Optimizing Games for AMD Athlon™ 64 Processors in 2006 and beyond

Game Developer's Conference
March 2006

Michael Wall
Senior Member of Technical Staff

Advanced Micro Devices, Inc.

Agenda

This talk is for PC Game Developers.
It's about maximizing game performance.

- Dual-core and multi-core CPUs: multi-threading
- SSE vectorization: fine-grained data parallelism
- 64-bit mode: more registers and large address space
- Threading recommendations from graphics experts
- Q & A

AMD

Multi-threading

AMD

Cores, cores, and more cores

- Dual-core CPUs.
- They’re everywhere.
- Typically twice as much CPU horsepower.
- And *multi*-core CPUs are expected.

- Games must be threaded or the CPU is wasted.
- Well-threaded games can get huge benefits.

- But how do you do it?

AMD

How to multi-thread games

- Use existing threaded libraries, game engines, etc.
  - Why reinvent the wheel?
- Run DirectX on a separate thread
  - Relatively easy to do
  - Can achieve speed gains on dual-core
  - Doesn't scale beyond 2 cores
- Functional threading and producer/consumer have limits
  - Limited number of threads, may leave quad-core idle
  - Poor load balancing between threads
  - Stalling the producer also stalls the consumer
- Data parallel threading is the way to go

AMD

Data-parallel threading

- Data parallel threading is a Really Good Thing
- Parallel algorithms
  - avoid restrictive data dependencies
- Large batch sizes
- Launch many threads: scalable!
- Threads run in parallel
  - Each thread can do the same stuff…
  - …but on different data

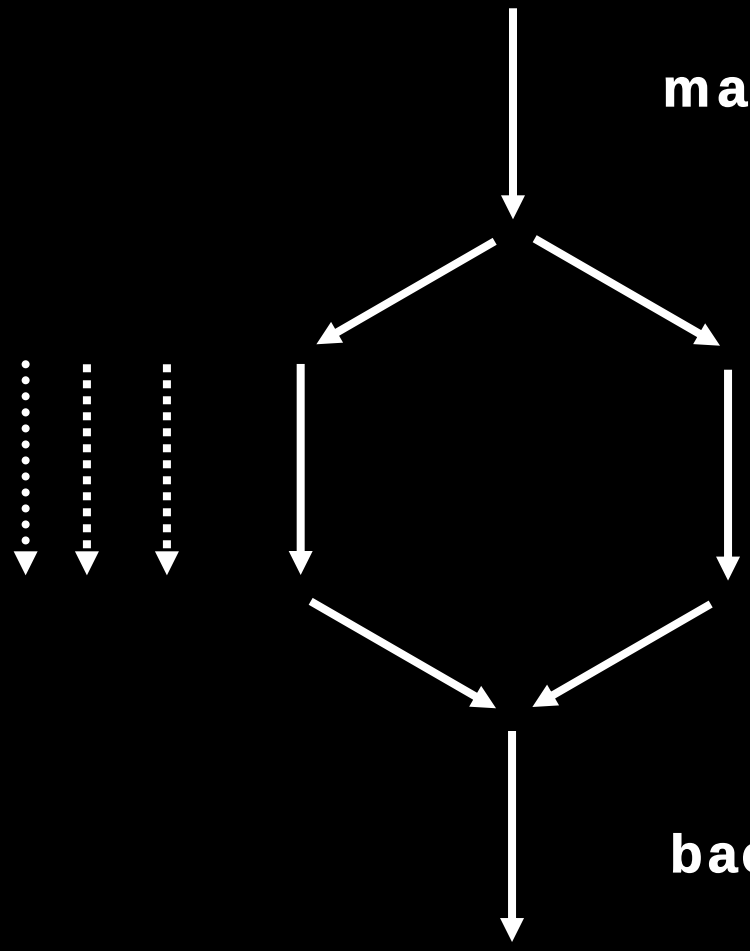
AMD

OpenMP for easy data-parallel threading

- OpenMP lets you multi-thread a loop with a single pragma!
  - `#pragma omp parallel for`
  - also options for thread scheduling, synchronizing, sections, and more
- OpenMP is supported in Visual Studio® 2005
- Implements the “fork and join” programming model
  - There is a pool of sleeping threads
  - Execution is single-thread until a “fork” is reached
  - Then Multiple threads proceed in parallel
  - Once all threads complete, resume single-thread
- The main backbone of your game is single-thread!
  - Less chance for weird synchronization bugs

AMD

Fork and Join Programming Model



main thread

fork to multiple threads

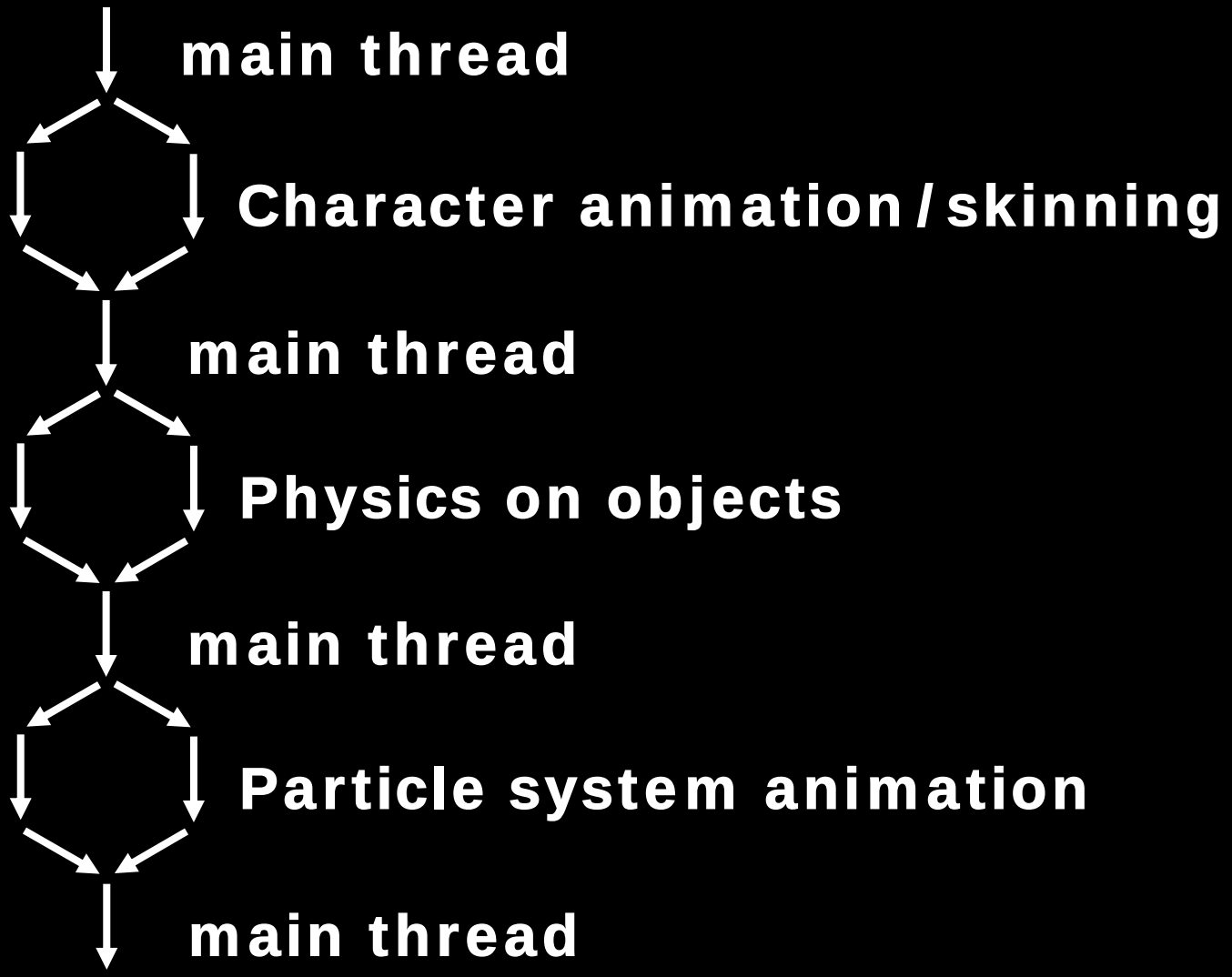
parallel execution of
heavy workload

join when all threads finish

back to main thread

AMD

Thread Example: game pipeline



AMD

OpenMP Rules of Thumb

- Thread your outer-most loop
  - Coarse-grained data parallelism
  - Minimum number of fork+join events
- Define local variables *inside* the loop
  - OpenMP will create a private copy for each thread
- Global variables
  - OK if the globals are *read only*
  - Writing to global variables will cause trouble
  - common but dangerous code:

```
*global++ = foo;   // danger! global pointer was modified
```

  - safe syntax:

```
*(global + local_offset++) = foo;
```

AMD

OpenMP isn't the only way

- OpenMP is simple to use, but has limits
- Doesn't allow control of thread priority or affinity
  - but GetCurrentThread() gives you a Windows thread handle
- Load balancing is crucial
  - all threads must wait for the slowest thread!
  - dynamic scheduling option in OpenMP can help here
- Good for threading data-parallel loops
- Windows® thread APIs work great on multi-core
  - _beginthreadex(), WaitForMultipleObjects, etc.
  - Always trust the OS about how many processors, etc.
    Don't try and directly read the hardware
  - Some good MSDN white papers about threading
    synchronize threads using interlocks, and other good stuff

AMD

A word about thread affinity

- The OS schedules threads and sets affinity for you
- Only the OS knows everything (in theory) about the execution environment
  - The PC is a general purpose multi-tasking device
  - New software and hardware can show up
- If you choose to manually set thread affinity, *be very careful*
- Test thoroughly on a variety of platforms
- Leave some secret way to adjust the behavior, just in case
- Other bits of code might be adjusting affinity too!
- DirectX can change your process affinity as a side effect:
  - IDirect3D9::CreateDevice
  - Direct3DDevice::Reset in D3D9
  - Be aware that the calling thread's affinity may get set

AMD

Maximum cache performance

- Each core has its own L1 cache, and its own L2 cache
- Greater parallelism enables added performance:
  - L2 cache latency does not increase when another L2 is added
  - Total cache bandwidth scales up linearly with the number of cores
  - Threads cannot evict another thread's data from other L2

- Avoid "cache thrashing" between cores
  - Avoid multiple threads *writing* to the same variables
  - Also avoid one thread frequently writing what another is reading
    This is just a standard rule of threaded programming
  - Beware of **false sharing** which can cause thrashing:
    *Two threads modifying **different variables** which occupy the **same cache line***
    Can happen in heap data or in static (stack) variables
    One safe approach: _aligned_malloc for 64-byte aligned heap chunks
    Another handy trick: __declspec(align(64))
    See MSDN for complete details on managing alignment

- Use AMD CodeAnalyst profiler to examine threads and cache events
  - Thrashing would appear as excessive cache refill events
  - Just one way CodeAnalyst can help you build faster code

AMD

Handy tips and tricks for threading

- Tricks for making a dual-core act like a single-core
  - Set process affinity in Task Manager to only one processor
  - Edit boot.ini file to include /numprocs=1 as alternate OS boot
- All system specs remain the same, except number of cores
- Benchmark and see the performance difference

- Don't use RDTSC instruction, use QueryPerformanceCounter
  - See Microsoft® application note about this in DX SDK

- Allow user tweaks to work around unexpected thread behavior
  - Who knows what other processes might be running
  - Offer some advanced control panel options?
    - *Adjust or limit number of threads*
    - *Set or disable thread affinity*

AMD

SSE vectorization

Games do lots of heavy arithmetic

- Much heavy processing in games can be SSE vectorized
  - Physics, particles, collision detection
  - Animation and skinning
  - Image processing, scaling, synthesis
  - Audio synthesis

- SSE vectorization can accelerate all this processing…

- How to implement SSE optimization?

AMD

Using vectorized SSE in your game

- Use existing libraries, engines, etc. which are SSE optimized
  - Why reinvent the wheel?
- Use the D3DX math functions in the DirectX SDK
  - Optimized for all popular CPUs
- Code your own algorithms directly using vectorized SSE
  - Data parallelism, logically much like data parallel threading
  - Use mask instructions to eliminate branch instructions
  - Floating point (SSE) and integer (SSE2) data types
  - Use CPU ID to detect current and future SSE capabilities

AMD

Compiler intrinsics = portable SSE

- Use Visual Studio® 2005 compiler intrinsic functions!
  - It's like ASM code at the C/C++ source level
  - The compiler does register allocation and scheduling
  - Unlike ASM, it compiles for 32-bit mode and for 64-bit mode
  - 64-bit mode transparently offers 2x more SSE registers

```
__m128 a = _mm_mul_ps( z_real, z_real );
__m128 b = _mm_mul_ps( z_imag, z_imag );

__m128 t = _mm_add_ps( a, b );  // check magnitude of z
__m128 t_mask = _mm_cmpgt_ps( t, _mm_set1_ps( 4.0 ));  // 4 compares

   int m = _mm_movemask_ps(t_mask); // get a 4 bit int mask
   if(m & 1) { "do stuff here" }    // no shuffling is needed!
   if(m & 2) { "do other stuff" }
```

AMD

MASM works fine for vectorized SSE

- Visual Studio® 2005 has 32-bit and 64-bit MASM
  - 64-bit ASM code is *not the same* as 32-bit
    same instruction set, but 64-bit pointers and more registers
  - VC2005 64-bit has no in-line ASM! Must use MASM 64 for ASM
  - New and improved calling convention for 64-bit Windows®
    args get passed in registers like fastcall

```
; random examples of 64-bit ASM code instructions
    movapd  xmm0, [rcx]
    movapd  xmm14, [rdx]
    xor     rax, rax
    mov     edx, 4
    shufpd    xmm15, xmm15, 1
    mov       r9, r8
    shl       r9, 32
    subpd     xmm4, xmm5
    addpd     xmm4, xmm14
```

AMD

Summary: use both kinds of data parallelism

- Data-parallel threading for coarse-grained
- Vectorized SSE for fine-grained
- Each CPU core is a complete x86+SSE unit
- Ergo: use vectorization *and* threading

AMD

64-bit mode

PC games need to keep raising the bar

- Gamers want more of everything
- Everything needs to be stored somewhere
- Windows® limits applications to 2GB address space
- And even graphic cards have 512MB now!

AMD

Memory space is a good thing

- Large memory is an advantage of PC over consoles!
- Game PCs have 1GB – 2GB *physical* RAM
  - Approaching the size of the *virtual* address space!
- 32-bit Windows® has 2GB virtual space for a game
  - Virtual address space can get fragmented
  - You may not even be able to fully use 1GB of physical RAM!
- The time is ripe for 64-bit virtual address space
  - Even if the machine has less than 4GB physical memory
  - And certainly for the enthusiast who stuffs > 2GB

AMD

64-bit mode: lots of benefits

- Large address space is not the only 64-bit benefit
- Bonus: twice as many registers
  - 16 GPR registers
  - 16 SSE registers (SSE and SSE2 instruction support guaranteed)
  - Register pressure was the biggest limit of x86… now it is fixed
- Visual Studio® 2005 x64 compiler uses all the registers
- Computationally intensive code can get a speed boost

AMD

64-bit mode: wait, there's more!

- Visual Studio® 2005 x64 compiler generates great code
  - Whole Program Optimization
  - Profile Guided Optimization
  - /fp:fast mode for improved floating point performance
  - _restrict keyword: hint to compiler about pointer aliasing
  - Intrinsic functions: data prefetch, streaming store, SSE, …
  - Optimized libc functions like memcpy, memset, strcmp, etc.
  - → Be sure to try /O1 minimize code size, especially for 64-bit
- DirectX and D3DX have optimized 64-bit code
- Graphics IHVs have significant investment in x64 performance

AMD

A word about data bloat

- In 64-bit mode, pointers are 8 bytes instead of 4 bytes
- Data structures containing pointers will grow
- In some cases, data structures will grow a lot

- Games make heavy use of system memory
- System memory bandwidth may be pushing the limit
  - How many bytes of memory traffic happen per frame?
- Data bloat can adversely impact performance

- Reduce data bloat:
  - Use array[index] or *(base+offset), instead of *pointer
  - Re-order elements of a struct for better packing

AMD

64-bit mode will soon be common

- Windows® Vista™ will offer both 32-bit and 64-bit flavors
  - 64-bit games for Vista are expected to get more attention
  - And 64-bit Windows XP Pro has been shipping since April 2005
- Competitive gaming CPUs all support 64-bit mode
- VS 2005 and DX SDK fully support 64-bit development
  - And so does AMD CodeAnalyst profiler
- Optimal scenario: design PC game for 64-bit from day #1
  - example: map your 10+ GB of resource files to virtual memory
- Caveat: all the code in the game must be native 64-bit
  - Cannot mix and match 32-bit and 64-bit
  - Can't use MMX in 64-bit code, use SSE2 which is always supported
  - Most game libraries, engines etc. are already 64-bit ported

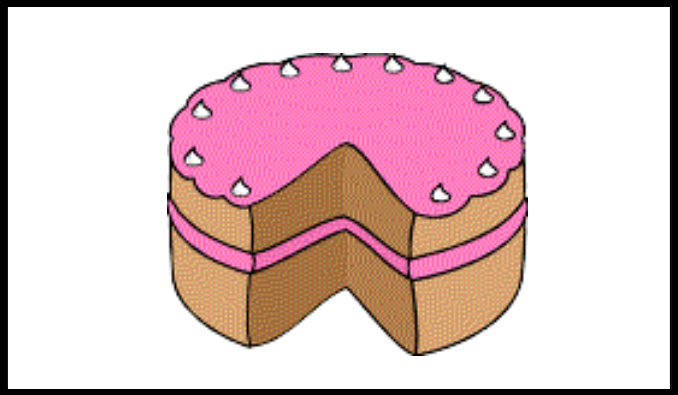
AMD

Putting it all together:

“The PC Game Layer Cake”

AMD

The PC Game Layer Cake



- The slices of cake are the many threads
  - Coarse-grained data parallelism, multi-threading
- The layers are the processing steps in each thread
  - Fine-grained data parallelism, opportunity for SSE
- 64-bit technology is frosting on the cake
  - Greater memory space and more registers to make it even better!

AMD

Resources

- Go to developer.amd.com and get all the AMD docs
  - Optimization Guide, Programmer's Manuals, white papers, etc.
  - Download and use the CodeAnalyst profiler, for 32 and 64-bit code
  - Visual Studio® 2005 demo project and white paper: "Performance Optimization of 64-bit Windows® Applications using Visual Studio 2005, on AMD Athlon™ 64 and AMD Opteron™ Processors"
  - Other presentations, including TechEd and previous GDCs
- Sign up to use the AMD Developer Center in Sunnyvale, CA
  - Work on-site
  - Remote access by VPN to servers
- Also check out amd.com and especially the "gaming" link
  - Success stories, downloads

mike.wall@amd.com

AMD

More Resources

- Go to MSDN and get the Windows® x64 OS, Visual Studio® 2005
  - DirectX® for x64: already released in DirectX 9.0 SDK
- Go to MSDN and Microsoft.com for more docs
  - Search for 64-bit, AMD64, x64 or “64-bit Extended”
  - Read about new 64-bit compiler features, OpenMP, intrinsics, etc.
  - Read about threading and fast thread synchronization tricks
  - Especially read about VS2005 “Whidbey” performance optimization features
- OpenMP is simple and powerful www.openmp.org