Atlas Creation Tool
Using the Command-line Tool

The Atlas Creation Tool is a command-line tool that accepts a collection of arbitrary textures and packs them into one or more texture atlases. At the end of packing it generates a human-readable texture-atlas index (**`.tai`**) file that describes how each original texture maps to an atlas (for example, which atlas file contains the original texture and which sub-rectangle of that atlas it occupies). The **`.tai`** file thus allows transforming the original texture-coordinates to texture-coordinates that access the identical data from an atlas.

To compare the output of the Atlas Creation Tool—the generated atlases—to the original texture data, you can run the Atlas Comparison Viewer (refer to the *Atlas Comparison Viewer User Guide*.)

Using texture atlases instead of individual textures helps in reducing state changes (**`SetTexture()`** in particular), which reduces the number of **`Draw()`** calls per frame. Reducing state change and **`Draw()`** calls per frame ultimately reduces CPU-utilization caused by driver overhead. The ShaderX3 chapter *Improved Batching Via Texture Atlases* (also available as the NVIDIA white paper *Improved Batching Via Texture Atlases*) discusses the theory, benefits, and limitations of texture atlases in more detail.

Using the Atlas Creation Tool

The Texture Atlas Tool has the following command-line invocation syntax:

```
TextureAtlasTool.exe [-h –help -? -nomipmap –volume
-halftexel –width <w> –height <h> -depth <d> -o <filename>]
<img1> <img2> ...

   -h             prints this usage help
   -help          prints this usage help
   -?             prints this usage help
   -nomipmap      only writes out the top-level mipmap
   -volume        only valid w/ -nomipmap: make atlases
                  volume textures
   -halftexel     adds a half-texel offset to the
                  generated texture coordinates
   -width <w>     limits texture atlases to a maximum width
                  of w texels
   -height <h>    limits texture atlases to a maximum
                  height of h texels
   -depth <d>     limits texture atlases to a maximum depth
                  of d texels
   -o <filenamee> mandatory option that specifies output
                  filename
```

What this means is that all parameters are optional except for the output filename and a list of two or more images to pack into an atlas. The tool supports the following file formats: .bmp, **.dds**, **.dib**, **.jpg**, **.png**, and **.tga**. Textures can be any size, including non-square.

Typing the following on the command-line:

```
TextureAtlasTool.exe –o grass.png graydirt.png
```

Given two images **grass.png** and **graydirt.png** of the same format generates the output shown in Figure 1. It also generates two new files: **TexAtlas0.dds** and **TexAtlas.tai.**

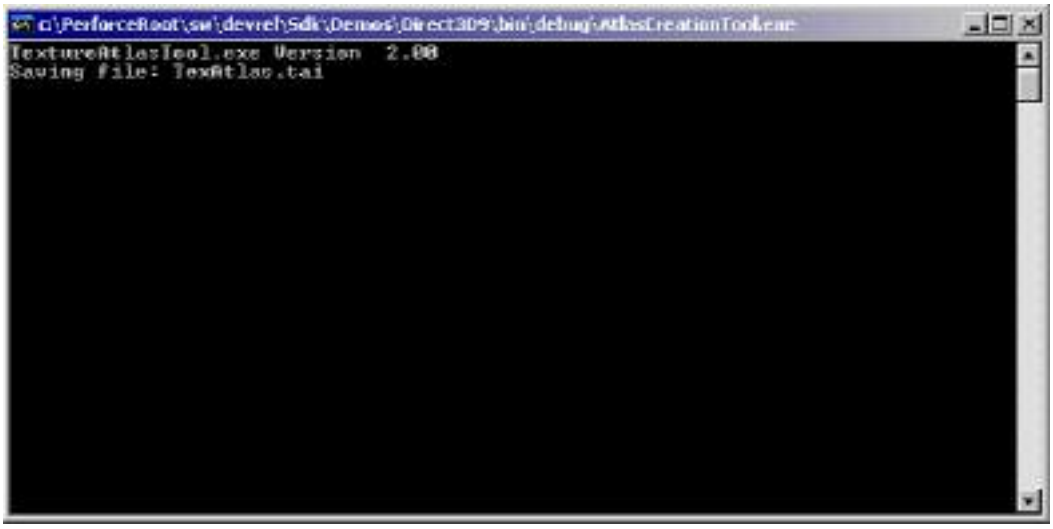


Figure 1. Command-line Output for Invoking Texture Atlas Tool with Minimal Parameters

TexAtlas0.dds is an image file containing **grass.png** and **graydirt.png** in the Direct3D format X8R8G8B8. The **TexAtlas.tai** file is human-readable (see Figure 2) and describes how the texture coordinates of models using these textures need to be modified to access the same data out of the generated atlas: For example, all models using the **grass.png** texture before require that their **u**-coordinate be divided by **2** and add **0.5**; to access the entire g**r**as**s.png** texture out of the **TexAtlas0.dds** the atlas texture-coordinates need to range from (**0.5, 0**) to (**1, 1**). Similarly, all models using the **graydirt.png** texture require that their u-coordinate be divided by **2**; to access the entire **graydirt.png** texture out of the **TexAtlas0.dds** the atlas texture-coordinates need to range from (**0, 0**) to (**0.5, 1**).

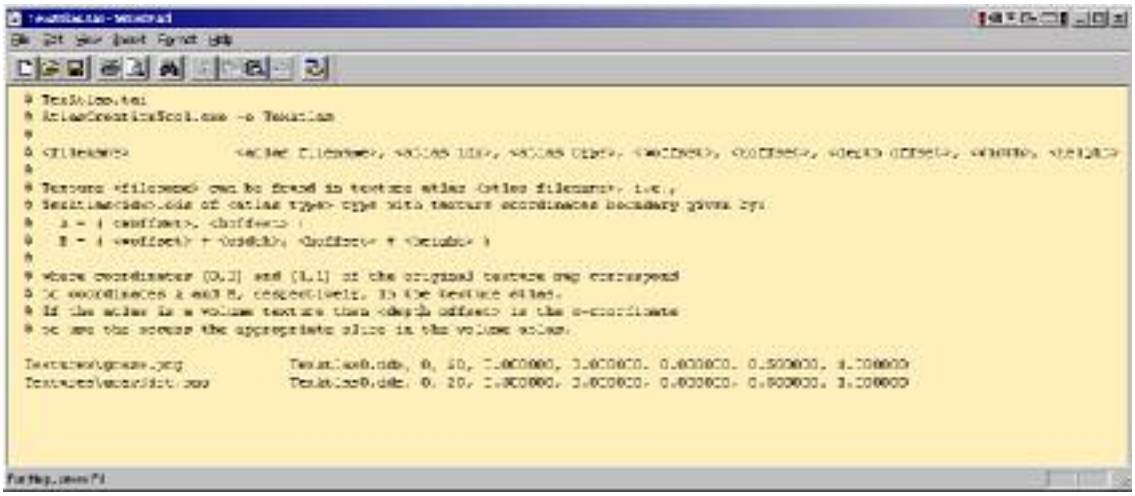
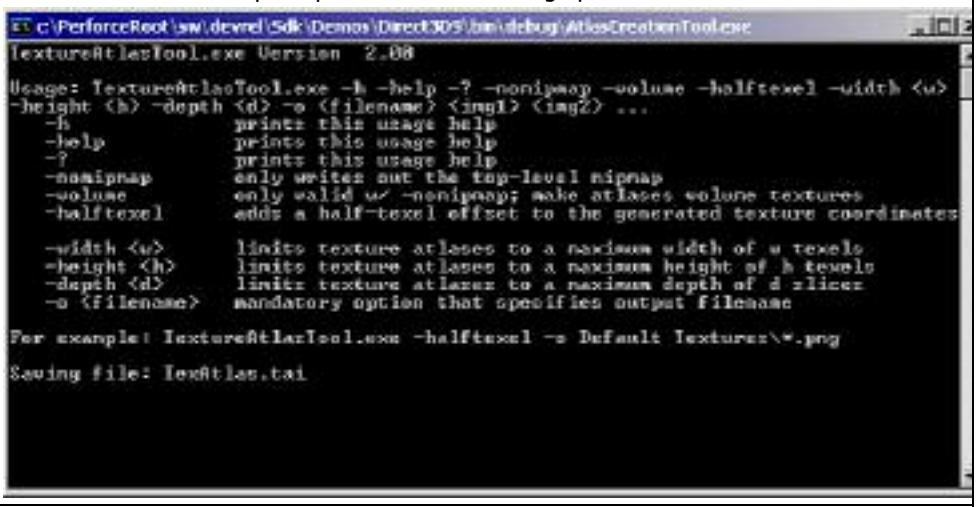


Figure 2. Generated TexAtlas.tai File

Command Line Options

Table 1 lists the commands used from the Atlas Creation tool.

Table 1. Command Line Options

Command	Descripription
-h **-help** **-?**	The –h command-line option prints the current usage pattern as shown: 
-nomipmap	The **-nomipmap** option instructs the application to ignore all mip-maps for all textures: as a result, all generated texture-atlases contain only the top-level texture, i.e., no mip-maps. This behavior is useful for user-interface or similar textures where it is guaranteed that none of the mip-map levels are ever accessed. In general, disabling mip-mapping is undesirable, as it causes prohibitive performance reductions. In contrast, not specifying **-nomipmap** loads and copies all of a texture's mip-maps into the respective atlas; if a texture does not contain any mip-maps or only a partial mip-map chain, then the Texture Atlas Tool relies on **D3DXCreateTextureFromFileEx()with D3DX_DEFAULT** parameters to generate the missing mip-maps. The resulting quality is thus questionable: using NVIDIA's texture tools (http://developer.nvidia.com/object/nv_texture_tools.html) to generate all mip-maps for all textures before passing them to the Texture Tool Atlas is preferred.
-volume	The **-volume** option specifies that all generated atlases are to be volume textures. This option requires that **-nomipmap** is also specified, since volume atlases cannot store the mipmaps of textures.
-halftexel	The **-halftexel** option offsets all generated texture coordinates by half a texel so as to ensure that only the texels of the original texture are accessed. Refer to the ShaderX^3 chapter for details.
-width <w>	The **-width** option limits the maximum width of all generated atlases to **w**.
-height <h>	The **-height** option limits the maximum height of all generated atlases to **h.**
-depth <d>	The **-depth** option limits the maximum depth of all generated volume atlases to **d**.
-o <filename>	The **-o** option specifies the name of the generated atlas and .tai files. For example, specifying –o Test will generate atlas files of the form **Test0.dds**, **Test1.dds**, etc, and a **Test.tai** file.

Typical Use Case

A graphics-engine programmer identifies a collection of textures that are responsible for a large number of **`DrawPrimitive()`** calls due to repeated calls to **`SetTexture()`**. These textures ideally are of or are converted to the same format to maximize the efficiency of packing them into atlases. If some of these textures do not contain complete mip-map chains, integrate NVIDIA's texture tools into the tool chain to generate and store these mip-maps together with the original textures. (Various file-formats, e.g., **`.dds`**, allow storing mip-map levels together with the original image.) Similarly, integrate the Atlas Creation Tool into the tool chain to convert these texture collections into collections of atlases. Create and integrate a separate tool that parses the generated .tai file to remap texture-coordinates of models to enable texturing out of the generated texture atlases.

Optionally, use the Atlas Comparison Viewer or a similar home-made tool able to display converted models to inspect the resulting image quality of texturing out of an atlas versus texturing from the original texture.

Known Bugs

Running with the DirectX 9 Summer 2003 DEBUG runtime and packing DXTn-format textures generates a Direct3D error and thus forces the application to exit. We are currently investigating the root cause for this error. Switching to the retail run-time works around that error.

Next Version Features

Following is a list of currently unimplemented features under consideration for the next version of this application.

- Add **`-cubemap`** option: only valid if at most six textures of the same format/dimension are supplied.
- Use the reference rasterizer as the hardware device: that way we get maximum capabilities, texture formats, and texture resolutions.
- Add a command-line option to force all generated atlases to a particular format.

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com