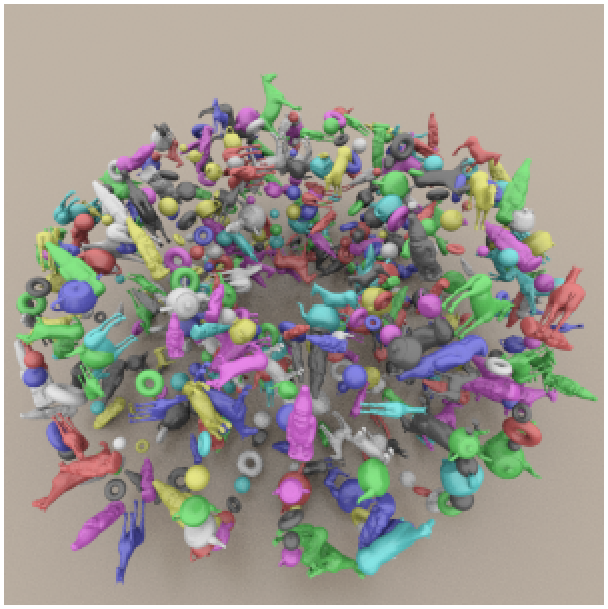
# Motion Vectors

Motion blur in Arnold is generally best accomplished by using the natural motion blur support, where transforms and vertices of objects are sampled over time from a camera's shutter open to shutter close. This requires using more AA samples to resolve noise from motion blur, but often those samples are needed anyway for noise resulting from the depth of field, direct and indirect lighting, volumetrics, and so on. In some cases where there are severe time constraints, it may be faster to instead output *motion vectors* and blur the images in a compositing package afterward. This type of motion blur is lower fidelity; it does not capture lighting changes as an object moves, nor complex interactions of depth relative to the camera. However, it can be sufficient in some cases.

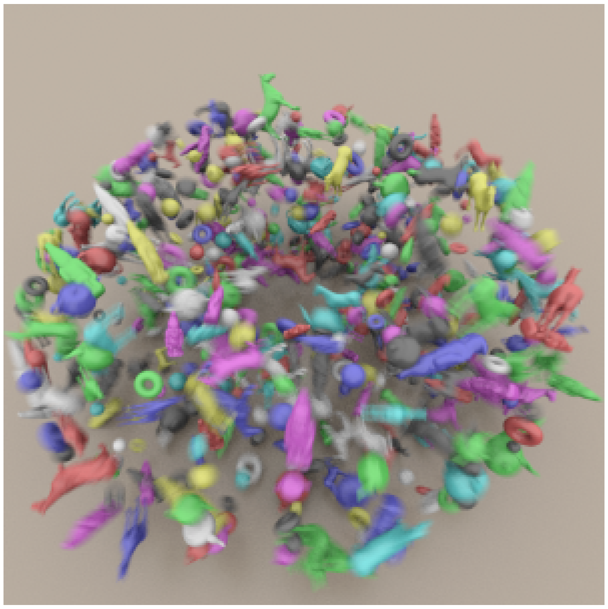
We demonstrate how to generate motion vectors for the cases where it is desirable. There are two methods to generate motion vectors, one using a shader and the other using a built-in AOV that Arnold provides. In both cases, an `ArnoldGlobalSettings` node must have the `output_motion_vectors` parameter checked and activated for motion vectors to be output.

## Scene Setup

The scene should be set up normally, with motion and animation defined as it normally would be.  In this example, we have motion generated by a script and then applied to vertices of various instanced objects.  The true motion blur takes this static scene and motion blurs it as follows:



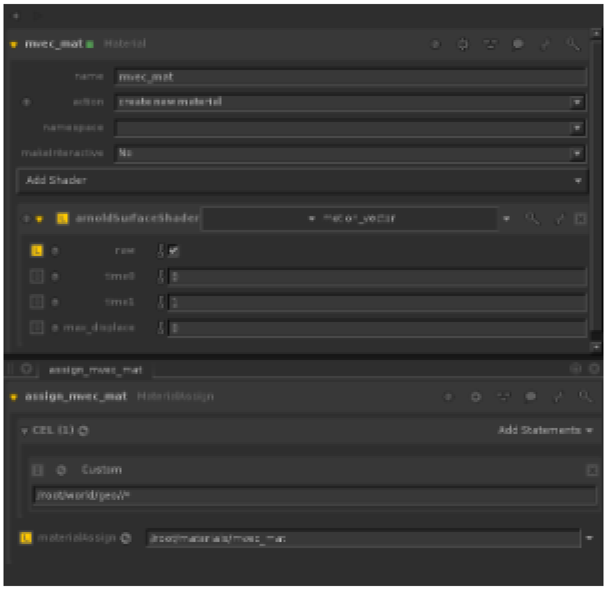
No motion



Shutter 0.0 to 1.0 (true motion blur)

## Motion Vector Shader

There are two ways to output motion vectors in Arnold; the first is to create a separate motion vectors pass, where all objects are assigned the `motion_vector` shader.  This is perhaps the most controllable, but it does require a separate render pass (and thus more time).  In this approach, a Material node is created with an Arnold surface shader of type motion_vector set, and it is assigned to all objects in the scene:



mvec_mat Material

name mvec_mat

action create new material

namespace

makeInteractive No

Add Shader

arnoldSurfaceShader motion_vector

raw ☑

time0 0

time1 1

max_displace 0

assign_mvec_mat MaterialAssign

CEL (1)

Add Statements

Custom

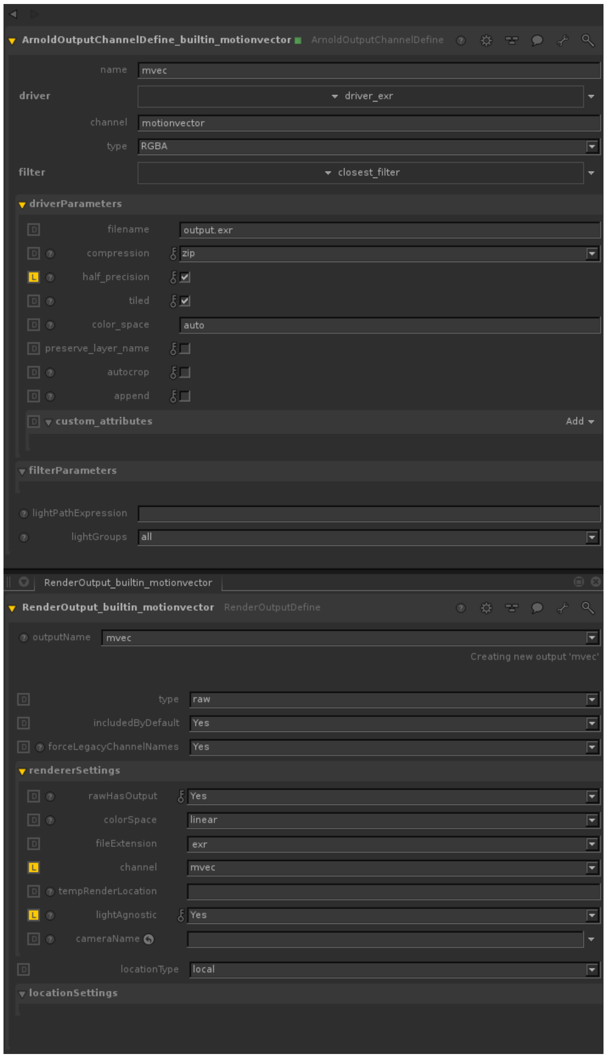
/root/world/geo//*

materialAssign /root/materials/mvec_mat

The values of the motion vectors themselves can be controlled with the parameters of the `motion_vector` shader, giving this method a little more control over the vector output.

## Motion Vector AOV

Arnold provides a built-in AOV for motion vectors.  By creating `ArnoldOutputChannelDefine` and associated `RenderOutputDefine` nodes, both the regular primary (unblurred) output and the motion vector data can be rendered in a single pass:



ArnoldOutputChannelDefine_builtin_motionvector ArnoldOutputChannelDefine

name: mvec

driver: driver_exr

channel: motionvector

type: RGBA

filter: closest_filter

driverParameters

- filename: output.exr
- compression: zip
- half_precision: ☑
- tiled: ☑
- color_space: auto
- preserve_layer_name: □
- autocrop: □
- append: □
- custom_attributes Add

filterParameters

lightPathExpression:

lightGroups: all

RenderOutput_builtin_motionvector

RenderOutput_builtin_motionvector RenderOutputDefine

outputName: mvec

Creating new output 'mvec'

type: raw

includedByDefault: Yes

forceLegacyChannelNames: Yes

rendererSettings

- rawHasOutput: Yes
- colorSpace: linear
- fileExtension: exr
- channel: mvec
- tempRenderLocation:
- lightAgnostic: Yes
- cameraName:

locationType: local

locationSettings

## Results

Motion vectors visually end up appearing like the following, and the EXR results can then be used to blur the primary RGBA output: