Desktop Window Manager

Desktop Window Manager (**DWM**, previously **Desktop Compositing Engine** or **DCE** in builds of pre-reset Windows Longhorn) is the compositing window manager in Microsoft Windows since Windows Vista that enables the use of hardware acceleration to render the graphical user interface of Windows.

Desktop Window Manager	
Developer	Microsoft
Initial release	November 30, 2006
Operating system	Microsoft Windows
Service name	UxSms[note 1]

It was originally created to enable portions of the new "Windows Aero" user experience, which allowed for effects such as transparency, 3D window switching and more. It is also included with Windows Server 2008, but requires the "Desktop Experience" feature and compatible graphics drivers to be installed.[1]

Architecture

The Desktop Window Manager is a compositing window manager, meaning that each program has a buffer that it writes data to; DWM then composites each program's buffer into a final image. By comparison, the stacking window manager in Windows XP and earlier (and also Windows Vista and Windows 7 with Windows Aero disabled) comprises a single display buffer to which all programs write.

DWM works in different ways depending on the operating system (Windows 7 or Windows Vista) and on the version of the graphics drivers it uses (WDDM 1.0 or 1.1). Under Windows 7 and with WDDM 1.1 drivers, DWM only writes the program's buffer to the video RAM, even if it is a graphics device interface (GDI) program. This is because Windows 7 supports (limited) hardware acceleration for GDI[2] and in doing so does not need to keep a copy of the buffer in system RAM so that the CPU can write to it.

Because the compositor has access to the graphics of all applications, it easily allows visual effects that string together visuals from multiple applications, such as transparency. DWM uses DirectX to perform the function of compositing and rendering in the GPU, freeing the CPU of the task of managing the rendering from the off-screen buffers to the display. However, it does not affect applications painting to the off-screen buffers – depending on the technologies used for that, this might still be CPU-bound. DWM-agnostic rendering techniques like GDI are redirected to the buffers by rendering the user interface (UI) as bitmaps. DWM-aware rendering technologies like WPF directly make the internal data structures available in a DWM-compatible format. The window contents in the buffers are then converted to DirectX textures.

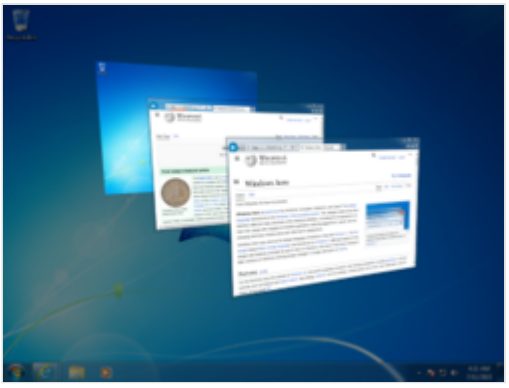
The desktop itself is a full-screen Direct3D surface, with windows being represented as a mesh consisting of two adjacent (and mutually inverted) triangles, which are transformed to represent a 2D rectangle. The texture, representing the UI chrome, is then mapped onto these rectangles. Window transitions are implemented as transformations of the meshes, using shader programs.[3] With Windows Vista, the transitions are limited to the set of built-in shaders that implement the transformations. Greg Schechter, a developer at Microsoft has suggested that this might be opened up for developers and users to plug in

their own effects in a future release.[4] DWM only maps the primary desktop object as a 3D surface; other desktop objects, including virtual desktops as well as the *secure desktop* used by User Account Control are not.[5]

Because all applications render to an off-screen buffer, they can be read off the buffer embedded in other applications as well. Since the off-screen buffer is constantly updated by the application, the embedded rendering will be a dynamic representation of the application window and not a static rendering. This is how the live thumbnail previews and Windows Flip work in Windows Vista and Windows 7. DWM exposes a public API that allows applications to access these thumbnail representations.[6] The size of the thumbnail is not fixed; applications can request the thumbnails at any size - smaller than the original window, at the same size or even larger - and DWM will scale them properly before returning. Aero Flip does not use the public thumbnail APIs as they do not allow for directly accessing the Direct3D textures.[7] Instead, Aero Flip is implemented directly in the DWM engine.

Aero Flip demonstrates multiple features of DWM: 3D transformation of 2D planes, scaling and translating the planes to a different position, embedding dynamic views of one application in another and use of custom shader programs.

The Desktop Window Manager uses Media Integration Layer (MIL), the unmanaged compositor which it shares with Windows Presentation Foundation, to represent the windows as *composition nodes* in a *composition tree*. The composition tree represents the desktop and all the windows hosted in it, which are then rendered by MIL from the back of the scene to the front.[8] Since all the windows contribute to the final image, the color of a resultant pixel can be decided by more than one window. This is used to implement effects such as per-pixel transparency. DWM allows custom shaders to be invoked to control how pixels from multiple applications are used to create the displayed pixel. The DWM includes built-in Pixel Shader 2.0 programs which compute the color of a pixel in a window by averaging the color of the pixel as determined by the window behind it and its neighboring pixels. These shaders are used by DWM to achieve the blur effect in the window borders of windows managed by DWM, and optionally for the areas where it is requested by the application.[3]



Aero Flip feature being used in Windows 7

Since MIL provides a retained mode graphics system by caching the composition trees, the job of repainting and refreshing the screen when windows are moved is handled by DWM and MIL, freeing the application of the responsibility. The background data is already in the composition tree and the off-screen buffers and is directly used to render the background. In pre-Vista Windows OSs, background applications had to be requested to re-render themselves by sending them the `WM_PAINT` message.[6] DWM uses double-buffered graphics to prevent flickering and tearing when moving windows.[3][6] The compositing engine uses optimizations such as culling to improve performance, as well as not redrawing areas that have not changed.[8] Because the compositor is multi-monitor aware, DWM natively supports this too.[8]

During full-screen applications, such as games, DWM does not perform window compositing and therefore performance will not appreciably decrease.

On Windows 8 and Windows Server 2012, DWM is used at all times and cannot be disabled, due to the new "start screen experience" implemented. Since the DWM process is usually required to run at all times on Windows 8, users experiencing an issue with the process are seeing memory usage decrease after a system reboot. This is often the first step in a long list of troubleshooting tasks that can help. It is possible to prevent DWM from restarting temporarily in Windows 8, which causes the desktop to turn black, the taskbar grey, and break the start screen/modern apps, but desktop apps will continue to function and appear just like Windows 7 and Vista's Basic theme, based on the single-buffer renderer used by XP. They also use Windows 8's centered title bar, visible within Windows PreInstallation Environment. Starting up Windows without DWM will not work because the default lock screen requires DWM unlike the fallback lockscreen that appears as a command line interface program when Windows.UI.Logon.dll isn't present on Windows versions such as 1507 and later, so it can only be done on the fly, and does not have any practical purposes. Starting with Windows 10, disabling DWM in such a way will cause the entire compositing engine to break, even traditional desktop apps, due to Universal App implementations in the taskbar and new start menu. Windows can still be partially usable without the presence of DWM but requires Sihost.exe to not be present due to it relying on DWM. Most of the applications in Windows 11 require DWM to render UI elements and transparency, Windows 11's new task manager requires dwm to render menus unlike the fallback -d version. Unlike its predecessors, Windows 8 supports basic display adapters through Windows Advanced Rasterization Platform (WARP), which uses software rendering and the CPU to render the interface rather than the graphics card. This allows DWM to function without compatible drivers, but not at the same level of performance as with a normal graphics card. DWM on Windows 8 also adds support for stereoscopic 3D.[9]

Redirection

For rendering techniques that are not DWM-aware, output must be redirected to the DWM buffers. With Windows, either GDI or DirectX can be used for rendering. To make these two work with DWM, redirection techniques for both are provided.

With GDI, which is the most used UI rendering technique in Microsoft Windows, each application window is notified when it or a part of it comes in view and it is the job of the application to render itself. Without DWM, the rendering rasterizes the UI in a buffer in video memory, from where it is rendered to the screen. Under DWM, GDI calls are redirected to use the Canonical Display Driver (cdd.dll), a software renderer.[10] A buffer equal to the size of the window is allocated in system memory and CDD.DLL outputs to this buffer rather than the video memory. Another buffer is allocated in the video memory to represent the DirectX surface, which is used as the texture for the window meshes. The system memory buffer is converted to the DirectX surface separately and kept in sync. This round-about route is required because GDI cannot output directly in DirectX pixel format. The surface is read by the compositor and is composited to the desktop in video memory. Writing the output of GDI to system memory is not hardware accelerated, nor is conversion to the DirectX surface. When a GDI window is minimized, invisible, or visible on the same monitor as a full-screen DirectX application, by limitation of GDI, the GDI bitmap buffer is no longer received by the application when requesting a device context

during painting or updating (this can sometimes be seen when a GDI operation copying from one window to another output black or empty regions instead of the expected window content). Thus, DWM uses the last bitmap rendered to the buffer before the application was minimized.[11]

Starting from Windows 7, Canonical Display Driver no longer renders to the system memory copy when a WDDM 1.1/DXGI 1.1 compliant video driver is present.

For applications using DirectX to write to a 3D surface, the DirectX implementation in Windows Vista uses WDDM to share the surface with DWM. DWM then uses the surface directly and maps it onto the window meshes. For Windows presentation foundation (WPF) applications (which are DirectX applications), the compositor renders to such shared surfaces which are then composited into the final desktop.[11] Applications can mix either rendering techniques across multiple child windows, as long as both GDI and DirectX are not used to render the same window. In that case, the ordering between DirectX and GDI rendering cannot be guaranteed, and as such it cannot be guaranteed that the GDI bitmap from the system memory has been translated to the video memory surface. This means that the final composition may not contain the GDI-rendered elements.[11] To prevent this, DWM is temporarily turned off, as long as an application that mixes GDI and DirectX in the same window is running.

Hardware requirements

In Windows Vista, DWM requires compatible physical or virtual hardware:[12]

- A GPU that supports the Windows Display Driver Model (WDDM)
- Direct3D 9 support
- Pixel Shader 2.0 support
- Support for 32 bits per pixel
- Passes the Windows Aero acceptance test in the Windows Driver Kit (WDK)

In Windows 7, the Desktop Window Manager has been reworked to use Direct3D 10.1, but the hardware requirements remain the same as in Windows Vista; Direct3D 9 hardware is supported with the "10 Level 9" layer introduced in the Direct3D 11 runtime. Windows 8 has the same requirements as 7, but it can also use software rendering when compatible video hardware is absent.[9]

Hardware virtualization software that emulate hardware required for DWM include VirtualBox 4.1 and later, VMware Fusion 3.0 and later, and VMware Workstation 7.0 onwards. In addition, Windows Virtual PC allows composition using the Remote Desktop Protocol.

Developer experience

Developer functionality related to the Desktop Window Manager is provided within the header file `dwmapi.h` within the Windows SDK.

See also

- Compiz

- Desktop environment
- Mutter
- Quartz Compositor

Notes

1. As of Windows 8, where DWM is a required component that must be running at all times, it is no longer a service.

References

1. "How to enable Windows Vista user experience features on a computer that is running Windows Server 2008 (MSKB947036)" (http://support.microsoft.com/kb/947036/en-us). *Knowledge Base*. Microsoft. January 15, 2008. Retrieved 2008-04-21.
2. "Engineering Windows 7" (http://blogs.msdn.com/e7/archive/2009/04/25/engineering-windows-7-for-graphics-performance.aspx). 25 September 2024.
3. Greg Schechter. "DWM's use of DirectX, GPU, and hardware acceleration" (http://blogs.msdn.com/greg_schechter/archive/2006/03/19/555087.aspx). *Greg Schechter's Blog*. MSDN Blogs. Retrieved 2007-10-14.
4. Greg Schechter. "Responding to Comments from "DWM's use of DirectX, GPU and hardware acceleration" " (http://blogs.msdn.com/greg_schechter/archive/2006/03/25/561110.aspx). *Greg Schechter's Blog*. MSDN Blogs. Retrieved 2008-04-20.
5. Chris Jackson. "Desktop Window Manager only runs on the primary desktop" (http://blogs.msdn.com/cjacks/archive/2006/11/09/a-desktop-of-your-own.aspx). *Chris Jackson's Semantic Consonance*. MSDN Blogs. Retrieved 2007-10-14.
6. Greg Schechter (6 March 2006). "Under the hood of Desktop Window Manager" (https://docs.microsoft.com/en-us/archive/blogs/greg_schechter/under-the-hood-of-the-desktop-window-manager). *Greg Schechter's Blog*. MSDN Blogs. Retrieved 2021-05-27.
7. "DWM Thumbnail Overview" (https://archive.today/20120717154553/http://msdn.microsoft.com/en-us/library/aa969541.aspx). *MSDN*. Archived from the original (http://msdn2.microsoft.com/en-us/library/aa969541.aspx) on 17 July 2012.
8. Greg Schechter. "How underlying WPF concepts and technology are being used in the DWM" (http://blogs.msdn.com/greg_schechter/archive/2006/06/09/623566.aspx). *Greg Schechter's Blog*. MSDN Blogs. Retrieved 2007-10-14.
9. "Desktop Window Manager is always on" (http://msdn.microsoft.com/en-us/library/windows/desktop/hh848042%28v=vs.85%29.aspx). *Windows 8 and Windows Server 2012 Compatibility Cookbook*. MSDN. Retrieved 4 September 2012.
10. "Comparing Direct2D and GDI - DirectX Developer Blog" (https://web.archive.org/web/20140408062122/http://blogs.msdn.com/b/directx/archive/2009/09/29/comparing-direct2d-and-gdi.aspx). Archived from the original (http://blogs.msdn.com/b/directx/archive/2009/09/29/comparing-direct2d-and-gdi.aspx) on 2014-04-08. Retrieved 2014-08-19.
11. Greg Schechter. "Redirecting GDI, DirectX, and WPF applications" (https://web.archive.org/web/20100305172826/http://blogs.msdn.com/greg_schechter/archive/2006/05/02/588934.aspx). Archived from the original (http://blogs.msdn.com/greg_schechter/archive/2006/05/02/588934.aspx) on 2010-03-05. Retrieved 2007-10-14.
12. "System requirements for Windows Vista" (http://support.microsoft.com/kb/919183). Microsoft. 2007-11-13. Retrieved 2009-02-11.

External links

- Desktop Window Manager (http://msdn.microsoft.com/en-us/library/aa969540.aspx)
- APIs in the Desktop Window Manager (http://blogs.msdn.com/greg_schechter/archive/2006/09/14/753605.aspx)
- Using the DWM APIs (http://weblogs.asp.net/kennykerr/archive/2006/08/10/Windows-Vista-for-Developers-_1320_-Part-3-_1320_-The-Desktop-Window-Manager.aspx)
- What is DWM.exe is it a virus? (http://www.groovypost.com/howto/geek-stuff/what-is-dwm-exe-virus-should-it-be-running)