Textures

When using Arnold it is best to use a tiled mipmapped texture format such as .exr or .tx that has been created using maketx.

.tx textures are:

1. Tiled (usually the tiles are 64×64 pixels).
2. Mip-mapped.

If you already have tiled and mipmapped EXR's that have been created by another renderer, you won't need to convert those files to .tx

Due to (1), Arnold’s texture system can load one tile at a time, as needed, rather than having to wastefully load the entire texture map in memory. This can result in faster texture load times, as texels that will never be seen in the rendered image will not even be loaded. In addition to the speed improvement, only the most recently used tiles are kept in memory, in a texture cache of default size 512 MB (can be tuned via options. texture_max_memory_MB). Tiles that have not been used for a long time are simply discarded from memory, to make space for new tiles. Arnold will never use more than 512 MB, even if you use hundreds or thousands of 4k and 8k images. But then, if you only use a handful of 1k textures, this will not matter.

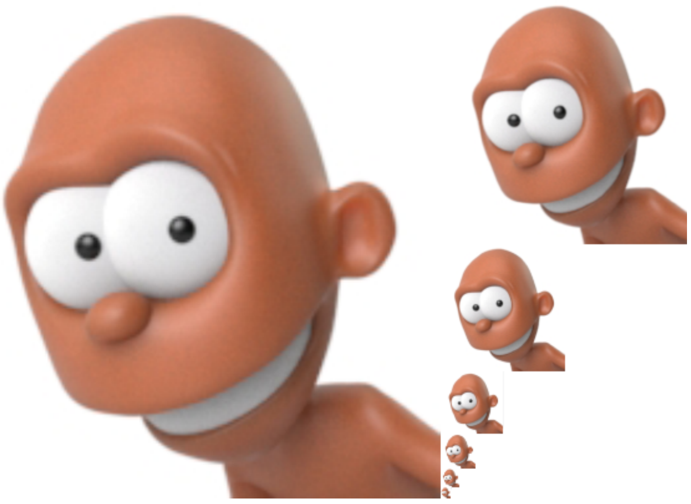
Due to (2), the textures are anti-aliased, even at low AA sample settings. Neither of these is possible with JPEG or other untiled/unmipped formats (unless you tell Arnold to auto-tile and auto-mip the textures for you, but this is very inefficient because it has to be done per rendered frame, rather than a one-time pre-process with maketx).

It should be worth pointing out that .tx files are basically .exr or .tif files that have been renamed to .tx. This means .tx files can be read by image editors, although you may need to rename the file to .exr or .tif so that the image editor will load it. However, .tx files have a few extra custom attributes set that are not normally included in .exr/.tif files which can make rendering even faster. For instance, they will include a hash so that if you try to load two different files that contain the same data, Arnold only needs to load this data once.

Benefits of Using .TX

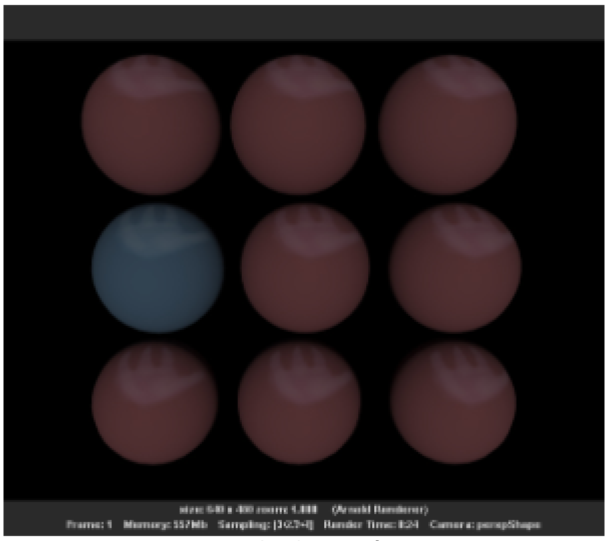
The first benefit is that you are assured of having mipmaps and tiles. These will dramatically improve time to the first pixel, overall render time, and allow using a smaller texture cache. This should be considered mandatory, with the possible exception being don't use .tx for the few images you are actively modifying if you don't want to wait for them to be constantly converted to .tx every time you make a change.

The previous level you could get from non-tx files that you saved with tiles and mipmapping. The second level of benefit is only with .tx files and that involves maketx adding metadata that lets Arnold make more optimizations, such as being able to detect duplicate textures and only loading a single copy into memory or detecting constant color images, such as an all-black UDIM, and instead of storing all those black pixels in memory Arnold can special case that.

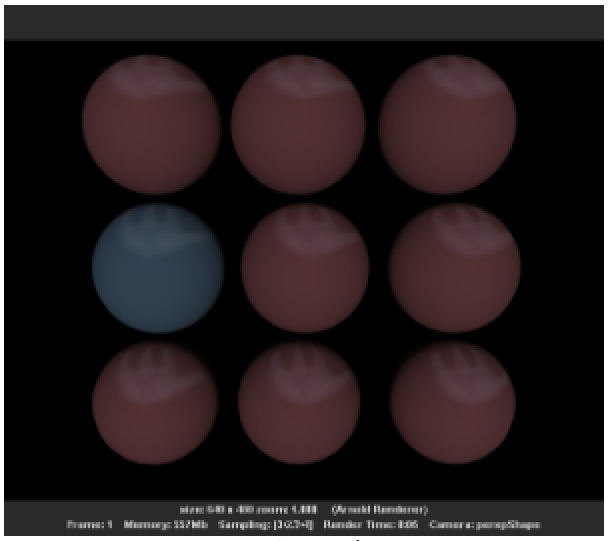


An example of mipmap image storage. The first image on the left is shown with filtered copies reduced in size.

The example below shows a speed increase of seven times when using .tx files compared to .jpg files:



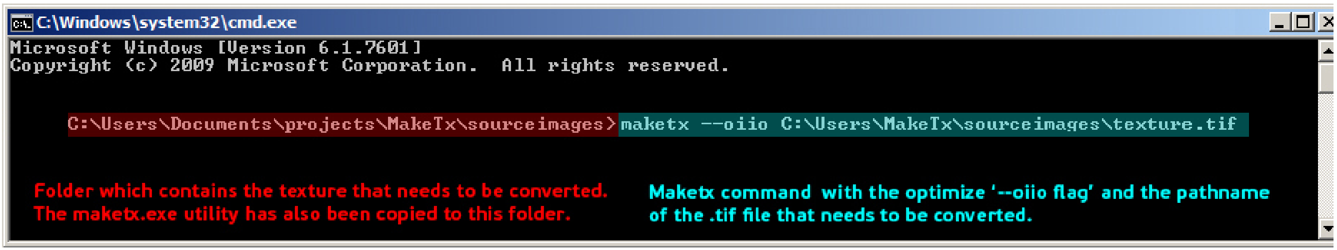
JPEG: 42 seconds



TX: 6 seconds

Manually Generate TX Textures

The process is quite simple. You will need the maketx.exe utility, the texture that needs to be converted, and a dos shell. Below is an example of how you would convert a .tif file to a .tx file.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.


    C:\Users\Documents\projects\MakeTx\sourceimages>maketx --oiio C:\Users\MakeTx\sourceimages\texture.tif
```

Folder which contains the texture that needs to be converted.
The maketx.exe utility has also been copied to this folder.

Maketx command with the optimize '--oiio flag' and the pathname of the .tif file that needs to be converted.

A tutorial that covers this process can be found here.

Auto-generate TX Textures

By default, tiled and mipmapped TX textures are automatically generated for each *image* shader. The resulting TX will be placed next to the original texture files. When the texture filename contains tags such as UDIMs, a TX texture will be generated for each sub-tile.

It can take a bit of time to convert a texture to TX, especially for large textures stored on a network share, but usually, this is done only for the first render. For subsequent renders, if an existing matching TX texture is detected, it won't be regenerated unless the source texture contents or colorspace has changed. Also note that if the input texture filename already has a `.tx` extension, it will be left as-is.

Disabling auto-generation

This behavior can be disabled per texture with the ***Auto-generate TX Textures*** toggle on the *Image* VOP and a similarly named toggle can be found in the **Textures** tab of the Arnold render settings to disable TX auto-generation globally.

Linearization

In addition, the TX texture will be linearized according to the ***Colorspace*** parameter. The linear, sRGB and Rec.709 colorspaces are currently supported. The default value ***Auto*** will heuristically determine the colorspace from the bit depth and type of the texture.

IPR

Changes to the texture filename and colorspace will correctly trigger IPR changes and rebuild the TX texture if necessary. The ***Refresh*** button next to the filename on the *Image* VOP can be used to force the regeneration of TX files.

Updating all textures

The ***Arnold > Update TX Textures*** menu entry will update all TX textures in the scene. Alternatively, you may use the following Python function:

```
import htoa.material
htoa.material.updateAllTx()
```