

# Re: Compressed textures

---

*Source:*

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.directx.graphics/2008-02/ms>

---

- *From:* "John Withe" <a@xxxx>
  - *Date:* Mon, 25 Feb 2008 21:04:44 +0100
- 

If your textures are dynamic and can't be compressed ahead of time, then load the texture into a plain surface in system memory and use `D3DXLoadSurfaceFromSurface` to transfer it to a compressed surface. The target surface can be wherever you like — D3DX will handle the different situations accordingly. There are other compression libraries from NVidia that you can use in place of D3DX, but D3DX is already there so you might as well use it.

This is only worthwhile if you infrequently load the texture data and frequently draw it. Otherwise the overhead of compression is too high.

I ship the application and the end user uses it to display user defined scenes with user defined movies in it, so I need the application to be able to read an arbitrary movie and use its frames as textures. The actual displaying needs to be fast, but if the conversion can be done ahead of time, thus allowing a fast load, then it is fine.

Why do you need to read out the binary data?

I do not, but if I don't then the conversion will need to happen every time the movie is loaded. I would much rather have a slow conversion and a subsequent fast loading and displaying of the graphics. Some movies might be too big to be kept in memory all at once. I see no other option than to precompress them and then runtime do a fast load and displaying of them. In essence I believe I need to deserialize the textures (one texture per frame) quickly runtime.

Hm. Perhaps things are still unclear.

This is an application used in videobroadcasting with live shows. Artists create video data which need to be displayed runtime in a 3d scene. Depending on the situation one or another video may be displayed here or there.

Currently I do two different things. When a movie is small enough, I read it into system memory as a binary stream defining the texels of different frames. I then copy different frames into a texture and use the texture as a surface for rendering the video in the scene. It is fast enough, but I would certainly like it to be faster. Whenever I need a new video frame in the texture, I lock the texture, write the bytes and unlock it.

## Re: Compressed textures

For larger movies I runtime render a frame into system memory and do the same copying. It is also fast enough, but somewhat slower than the version in system memory.

The real bottleneck seems to be the copying of texels each frame. If I use compressed textures, then I will transfer less data and things will be faster... or so I hope.

If the video file needs to be preprocessed slowly so it can later load quickly then it is quite alright.

I must admit that the dx part of this system is the one where I have the problems. I am still learning that area, so I may be doing something quite silly. A nvidia quadro fx5500 can render a 720\*576\*ARGB video with around 1000fps from memory and 800fps when streaming directly from disk. Using nvidia core 2 duo 2GHz