- Application Notes
- White Papers
- Benchmarks
- AGP Building Blocks
- AGP Tutorial
- AGP Demonstration
- AGP Specification, Revision 1.0
- AGP Implementors Forum

Chapter 6: What This Means for Software Developers

So what should an application software developer do about AGP? There ar two possibilities: 1) Do nothing, or 2) Optimize for AGP. For both cases, tl big benefit of AGP is more and larger textures for 3D graphics realism witl loss of real-time performance. Today's applications usually must limit themselves to less than 2 MBytes of textures at any time with graphics hardware controllers. AGP will change that, assuming the application inclu the scalability of high-end texture content. Furthermore, any existing applications as well as new applications written without special efforts for . will run faster on AGP systems.

True AGP-compliant hardware can actually make applications simpler. Bu hardware with AGP will come in three flavors, and software will probably to support all three:

- Type 1: This hardware has an AGP interface, but does not exploit its texturing features. It just transfers data faster than a PCI device could probably does not exploit the pipelining capability or sideband addressing.

- Type 2: This hardware renders textures from AGP memory, thus the application does not need to swap textures into local memory. The hardware may or may not be able to texture from local memory also. may perform faster when not texturing from local memory, due to conflicts for access to local memory for pixel writes, screen refresh, t reads, and Z-values.

- Type 3: This hardware runs best when concurrently exploiting both lo memory and AGP memory for texturing. Frequently-used textures or smaller textures would best reside in local memory, while larger less-frequently used textures should reside in system memory. Thus t bandwidth drain on main memory is minimized, reducing conflicts between the CPU and graphics controller.

DOS Applications

Of course direct memory execution of textures requires the GART, becaus the virtual addressing scheme used in today's operating systems. But for applications running under yesterday's operating systems (e.g., DOS) with virtual addressing, the GART serves no purpose. Old applications running under DOS will see the benefit of faster AGP speed, but will require some driver work to turn on the graphics controller's ability to directly access textures in system memory .

Windows Applications*

Unmodified Windows* applications can benefit from AGP, because the OS

DirectDraw* have changed slightly to support it by default. For details see
Microsoft Web site on AGP.

For current hardware implementations, the OS will make AGP memory (lil other video memory) non-cacheable, so that there is no coherency problem between the CPU caches and the data that the graphics controller uses. Otherwise, graphics controller accesses to AGP memory would require "snooping" the CPU caches, which would cause delays in execution in som cases. CPU reads from uncached memory are slow, so algorithms should a CPU reads from AGP main memory as well as from graphics local memor

Note that in Pentium II processor-based systems, this non-cached graphics memory will be marked by the OS as "Write Combining" (WC), which get significantly faster CPU write-access than straight "Uncacheable" (UC). W memory areas let the CPU "combine" multiple discrete writes into a burst-v on the memory bus when the bus is available, using dedicated write-buffer built-in to the chip. Except for the faster speed, WC should remain transpar to applications. While the CPU read-access speed is no faster for WC than the use of UC memory will cause Pentium II processors to serialize executi which will probably slow the execution significantly. The fact that multiple writes can get combined together before getting outside the CPU can have impact on hardware device drivers, which may depend on multiple sequent writes to the same location, and "strong ordering" of memory writes.

Default DirectDraw Memory Allocations

Unless the application specifically requests otherwise, Microsoft DirectDra will by default allocate memory for textures in the following order:

- Local graphics controller memory.
- AGP main memory.
- System memory.

What if the graphics controller cannot texture from AGP memory? Well, ir situation DirectDraw can be prevented from allocating any non-local video memory for texturing. The graphics controller driver reports its capabilities the OS and DirectDraw, and if the graphics controller cannot directly acces system memory, then DirectDraw will allocate only local video memory ar system memory to the application. Similarly, if the graphics chip cannot te from local video memory, DirectDraw will not allocate any textures locally

If it could not fit all textures into the AGP memory which DirectDraw agre allocate, then the application must eventually copy some more textures fro disk into AGP memory. Very realistic flight simulators or other application using large amounts of textures may need to stream textures from disk or network into AGP memory, no matter how much memory DirectDraw gav them.

The application may benefit from using MIP-maps with AGP, as MIP-map (pre-filtered multi-resolution texture maps) tend to increase the "locality" o

memory access during texturing. That is, the lower-resolution version fits i
small area of system memory, and as the graphics chip puts the texture on a
object far from the viewpoint, it accesses that sub-sampled version of textu
within a small memory region. Without MIP-mapping, the chip must skip o
many bytes of the single-resolution larger texture to find the right texel for
pixel - so memory addresses jump in large increments, and the memory
bandwidth is lowered.

Table of Contents