User Guide

nvDXTLib Compression / Decompression Library

November 2004

Introduction

The compression library accepts uncompressed data and writes out compressed MIP maps either to the function call *WriteDTXnFile* or an app supplied callback.

The formats supported are:

- RGBA – red, green, blue, aplah. 8 bits per color channel. 4 color channels
- RGB – red, green, blue. 8 bits per color channel. 3 color channels
- BGRA – blue, green, red, alpha. 8 bits per color channel, 4 color channels
- BGR – blue, green, red. 8 bits per color channel. 3 color channels
- RGBAImage structure. Defined in tPixel.h
- fpImage structure tPixel.h. 32 bit per color channel, floating point

See nvDXT.cpp for example for calling example.

```
nvDXTcompressRGBA, nvDXTcompressBGRA - Image compression.

Pass unsigned char * parameter in RGBA or BGRA order.

plane == 3 indicates no alpha is present.

nvDXTcompressVolumeRGBA, nvDXTcompressVolumeBGRA - volume texture
creation

nvDXTcompress32F - floating point input

nvDXTcompress - RGBAImage struct input

HRESULT nvDXTcompressRGBA(unsigned char * src_data, // pointer to data (24 or
32 bit)
      unsigned long w, // width in texels
      unsigned long h, // height in texels
      DWORD byte_pitch,
      CompressionOptions * options,
      DWORD planes, // 3 or 4 color channels
      MIPcallback callback = NULL,  // callback for generated levels
      RECT * rect = NULL);   // subrect to operate on, NULL is whole image
```

```
// define color order
HRESULT nvDXTcompressBGRA(unsigned char * src_data,
      unsigned long w, // width in texels
      unsigned long h, // height in texels
      DWORD byte_pitch,
      CompressionOptions * options,
      DWORD planes, // 3 or 4 color channels
      MIPcallback callback = NULL,  // callback for generated levels
      RECT * rect = NULL);



HRESULT nvDXTcompressVolumeRGBA(unsigned char * src_data,
      unsigned long w, // width in texels
      unsigned long h, // height in texels
      unsigned long depth, // depth of volume texture
      DWORD byte_pitch,
      CompressionOptions * options,
      DWORD planes, // 3 or 4
      MIPcallback callback = NULL,  // callback for generated levels
      RECT * rect = NULL);   // subrect to operate on, NULL is whole image

HRESULT nvDXTcompressVolumeBGRA(unsigned char * src_data,
      unsigned long w, // width in texels
      unsigned long h, // height in texels
      unsigned long depth, // depth of volume texture
      DWORD byte_pitch,
      CompressionOptions * options,
      DWORD planes, // 3 or 4
      MIPcallback callback = NULL,  // callback for generated levels
      RECT * rect = NULL);   // subrect to operate on, NULL is whole image


// floating point input
HRESULT nvDXTcompress32F(fpImage & srcImage,
      CompressionOptions * options,
      MIPcallback callback = NULL,  // callback for generated levels
      RECT * rect = NULL);   // subrect to operate on, NULL is whole image


HRESULT nvDXTcompress(RGBAImage & image,
      CompressionOptions * options,
      MIPcallback callback,
      RECT * rect);
```

If callback is == 0 (or not specified), then WriteDTXnFile is called with all file info instead of your callback

```
typedef HRESULT (*MIPcallback)(
void * data, // pointer to the data to compressed data
int miplevel, // what MIP level this is
DWORD size, // size of the data
int width,   // width of MIP map
int height,   // height of MIP map
void * user_data);  // user pointer

   // You must write the routines (or provide stubs) for
   WriteDTXnFile and ReadDTXnFile

   void WriteDTXnFile(DWORD count, void * buffer, void * userData);

   void ReadDTXnFile(DWORD count, void * buffer, void * userData);
```

See the file nvdxt_options.h for the definition of CompressionOptions

```
    // error return codes
  #define DXTERR_INPUT_POINTER_ZERO -1
  #define DXTERR_DEPTH_IS_NOT_3_OR_4 -2
  #define DXTERR_NON_POWER_2 -3
```

Example callback to store compressed image in a Direct3D texture

```
HRESULT LoadAllMipSurfaces(void * data, int iLevel, DWORD size,
                 int Width, int Height, void * user)
{
    HRESULT hr;
    LPDIRECT3DSURFACE9 psurf;
    D3DSURFACE_DESC sd;
    D3DLOCKED_RECT lr;

    hr = pCurrentTexture->GetSurfaceLevel(iLevel, &psurf);

    if (FAILED(hr))
        return hr;
    psurf->GetDesc(&sd);

    hr = pCurrentTexture->LockRect(iLevel, &lr, NULL, 0);

    if (FAILED(hr)) return hr;

    memcpy(lr.pBits, data, size);

    current_size += size;

    hr = pCurrentTexture->UnlockRect(iLevel);

    ReleasePpo(&psurf);

    mips_completed++;

    if(g_d3d) {
        g_d3d->Render3DEnvironment();
    }

    return 0;
}
```

You link to different libraries depending on your compile options.
There are pragma that should link automatically to the correct library.

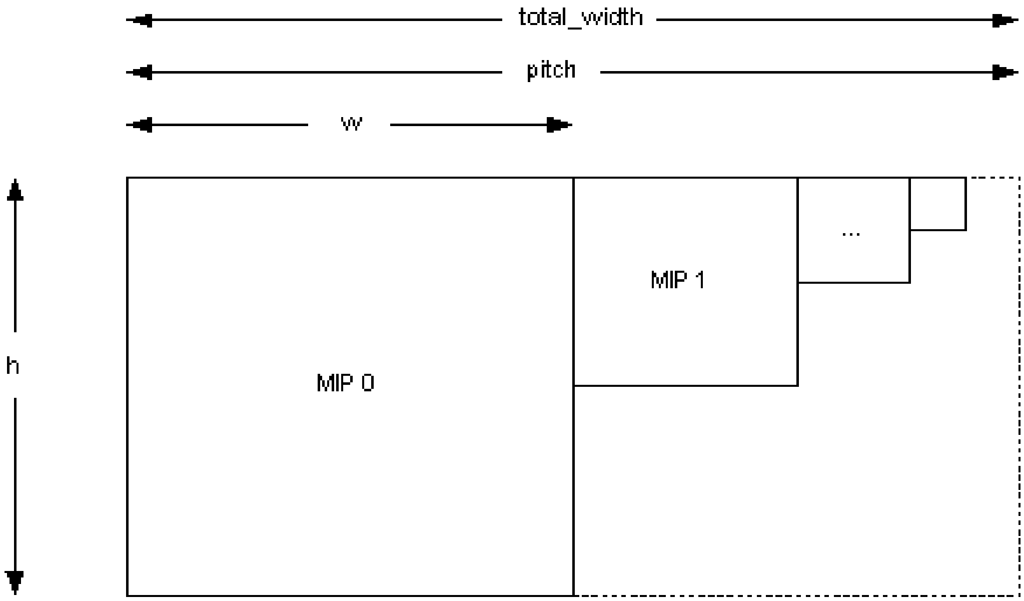
nvDXTLib.lib - release

nvDXTLibMT.lib – release multi-threaded

nvDXTLibMTDLL.lib – release multi-threaded dll

The _S options is used when _STATIC_CPPLIB is defined.

If you have existing MIP maps you must combine them so each MIP level is followed by its next MIP level. Conceptually, it looks like this:



Compression Options

CompressionOptions is the structure where you pass the compression options to the compressor. See *nvdxt_options.h* for details about this structure.

```
MipMapType = dUseExistingMipMaps;
```

You specify how map MIP levels to write out

```
nvDXTcompress((unsigned char *)raw_data, width, height, pitch,
&options, depth, 0);
```

Decompression

To decompress an image use the *nvDXTdecompress* call to read all MIP chains into one buffer:

```
unsigned char * nvDXTdecompress(int & w, int & h, int & depth,
    int & total_width, int & rowBytes, int & src_format,
    int SpecifiedMipMaps = 0);
```

returns
pointer to image data
w : image width
h : image height
depth : number of bytes per pixel, 3 or 4

row_bytes: pitch of main image
The first image starts at 0, the next MIP map image starts at base + row_bytes, next one starts at base + row_bytes / 2, etc.

src_format: format of the file
SpecifiedMipMaps. Load in only this number of MIP maps. zero means read all MIP levels

pitch = row_bytes * 2

see readdxt.cpp for example

Trademarks

Copyright

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