Image Sequence Mipmapping

This document provides an overview of using mipmapping for image sequences.

Introduction

This document provides an overview of using mipmapping for image sequences.

As of 5.1, you can use the Process EXR tool to generate `.exr` files with the correct settings to use with the Media Plate Actor

Mipmaps

Mipmaps are used in image sequences to reduce the amount of data loaded in.

Mipmaps are not imported into the Engine as a normal texture, therefore there will not be a UAsset created for the EXR mipmaps. Due to this behavior, the mip levels need to be generated manually before using them in the Engine. See Using Nuke and Python Scripts to Create EXR Mipmaps for how to generate your own mipmaps.

Limitations

- Only EXR files are supported at this time.
- Anisotropic mips are not supported.
- Reduction of data loaded in is achieved by only loading the required mip levels.
- For GPU optimized streaming into Unreal, all EXR Image Sequences should be uncompressed.
- You will only get GPU acceleration for loading EXRs if they are uncompressed.
- All mip levels need to be either compressed or un-compressed. Having one level compressed and another uncompressed will break the chain.

File Directory Structure

The mipmap file directory structure should look like this, which follows the Cineon naming convention and is the industry standard.

```
Smoke_Element/2048x2048/Smoke_Element.00001.exr
Smoke_Element/1024x1024/Smoke_Element.00001.exr
Smoke_Element/512x512/Smoke_Element.00001.exr

```

Copy full snippet

File and folder names must follow these rules:

- All folder names need to be a power of two, for example 1024×512.
- Mip level images should be named exactly the same as the source image.

Editor Setup

To use the mipmap functionality, the `ImgMediaPlayback` component must be added to all objects that display the images. If the component is not present on an object, then the object will not be used in determining what mip levels should be used.

You can enable debugging with the following console command:

`ImgMedia.MipMapDebug 1`

This will display which mip level each image sequence is currently using.

Mip Level Selection

Mip levels are selected based on an estimated pixel to texel density for each object displaying the image.

The camera position is used for these calculations, so fast moving cameras can introduce more errors in the estimation. Further work could improve the estimation and take camera movement into account.

The mip level selected can be manually adjusted with the **LODBias** setting on the `ImgMediaPlayback` component.

Using Nuke and Python Scripts to Create EXR Mipmaps

Nuke and Python scripts are available to aid you in the auto-generation of your mip levels. Each script is available for both downloading and viewing on this page.

nukeMipMap.py is a Python script you can run in Nuke that will set up the proper LOD generating tree.

- To use the script, select your EXR sequence, then set the number of mip levels desired.
- At the top of the script, select the read nodes you want to create mip maps for, and execute.
- All the necessary reformats and write nodes will be created for you in Nuke.
- All the necessary resolution folders for each mip render will be created for you.
- The paths of the mip renders will be based on the selected read node on execution.

nukeMipMap.py

```
# All sequences need to live in a folder named after the resolution of the images.
# Example - D:/Perforce/EXR_Sequences/Smoke/2048x2048/
```

```
D:/Perforce/EXR_Sequences/Smoke/1024x1024/
D:/Perforce/EXR_Sequences/Smoke/512x512/

# For GPU optimized streaming into Unreal all Exr image Sequences should be uncompressed.
# You will only Get GPU acceleration for loading Exr files if they are uncompressed.
# For faster loading into Unreal, the uncompressed format is preferable.
# All mip levels need to be either compressed or uncompressed. Having one level compressed and another uncompressed will break the chain.
# All mip levels need to be named exactly the same as the source.
# To use the script select your Exr sequence, then set the number of mip levels desired.
# At the top of the script, select the read nodes you want to mip, and execute.
# All the necessary reformats and write nodes will be created for you in Nuke.
# All the necessary resolution folders for each mip render will be created for you.
# All folder names need to be power of 2. Example - (128,256,512,1024)
# The paths of the mip renders will be based on the selected read node on execution.

import nuke
import os

#How many mip levels do you want?
mipLevels = 3

#Grabs node selection
selectedRead = nuke.selectedNodes()
addLevel = mipLevels + 1

#Gets Height and Width of Image Sequence
def getHeightWidth(read):
getFormat = []
getHeight = []
getWidth = []
getFormat = read.format()
getHeight = getFormat.height()
getWidth = getFormat.width()
dirResName = str(getWidth) + 'x' + str(getHeight)
```

```

return dirResName

def getFilePathName(readNode):
getName = readNode['file'].value()

return getName

#Create directory
def createDirectories(readNode,read):
getNameLocal = getFilePathName(readNode)
getHeightWidthLocal = getHeightWidth(read)
getSequenceName = []
parentPath = []
dirResName = []
dirName = []
setRenderPathName = []
getSequenceName = getNameLocal.split('/')[-1]
parentPath = getNameLocal.split(getSequenceName)[0]
dirName = parentPath + getHeightWidthLocal

#Sets Render Path Name
setRenderPathName = dirName + '/' + getSequenceName
isThere = os.path.isdir(dirName)
if isThere == False:
os.makedirs(dirName)

return setRenderPathName

#Creates reformat
def createReformatNodes(connectReformat):
createScale = nuke.nodes.Reformat()
createScale['type'].setValue("scale")
createScale['scale'].setValue(0.5)
createScale.connectInput(1,connectReformat)
return createScale

#Creates write node
def createWriteNodes(path,connect):
createWrite = nuke.nodes.Write()
createWrite['file'].setValue(path)
createWrite['file_type'].setValue('exr')
```

```
createWrite['compression'].setValue('none')
createWrite.connectInput(1,connect)
return createWrite

#Generate Tree
if len(selectedRead) > 0:
for x in selectedRead:
getFilePathName(x)
for index in range(addLevel):
if index == 0:
getHeightWidth(x)
setPathLocal = createDirectories(x,x)
createWriteLocal = createWriteNodes(setPathLocal,x)
else:
createScaleLocal = createReformatNodes(createWriteLocal)
getHeightWidth(createScaleLocal)
setPathLocal = createDirectories(x,createScaleLocal)
createWriteLocal = createWriteNodes(setPathLocal,createScaleLocal)
else:
nuke.alert("Nothing selected. Please select your EXR sequence READ NODES to generate mipmaps")

```

Copy full snippet

Unreal_ExrMipMap_GenerationExample.nk is an example Nuke script with a tree already created for you. It demonstrates what folder structure is required for the mipmapping to generate the proper LOD scaling and displays the write node configuration without compression.

Unreal_ExrMipMap_GenerationExample.nk

```
#! C:/Program Files/Nuke12.0v3/nuke-12.0.3.dll -nx
version 12.0 v3
define_window_layout_xml {<?xml version="1.0" encoding="UTF-8"?>
<layout version="1.0">
<window x="-1" y="-8" w="2560" h="1377" maximized="1" screen="0">
<splitter orientation="1">
<split size="40"/>
<dock id="" hideTitles="1" activePageId="Toolbar.1">
<page id="Toolbar.1"/>
```

```
</dock>
<split size="2516" stretch="1"/>
<splitter orientation="2">
<split size="1333"/>
<dock id="" activePageId="DAG.1" focus="true">
<page id="DAG.1"/>
<page id="Curve Editor.1"/>
<page id="DopeSheet.1"/>
</dock>
</splitter>
</splitter>
</window>
<window x="3219" y="212" w="1885" h="746" screen="1">
<splitter orientation="2">
<split size="746"/>
<dock id="" activePageId="Viewer.1">
<page id="Viewer.1"/>
</dock>
</splitter>
</window>
</layout>
}
Root {
inputs 0
name C:/Users/Desktop/EXR_Mipmap/Unreal_ExrMipMap_GenerationExample.nk
format "2048 1556 0 0 2048 1556 1 2K_Super_35(full-ap)"
proxy_type scale
proxy_format "1024 778 0 0 1024 778 1 1K_Super_35(full-ap)"
colorManagement Nuke
workingSpaceLUT linear
monitorLut sRGB
int8Lut sRGB
int16Lut sRGB
logLut Cineon
floatLut linear
}
BackdropNode {
inputs 0
name LOD1
tile_color 0x999dbcff
gl_color 0x3f4cccff
```

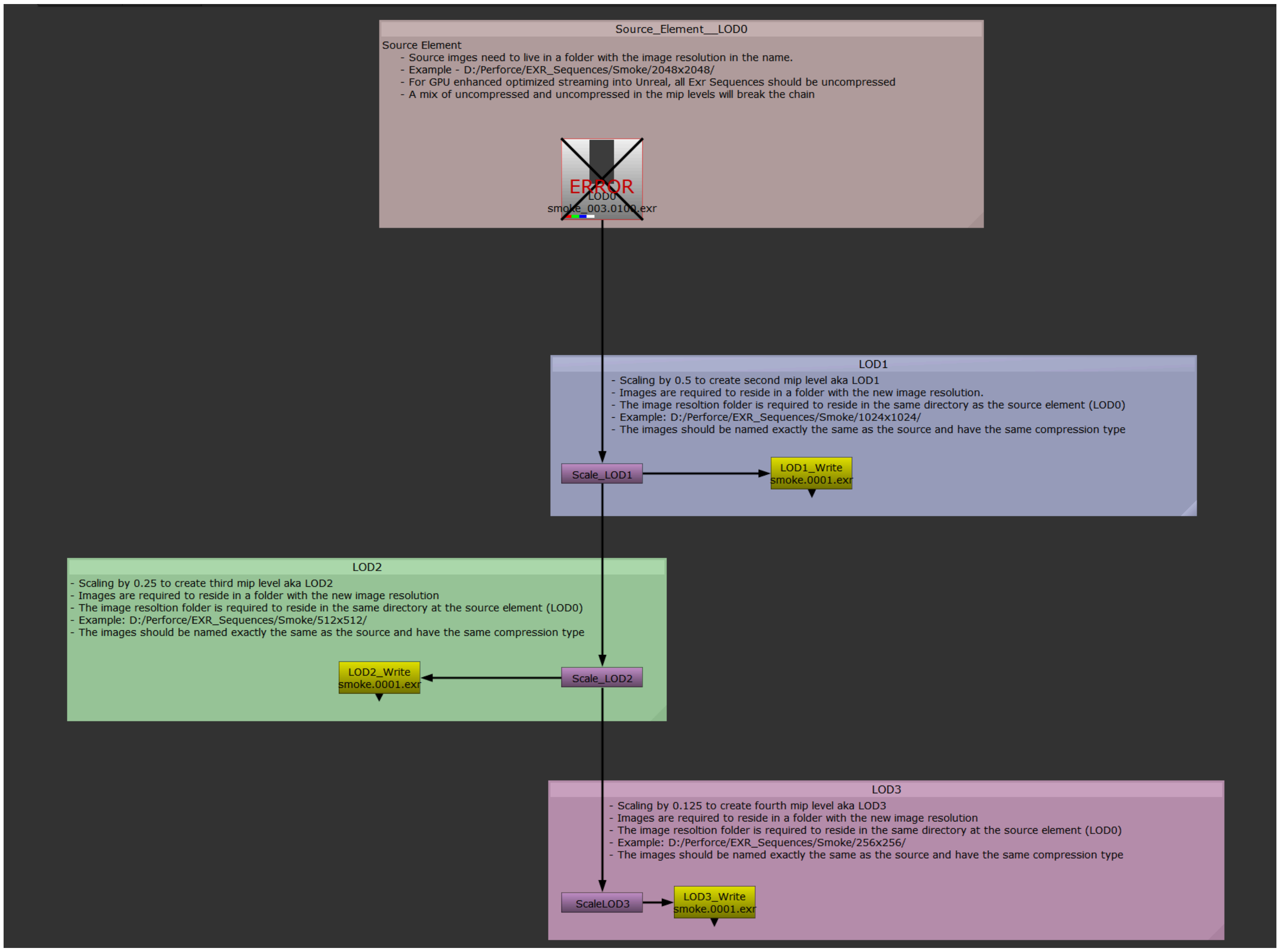
```
label "\t- Scaling by 0.5 to create second mip level aka LOD1\n\t- Images are required to reside in a folder with the new image resolution.\n\t- The image resolution folder is required to reside in the same directory as the source element (LOD0)\n\t- Example: D:/Perforce/EXR_Sequences/Smoke/1024x1024/\n\t- The images should be named exactly the same as the source and have the same compression type"
xpos -584
ypos -19
bdwidth 633
bdheight 157
}
BackdropNode {
inputs 0
name LOD2
tile_color 0x96c499ff
gl_color 0x73cc71ff
label "- Scaling by 0.25 to create third mip level aka LOD2\n- Images are required to reside in a folder with the new image resolution\n- The image resolution folder is required to reside in the same directory as the source element (LOD0)\n- Example: D:/Perforce/EXR_Sequences/Smoke/512x512/\n- The images should be named exactly the same as the source and have the same compression type"
xpos -1057
ypos 180
bdwidth 587
bdheight 160
}
BackdropNode {
inputs 0
name LOD3
tile_color 0xb790aaff
label "\t- Scaling by 0.125 to create fourth mip level aka LOD3\n\t- Images are required to reside in a folder with the new image resolution\n\t- The image resoltion folder is required to reside in the same directory at the source element (LOD0)\n\t- Example: D:/Perforce/EXR_Sequences/Smoke/256x256/\n\t- The images should be named exactly the same as the source and have the same compression type"
xpos -586
ypos 398
bdwidth 662
bdheight 156
}
```

```
BackdropNode {
inputs 0
name Source_Element__LOD0
tile_color 0xaf9f9fff
label "Source Element\n - Source images need to live in a folder with the image resolution in the name.\n - Example - D:/Perforce/EXR_Sequences/Smoke/2048x2048/\n - For GPU enhanced optimized streaming into Unreal, all Exr Sequences should be uncompressed\n - A mix of uncompressed and uncompressed in the mip levels will break the chain"
selected true
xpos -751
ypos -347
bdwidth 592
bdheight 204
}
Read {
inputs 0
file_type exr
file D:/Perforce/Project/Movies/EXR_Sequences/AtmosSmoke_003/smoke_003.####.exr
format "1152 2048 0 0 1152 2048 1 "
first 100
last 360
origfirst 100
origlast 360
origset true
in_colorspace scene_linear
out_colorspace scene_linear
name LOD0
selected true
xpos -573
ypos -230
disable true
}

Reformat {
type scale
scale 0.5
name Scale_LOD1
xpos -573
ypos 87
}
```

```

set N9841b000 [stack 0]
Reformat {
type scale
scale 0.5
name Scale_LOD2
xpos -573
ypos 287
}

set N9841a800 [stack 0]
Reformat {
type scale
scale 0.5
name ScaleLOD3
xpos -573
ypos 507
}

Write {
file D:/Perforce/EXR_Sequences/Smoke/256x256/smoke.####.exr
file_type exr
compression none
first_part rgba
version 5
in_colorspace scene_linear
out_colorspace scene_linear
name LOD3_Write
xpos -463
ypos 501
}

push $N9841b000

Write {
file D:/Perforce/EXR_Sequences/Smoke/1024x1024/smoke.####.exr
file_type exr
compression none
first_part rgba
version 6
in_colorspace scene_linear
out_colorspace scene_linear
name LOD1_Write
xpos -368
ypos 81
}

push $N9841a800

Write {
file D:/Perforce/EXR_Sequences/Smoke/512x512/smoke.####.exr
file_type exr
compression none
first_part rgba
version 5
in_colorspace scene_linear
out_colorspace scene_linear
name LOD2_Write
xpos -791
ypos 281
}
```

Copy full snippet

The generated image below displays an example screenshot of the `Unreal_ExrMipMap_GenerationExample.nk` script.



Source_Element__LOD0

Source Element

- Source imges need to live in a folder with the image resolution in the name.
- Example - D:/Perforce/EXR_Sequences/Smoke/2048x2048/
- For GPU enhanced optimized streaming into Unreal, all Exr Sequences should be uncompressed
- A mix of uncompressed and uncompressed in the mip levels will break the chain

LOD1

- Scaling by 0.5 to create second mip level aka LOD1
- Images are required to reside in a folder with the new image resolution.
- The image resoltion folder is required to reside in the same directory as the source element (LOD0)
- Example: D:/Perforce/EXR_Sequences/Smoke/1024x1024/
- The images should be named exactly the same as the source and have the same compression type

LOD2

- Scaling by 0.25 to create third mip level aka LOD2
- Images are required to reside in a folder with the new image resolution
- The image resoltion folder is required to reside in the same directory at the source element (LOD0)
- Example: D:/Perforce/EXR_Sequences/Smoke/512x512/
- The images should be named exactly the same as the source and have the same compression type

LOD3

- Scaling by 0.125 to create fourth mip level aka LOD3
- Images are required to reside in a folder with the new image resolution
- The image resoltion folder is required to reside in the same directory at the source element (LOD0)
- Example: D:/Perforce/EXR_Sequences/Smoke/256x256/
- The images should be named exactly the same as the source and have the same compression type

autoGenEXR_mipmap.py is a script for users who do not have Nuke. This Python script will automatically create the necessary folders in addition to scaling and writing out the mip levels to the disk.

In order for this script to work, you will need to install the following free python modules numpy,opencv, and shutil.

autoGenEXR_mipmap.py

```python
import os
os.environ["OPENCV_IO_ENABLE_OPENEXR"]="1"
import cv2 as cv
import numpy as np
import glob
import shutil

```

```python
setMipLevel = 3
fileInDir = glob.glob("C:\\Users\\User\\Desktop\\smokeCards\*.exr")

#Gets file path to parent sequence
grabFirst = fileInDir[0]
splitFile = grabFirst.split('\\')[-1]
getParentPath = grabFirst.replace(splitFile,'')

#Getting image resolution
img = cv.imread(grabFirst, cv.IMREAD_UNCHANGED)
height = np.size(img, 0)
width = np.size(img, 1)

#Creates folders
def createFolders(dirName):
isThere = os.path.isdir(dirName)
if isThere == False:
os.makedirs(dirName)

#Builds folder path for LOD0 and copies sequence to LOD0 folder
LOD0_folderName = getParentPath + str(width) + 'x' + str(height)
createFolders(LOD0_folderName)

for x in fileInDir:
getFileName = x.split('\\')[-1]
newFile = LOD0_folderName + '\\' + getFileName
if os.path.isfile(newFile) == 0:
shutil.copyfile(x, newFile)
print('copying ' + newFile + ' to correct file path')

#Creates mipped EXR files
def createFiles(file, mipWidth, mipHeight, folderPath):
getName = file.split('\\')[-1]
imageSize = (int(mipWidth), int(mipHeight))
readFile = cv.imread(file, cv.IMREAD_UNCHANGED)
resizeFile = cv.resize(readFile, imageSize, interpolation =
cv.INTER_LANCZOS4)
newFile = folderPath + '\\' + getName
cv.imwrite(newFile, resizeFile, [cv.IMWRITE_EXR_TYPE,
cv.IMWRITE_EXR_TYPE_HALF])
print("saving mipped file to " + newFile)

```

```
#Does math for the mip levels, creates mipped folders and files
for index in range(setMipLevel):
if index == 0:
newWidth = width / 2
newHeight = height / 2
LOD1FolderPath = getParentPath + str(int(newWidth)) + 'x' + str(int(newHeight))
createFolders(LOD1FolderPath)
for file in fileInDir:
createFiles(file, newWidth, newHeight, LOD1FolderPath)
else:
mipWidth = newWidth / 2
mipHeight = newHeight / 2
newWidth = mipWidth
newHeight = mipHeight
lowerMipFolderPath = getParentPath + str(int(newWidth)) + 'x' + str(int(newHeight))
createFolders(lowerMipFolderPath)
for file in fileInDir:
createFiles(file, mipWidth, mipHeight, lowerMipFolderPath)

```

Copy full snippet