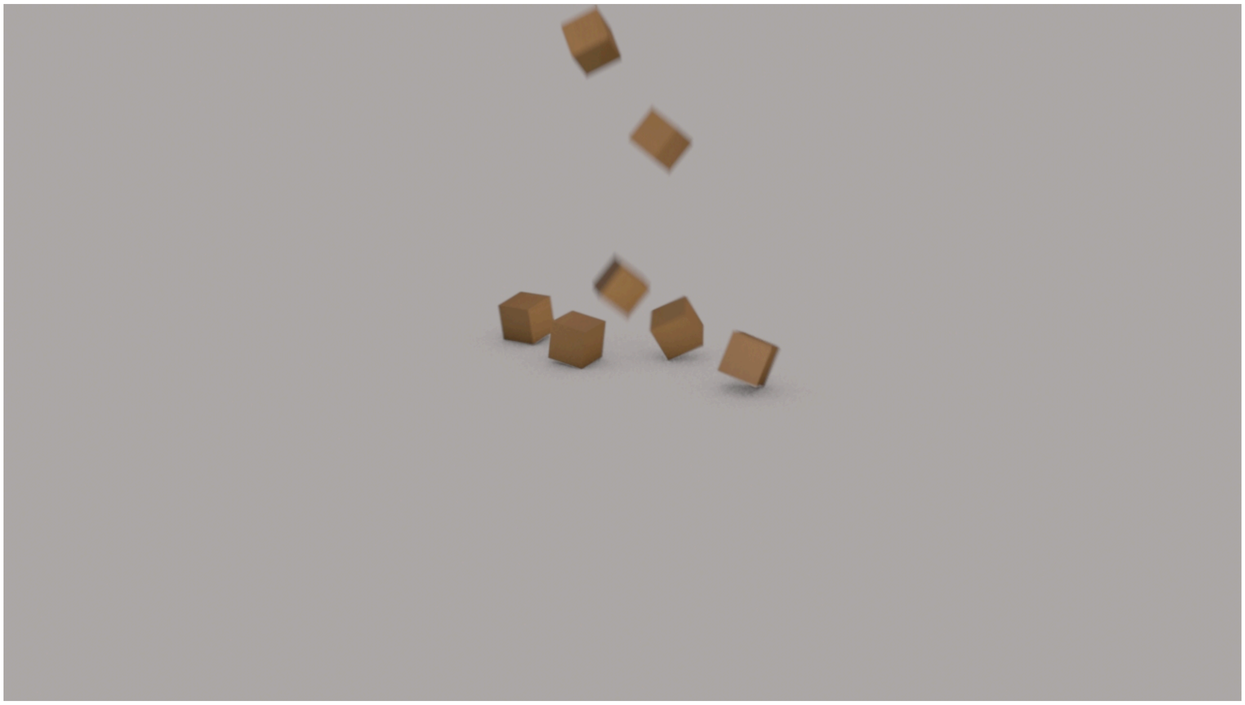
mix: 1

That is all. Below are three pictures showing MB in C4D (Physical MB), NUKE (Vector Blur) and AE (RSMB Pro). Shown in this order. As you can see you can get very useable results with C4D’s Motion Vector pass in NUKE. Ignore the color differences :).

C4D (Physical Motion Blur)



NUKE (Vector Blur)



AE (RSMB Pro)