Dark Ages Texture Mods

Getting Started

Creating texture mods requires the following resources:

1. **Atlan Resource Extractor** - Use this to extract material2 decls from the game. These plaintext files define the game's materials and directly reference image assets.

2. **Valen** - A resource extractor that can extract vanilla image files from the game. Atlan Resource Extractor does *NOT* support extracting textures. Use this instead.

3. **Atlan Mod Packager** - Packages your custom texture PNG files into a format the game recognizes. This is a necessary step to reduce mod loading times, as converting the images is an expensive process. You must use Atlan Mod Packager to prepare your finished texture mods for distribution

4. **Mod Config File** - It is *highly recommended* you use the aliasing system provided by your mod's configuration file! Image files have long names that don't clearly describe what piece of a mesh they're for! The aliasing system is helpful for organization, and prevents you from needing to change the extension on your PNGs to match the vanilla filename!

Supplemental resources may also exist for common areas of texture modding:

1. Custom Slayer Skins: Check out Kuddly Kraken's **Creation Kit**

Assembling Your Mod Zip

All image mods must have `image/` at the beginning of their filepath (or configuration alias). After this, simply recreate the path of the image file you want to replace. **This includes the extension properties.** For Example:

```
image/models/customization/characters/doomslayer/set_b/1001/doomslayer_set_b_torso_1001.tga$streamed$mtlkind=albedo
image/models/customization/characters/doomslayer/set_b/1001/doomslayer_set_b_torso_1001.tga
```

Path #1 refers to a vanilla image file. However, Path #2 does not exist and the game will not recognize your image. **Extension metadata is important! It must be recreated perfectly when you want to replace an existing image.**

As with other resource types, an `@` or a `\` can be used in place of a `/` character.

Aliasing

As you can see, these image names grow long and complicated very quickly. The aliasing system can alleviate this problem. Lets say we have an image called `helmet_doodles.png` and we want it to replace the above image. Using our config aliasing system:

```
aliasing = {
    "helmet_doodles.png" =
"image/models/customization/characters/doomslayer/set_b/1001/doomslayer_set_b_torso_1001.tga$
streamed$mtlkind=albedo"
}
```

This is equivalent to renaming your PNG to that filepath.

Developing and Packaging

Rinse and repeat this process until you've built out your entire texture mod folder. When you're ready to ship, run AtlanModPackager to package it into a zip file ready for loading your mods!

While developing and testing your mod, remember to take advantage of Atlan Mod Loader's **unzipped mod loading capabilities.**

Sample Mod

The following link takes you to a sample mod implementing the above example! It uses the aliasing system to replace the Nightmare Slayer skin's helmet texture! To test this mod:

1. Download and unzip it's contents
2. Run AtlanModPackager on the unzipped folder
3. Use Atlan Mod Loader to load the newly created mod!

Google Drive Link

Adding New Images

> Adding original images to the game is more complicated than replacing existing textures. Please pay careful attention to all of the information in this section. If all you're doing is replacing existing textures, you can skip this section

If you want to add an original image to the game, instead of replacing an existing image, you'll need to follow several more steps:

1. Reference your new images inside one (or multiple) material2 decls.
2. Determine the correct extension properties for your image
3. Specify your image's encoding information. This step is not always necessary depending on the image.

Material2 Decls

Once you've loaded your new image assets into the game, you must reference them in one or more material2 decls! You can edit a vanilla decl to use your texture, or create an entirely new material2 to reference in other files. As the mod author, you make the design decisions!

Extension Properties

When deciding the asset name of your new image, you can choose any unique name:

```
image/my_awesome_image.tga
image@your_username_here@your_modname_here@image_1.tga
image/cool_skin/helmet/helmet_albedo.png
```

However, the extension properties **must be written correctly.** If you fail to write them accurately, the game will not be able to find and load your modded image!

The exact type varies by the type of image you're adding, and by the inherited template of the material2 decl you're using it in. For example, if you're adding an albedo image, your extension properties should be `$streamed$mtlkind=albedo`

There is no exhaustive list of correct extension properties. The best way to be certain you're correct, is by studying the names of vanilla images produced by that particular material2 template. More obscure templates may require some trial and error to get the extension properties correct.

Encoding Information

When adding a new image, Atlan requires two key pieces of information: the image format (i.e. BC7, BC1, etc.) and the material type (i.e. albedo, normal, specular, etc.)

For *most* cases, you probably won't need to worry about this. Atlan will analyze the extension properties discussed above, and correctly deduce these settings. However, there are cases where the extension properties are insufficient for multiple reasons:

1. Ambiguity: Depending on the material template, the format cannot be determined with absolute certainty! For example an image with properties `$streamed$mtlkind=blendmask` can use different formats depending on the material

type!

2. Absence: Some material templates omit extension properties from their images entirely!
3. In very rare cases, the extension properties may flat-out lie about the type or format used.

In these circumstances, you must specify the correct type and format by adding them to the end of the image name (or alias name, if it has an alias). For Example:

```
aliasing = {
    "my_blendmask.png" =
"image/my_blendmask.tga$streamed$mtlkind=blendmask~FMT_BC7~TMK_BLENDMASK"
}
```

As you can see, the format is `~FORMAT_CODE~MATERIAL_TYPE_CODE`

The following code block lists all possible values for these fields. Determining the correct values for a given image may take some careful analysis, depending on the material template.

```
enum textureMaterialKind_t {
    TMK_NONE            = 0,
    TMK_ALBEDO          = 1,
    TMK_SPECULAR        = 2,
    TMK_NORMAL          = 3,
    TMK_SMOOTHNESS      = 4,
    TMK_COVER           = 5,
    TMK_SSSMASK         = 6,
    TMK_COLORMASK       = 7,
    TMK_BLOOMMASK       = 8,
    TMK_HEIGHTMAP       = 9,
    TMK_DECALALBEDO     = 10,
    TMK_DECALNORMAL     = 11,
    TMK_DECALSPECULAR   = 12,
    TMK_LIGHTPROJECT    = 13,
    TMK_PARTICLE        = 14,
    TMK_DECALHEIGHTMAP  = 15,
    TMK_AO              = 16,
    TMK_UNUSED_3        = 17,
    TMK_UI              = 18,
    TMK_FONT            = 19,
    TMK_LEGACY_FLASH_UI = 20,
    TMK_UNUSED_4        = 21,
    TMK_BLENDMASK       = 22,
```

```
    TMK_PAINTEDDATAGRID = 23,
    TMK_COUNT           = 24,
};

enum textureFormat_t {
    FMT_NONE            = 0,
    FMT_RGBA32F         = 1,
    FMT_RGBA16F         = 2,
    FMT_RGBA8           = 3,
    FMT_RGBA8_SRGB      = 32,
    FMT_ARGB8           = 4,
    FMT_ALPHA           = 5,
    FMT_L8A8_DEPRECATED = 6,
    FMT_RG8             = 7,
    FMT_LUM8_DEPRECATED = 8,
    FMT_INT8_DEPRECATED = 9,
    FMT_BC1             = 10,
    FMT_BC1_SRGB        = 33,
    FMT_BC1_ZERO_ALPHA  = 54,
    FMT_BC3             = 11,
    FMT_BC3_SRGB        = 34,
    FMT_BC4             = 24,
    FMT_BC5             = 25,
    FMT_BC6H_UF16       = 22,
    FMT_BC6H_SF16       = 36,
    FMT_BC7             = 23,
    FMT_BC7_SRGB        = 35,
    FMT_DEPTH           = 12,
    FMT_DEPTH_STENCIL   = 13,
    FMT_DEPTH16         = 31,
    FMT_X32F            = 14,
    FMT_Y16F_X16F       = 15,
    FMT_X16             = 16,
    FMT_Y16_X16         = 17,
    FMT_RGB565          = 18,
    FMT_R8              = 19,
    FMT_R11FG11FB10F    = 20,
    FMT_R9G9B9E5        = 57,
    FMT_X16F            = 21,
    FMT_SMALLF          = 60,
    FMT_MAINVIEW_SMALLF = 61,
```

```
    FMT_RG16F           = 26,
    FMT_R10G10B10A2     = 27,
    FMT_RG32F           = 28,
    FMT_R32_UINT        = 29,
    FMT_RG32_UINT       = 58,
    FMT_R16_UINT        = 30,
    FMT_R8_UINT         = 55,
    FMT_ASTC_4X4        = 37,
    FMT_ASTC_4X4_SRGB   = 38,
    FMT_ASTC_5X4        = 39,
    FMT_ASTC_5X4_SRGB   = 40,
    FMT_ASTC_5X5        = 41,
    FMT_ASTC_5X5_SRGB   = 42,
    FMT_ASTC_6X5        = 43,
    FMT_ASTC_6X5_SRGB   = 44,
    FMT_ASTC_6X6        = 45,
    FMT_ASTC_6X6_SRGB   = 46,
    FMT_ASTC_8X5        = 47,
    FMT_ASTC_8X5_SRGB   = 48,
    FMT_ASTC_8X6        = 49,
    FMT_ASTC_8X6_SRGB   = 50,
    FMT_ASTC_8X8        = 51,
    FMT_ASTC_8X8_SRGB   = 52,
    FMT_DEPTH32F        = 53,
    FMT_RGBA16_UINT     = 56,
    FMT_RG16_UINT       = 62,
    FMT_RGBA16          = 59,
    FMT_NEXTAVAILABLE   = 63,
};
```

Revision #4
Created 3 June 2026 22:10:50 by FlavorfulGecko5
Updated 28 July 2026 12:12:12 by FlavorfulGecko5