IWI v.8

Byte #1	Byte #2	Byte #3	Byte #4	Byte #5	Byte #6	Byte #7	Byte #8
Magic			Version	*Unknown 1*	*Unknown 2*	Usage	*Unknown 3*
Format	*Unknown 4*	Width		Height		*Unknown 5*	*Unknown 6*
File size				Offset of the texture			
Offset of the 1st mipmap				Offset of the 2nd mipmap			

Magic

File magic number, uniquely identifies the file as IWI. Must contain this value:

`0x49 0x57 0x69`

„IWi“ in ASCII

Version

Identifies the version of the file, for IWI v.8 this field must contain this value:

`0x08`

Usage

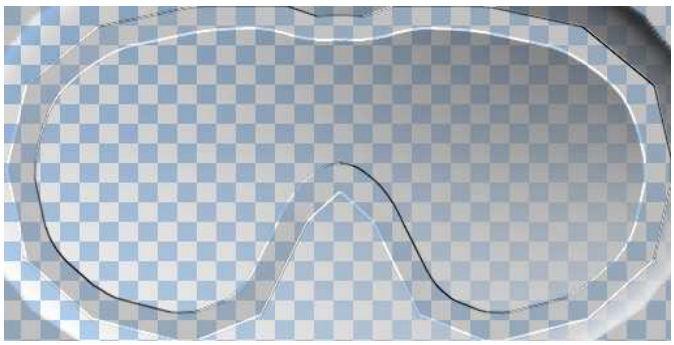
Describes the usage of the texture, usually color image (files have „col“ in their name as „color“), normal texture (files have „nml“ in their name as „normal“) or skybox collection. Possible values: [1)]

`0x00` – Color image
`0x01` – Skybox collection
`0x04` – Additive image (normal map)
`0x09` – Skybox collection (images are usually with alpha channel)

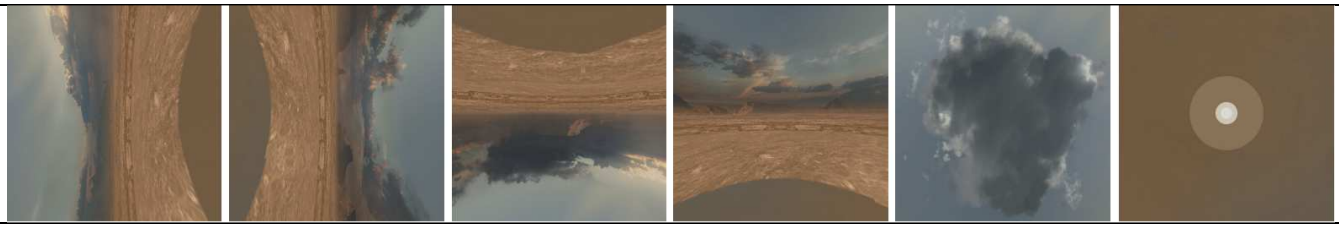
Examples:

`0x00 (Color image)`
MW2: iw_00.iwd/airborne_alpha_a_col.iwi

`0x04 (Additive image – normal map)`
MW2: iw_00.iwd/airborne_alpha_a_nml.iwi



`0x01 (Skybox collection)`
MW2: iw_04.iwd/ sky_mp_boneyard_ft.iwi



Format

Format in which are the pixels stored in the file. IWI supports several pixel formats, both compressed (DXT) and uncompressed. Table of all known formats: [1)]

Value	Name	Description
0x01	ARGB 32bit	Pixels are stored in this order: ARGB, 8 bits per color
0x02	RGB 24bit	Pixels are stored in this order: RGB, 8 bits per color
0x03	GrayScale + Alpha 16bit	Pixels are stored in this order: GA (G = luminosity, 0x00 is black, 0xFF is white), 8 bit per channel
0x04	Alpha 8bit	Alpha channel only, 8 bits
0x0B	DXT1	Compressed, 16 input pixels (block 4x4 px) are compressed into 64 bits of output, alpha resolution is 1 bit (opaque / transparent)
0x0C	DXT3	Compressed, 16 input pixels (block 4x4 px) are compressed into 128 bits of output, sharp alpha (explicit)
0x0D	DXT5	Compressed, 16 input pixels (block 4x4 px) are compressed into 128 bits of output, gradient alpha (interpolated)

Examples on how to decode DXT image into GDI+ bitmap (this example uses squish to decode DXT compressed data into the 4x4 block of pixels):

```
#define MAKE_DIM_DXT(X) if (X % 4 != 0) X = ((X/4)+1)*4;
#define DXT_GET_BLOCK_DIM(X,IT) ((X%4) != 0 ? (IT*4 < X - X%4 ? 4 : X%4) : 4)

bool _decodeImageDXT1(HANDLE File, unsigned long Offset, unsigned int Width,
   unsigned int Height, Gdiplus::BitmapData * BitmapData)
{
  SetFilePointer(File, Offset, 0, FILE_BEGIN);
  UINT * _pixels = (UINT*)(BitmapData->Scan0);
  SIZE szTex,szOrig;
  szTex.cx = szOrig.cx = Width;
  szTex.cy = szOrig.cy = Height;
  DWORD read;
  MAKE_DIM_DXT(szTex.cx)
  MAKE_DIM_DXT(szTex.cy)
  squish::u8 block[8];
  squish::u8 pixels[16*4];
  for (int brow = 0; brow < szTex.cy / 4; brow++)
  {
    for (int bcol = 0; bcol < szTex.cx / 4; bcol++)
    {
      ReadFile(File, &block, 8, &read, 0);
      squish::Decompress(pixels,block,squish::kDxt1);
      for (int a = 0; a < DXT_GET_BLOCK_DIM(szOrig.cy, brow); a++)
      {
        for (int b = 0; b < DXT_GET_BLOCK_DIM(szOrig.cx, bcol); b++)
          _pixels[(brow*4+a) * BitmapData->Stride / 4 + bcol*4+b] =
            (pixels[a * 16 + b * 4]     << 16) |
            (pixels[a * 16 + b * 4 + 1] << 8 ) |
            (pixels[a * 16 + b * 4 + 2]      ) |
            (pixels[a * 16 + b * 4 + 3] << 24);
      }
    }
  }
  return true;
}
```

Width

Height

Width and height of the texture, value is stored as uint16_t:

```
0x80 0x00 = 0x0080 = 128
0x00 0x04 = 0x0400 = 1024
```

File size

Total size of the file, value is stored as uint32_t:

```
0x90 0x55 0x15 0x00 = 0x00155590 = 1398160
```

Offset of the texture

This value is the offset of the main image. Note that IWI stores mipmaps first (from the smallest), then the main image.

When the Usage field is set to 0x01 or 0x09 [1] (Skybox) or the image isn't mipmapped, this field contains the total size of the file.

Offset of the 1st mipmap

This value is the offset of the first mipmap image [2].

If the main image isn't mipmapped, this value is the same as „Offset of the texture"

When the Usage field is set to 0x01 or 0x09 [1] (Skybox), this field contains the total size of the file.

Offset of the 2nd mipmap

This value is the offset of the second mipmap image [2].

If the main image isn't mipmapped, this value is the same as „Offset of the texture"

When the Usage field is set to 0x01 or 0x09 [1] (Skybox), this field contains the total size of the file.

Images

Usage field set to 0x00 or 0x04 [1]:

If the texture is mipmapped („Offset of the texture" != „Offset of the 1st mipmap"), the mipmaps are stored right after the end of the header starting with the smallest one (1x1 px).

The original image (original dimensions) is stored after the mipmaps.

Usage field set to 0x01 or 0x09 [1]:

6 images are stored right after the end of the header, starting with 4 side images, 1 top image and 1 bottom image, creating a skybox texture. Every image has the same dimensions, therefore the size of the single image in bytes is always the same (applies to the listed storage formats (rgb images or fixed compression ratio (DXT)

[1] – not all possible types are listed or some are unknown

[2] – I am not sure what are these values if the main image is for example 2x2, the first mipmap is 1x1, the second mipmap doesn't exist. Value of the 2nd mipmap offset is unknown for me, however IWIViewer will set the value of the 2nd mipmap offset to zero.

Matej Tomčík
matej_tomcik@msn.com