AMD DEVELOPER SUMMIT

MANTLE UNLEASHED: HOW MANTLE CHANGES THE GAME

DAN BAKER – PARTNER, OXIDE GAMES
TIM KIPP – PARTNER, OXIDE GAMES

AMD

About Oxide Games

- New Studio founded from team with decades of experience in strategy games
- Games include : Sid Meier's Civilization V, Command & Conquer Kane's Wrath, Lord of the Rings: Rise of The Witch King and many more*
- Committed to PC gaming – Oxide believes the best is yet to come for PC strategy
- www.oxidegames.com

Why Nitrous?

- Strategy games have unique demands
- Landscape has changed, 64 bit, 8 CPU cores now common, GPUs have terraflops of power
- But… difficult to use all this power
- Engineering difficulties involved in fully utilizing modern PC system are daunting
- Limiting factor: Strategy games all about the number of units

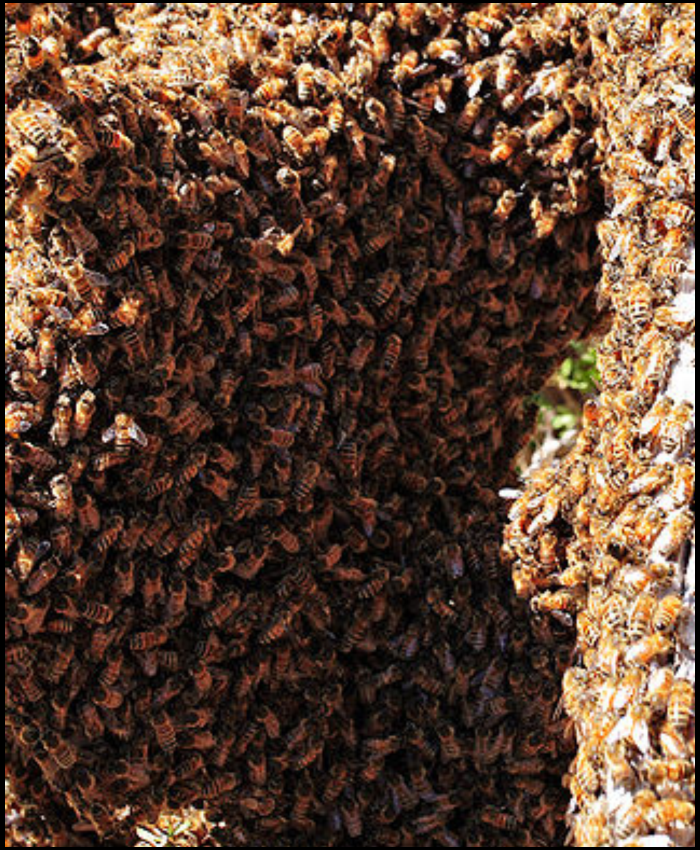
Nitrous Rendering

- Next gen engine: Built more like a film renderer
- Shading decoupled from rasterization
- Color values all HDR
- Tone mapping curve stolen from the movie industry
- Bloom, motion blur, lens effects are all emergent properties of the camera simulation, rather than effects

SWARM

- **S**imultaneous
- **W**ork
- **A**nd
- **R**endering
- **M**odel

Fancy Acronym, what is it really about?



SWARM – What is it?

- Task based application model
- Based on years of multi-core experience
- Designed to handle large complex applications/sims.
- Combination of independent systems (physics, AI, Animation, Graphics, etc)
- Scalable, > 10k tasks per frame with minimal performance implications, and **eager** for more cores

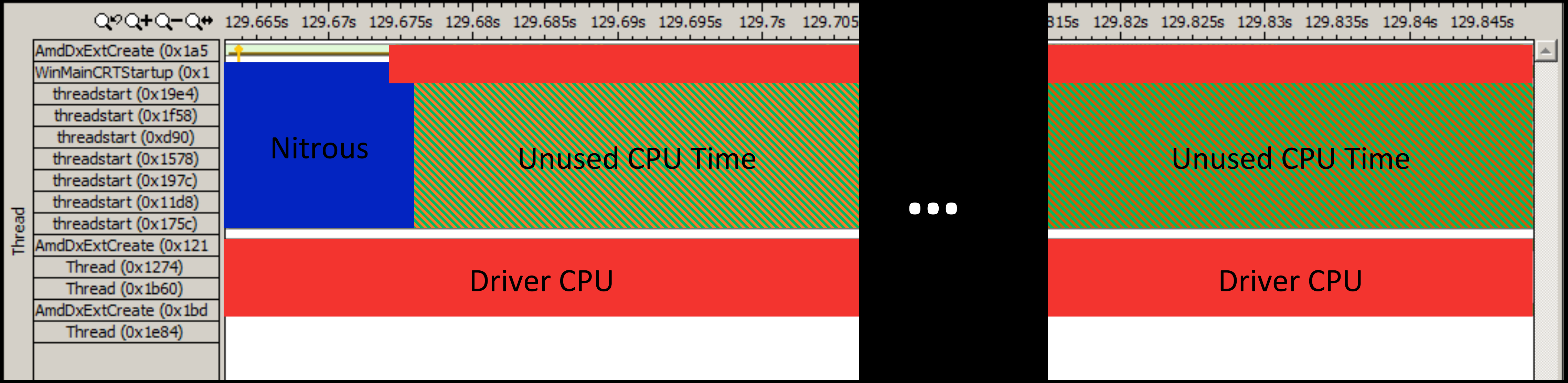
SWARM – Goals

- Push the boundaries of games: Go where no one has gone before
- Use as many cores as users have: 4 Cores is standard
- Plan for future: > 8 cores.
- Remove serial threads, e.g. functional threading: All components should scale across cores
- Fully utilize the GPU: Co-operative task scheduling/work sharing for CPU and GPU

Bottlenecks

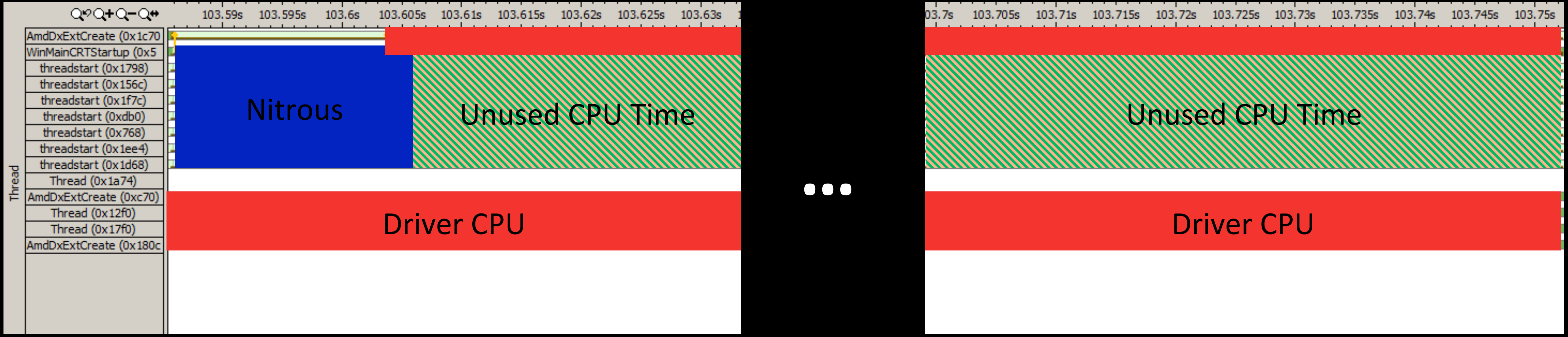
- GPUs are fast – if we can keep them fed
- PCs have many CPU cores + a tons of RAM. How can we utilize all this power?
- All segments of the engine need to be dealt with
  - Physics
  - AI
  - Scene management
  - Gameplay
- SSE, cache , memory management all critical to getting performance
- Nitrous manages all of these things well so that we can handle scenes with 10,000+ units at up to 60fps

A frame without Mantle



- 4 Core 8 Thread CPU
- ~10k units, ~50k batches
- > ~99ms CPU execution + 0ms GPU = > ~99ms frame time.
- 6 Additional Driver Threads, 3 Active

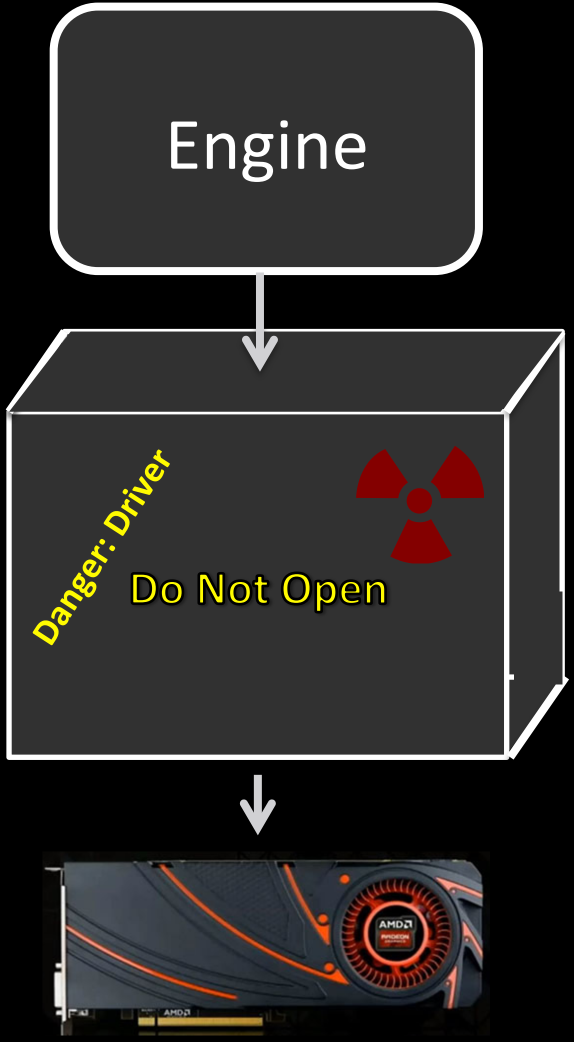
A frame without Mantle



- Deferred Contexts – Still lots of (unusable) green!
- 4 Core 8 Thread CPU
- ~10k units, ~50k batches
- > ~99ms CPU execution + 0ms GPU = > ~99ms a frame.
- 6 Additional Driver Threads, 5 Active

Hard to optimize for driver

- A big black box, not clear what causes overhead
  - Trial and error
- Driver threads conflict with our threads
- Solutions:
  - Organize art assets to minimize state changes
  - Move camera in closer
  - Decrease the number of possible units
  - Back off on number of threads we use
- For a strategy game, difficult to get GPU limited



What is a batch?

- Individual command sent to the GPU
  - Models
  - Effects
  - Particles
  - GPU Compute
- Analogous to a function call
  - 10s of millions per frame on CPU, but can do comparatively few on GPU

"You know it's bad when…"

- Optimization for driver passes all the way through to production artists
- Artists pack things into same textures, even when not really appropriate
- Artists optimize batch count
- Designers have to restrict design purely because of batch count issues

  "Hey, do you think we have 20 armies, each with 100 units? Oh, and 4 factions. Oh, and shadows. Oh, and reflections in the water."

  "I'd really like to be able to zoom out further. Yeah, seeing the entire map at once would be great!"

Lots of suggestions

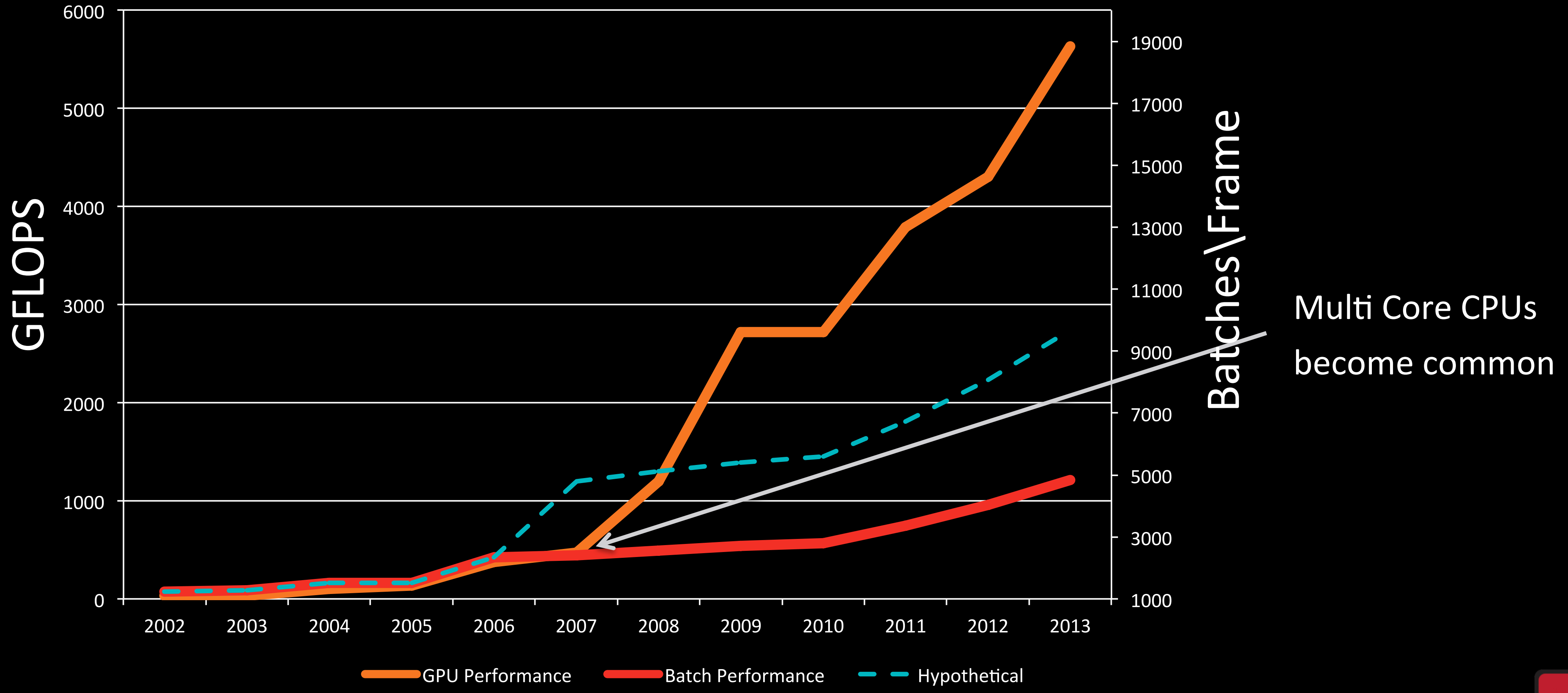
- Bindless textures
  - Makes textures global, shader logic more complex, management more complex (and textures already typically grouped by artists)
- Geometry Instancing
  - Optimizes the fastest type of call, serializes engine and increases costs on development
- Object packing
- Deferred Contexts – failure. Minimal measurable gains

Just Band-Aids

Hurts the game

- Why should we ask game designer to limit the user experience for avoidable problems?
- Cost us money just to iterate on ideas
- What cool games are we missing out on? Epic Space Battles? Massive Medieval Warfare? Huge, sprawling cities?
- Batch performance is now a crisis level situation

Why has this happened?



Causes of driver overhead?

- Oxide has spent a good deal of time analyzing driver overhead:
  - Mismatch of API to hardware capabilities
  - Lack of ability for app to manage GPU memory in any meaningful way
  - Requirement for driver to track and deal with hazards
  - Driver and API to have serial points, limits scalability across cores
  - **Mile high view: APIs have not evolved with hardware**

Is there a solution?

- Overhead much lower on consoles
- But… at the expense for devs to manage more ourselves
- Potential to lose abstraction over hardware
- What if… could abstract hardware but still provide console like performance?
- Developers have been asking for lower level solution for years

Now is the time

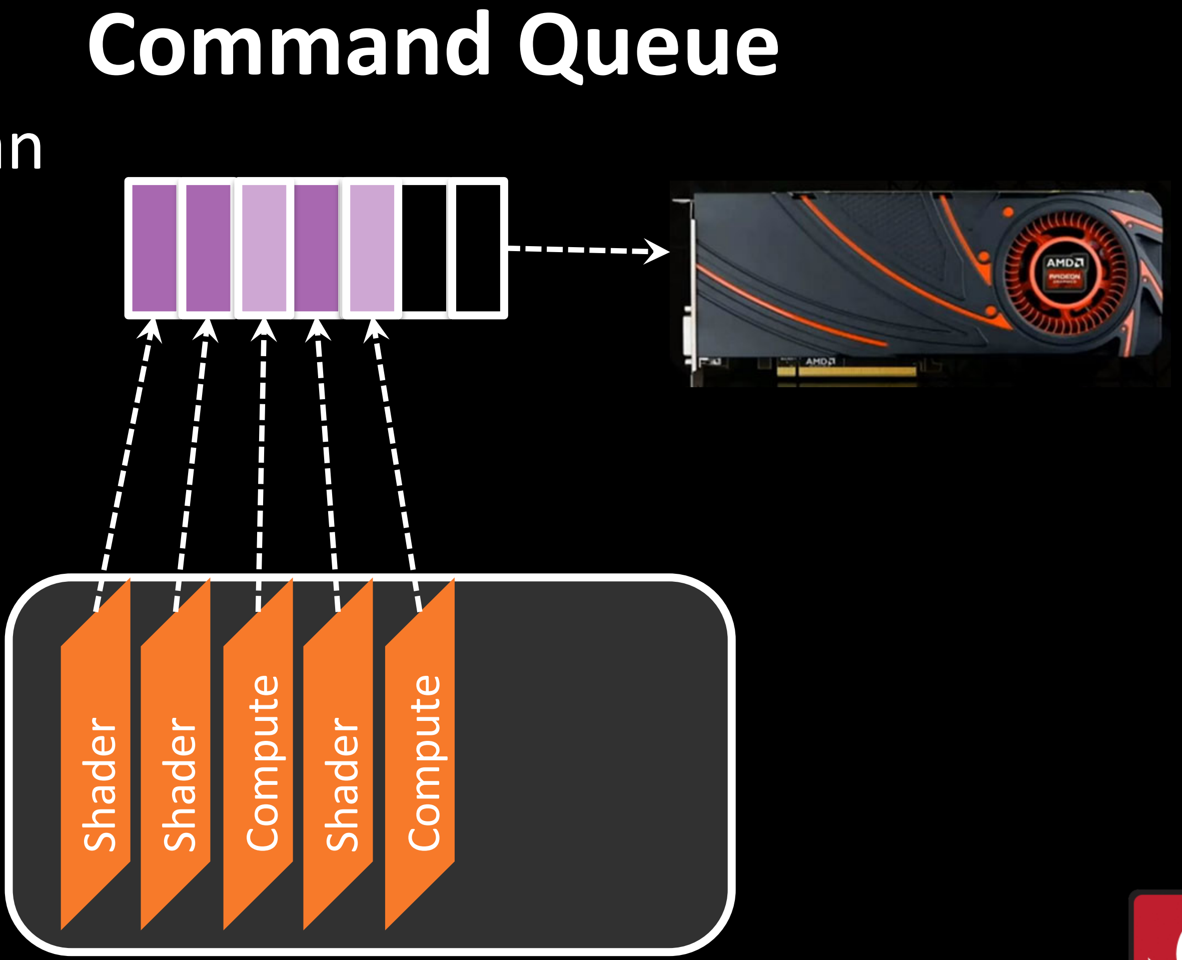
- GPU architectures are becoming more stable, less radical from version to version
- Data types are standardized, GPU and CPU share many data formats
  - Long term trend, GPU/CPU converge
- GPU is a general computing device, limited mostly by performance
- Unified Memory is coming…
- If not now, when?

Enter Mantle

- Build on the concept: Less is more
- Puts trust in developers – a contract between the app and the driver, app doesn’t do ‘bad’ things, driver can be faster
- Hardware still abstracted, but memory, command hazards are not
- However, Nitrous already handles these issues
    - All high performance engines need to
    - All console engines need to
    - Driver handling these just causes slow down and decreased reliability

Anatomy of a modern API

- GPUs are processors that run programs (shaders), each has an input, and an output
- API need do no more than provide a way of executing shader programs
- No implicit CPU cost, hidden threads, hidden perf implications



Supporting Mantle:

What it took to get Nitrous running on Mantle?

- For a modern Engine, Mantle not difficult or expensive to support
- Approximately 2 man months of work to support Mantle
  - On an pre-alpha version of Mantle, with minimal sample code
  - Nitrous uses everything, UAVs, atomic instructions, compute shaders, GPU generate commands etc
- Only minor changes to the Engine
- Mantle not significantly more code:
  - Mantle ‘driver’: about 4500 lines of code
  - Non Mantle: about 3500 lines of code

Supporting Mantle:

What it means for Oxide

- High batch performance decreases cost
- Enables new types of games, perhaps new genres
- New rendering and gameplay techniques
- Less time spent on stability, optimizations
- Huge performance gains even without using Mantle specific features
  - It's just… faster….

Business case:

- Inexpensive to support
- Changes made to support Mantle typically helps other APIs
- The bulk of cost is fixed. Port once, use for years
- Huge benefits to a large percentage of our customers
- Helps push the industry forward

Supporting Mantle:

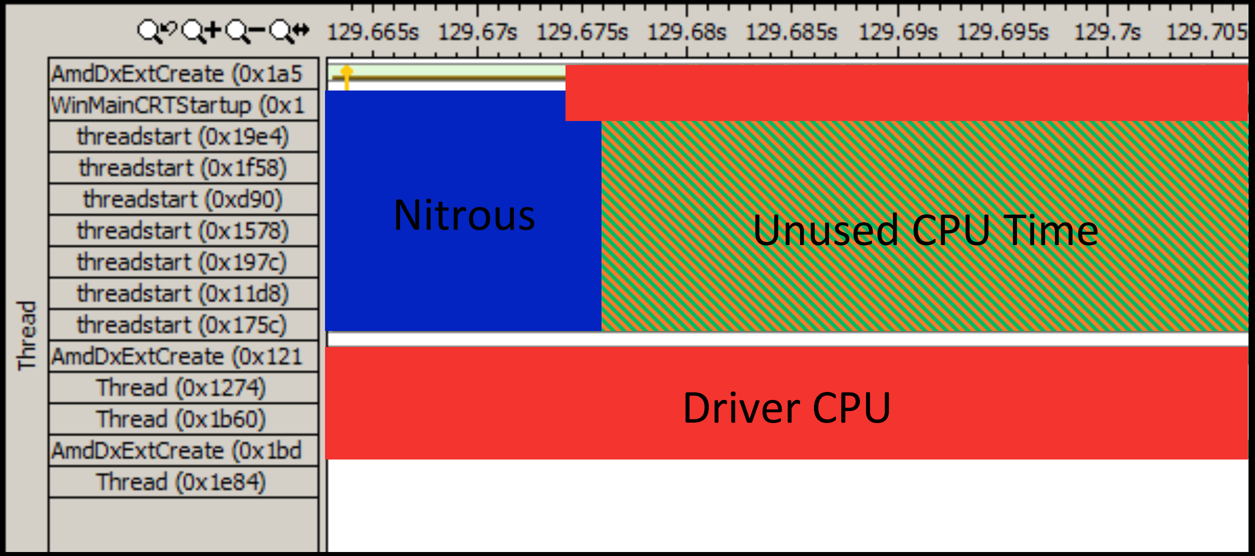
What it means for consumers

- More responsive app
- New effects, realism
  - Allows Nitrous to add features such as Film Quality Motion Blur
- Unlocks high end GPUs
- Blazing fast CPU not necessary, a mid-range CPU will be sufficiently fast to drive the best GPU
- More CPU cores = clearly better. Not limited by the speed of 1 core

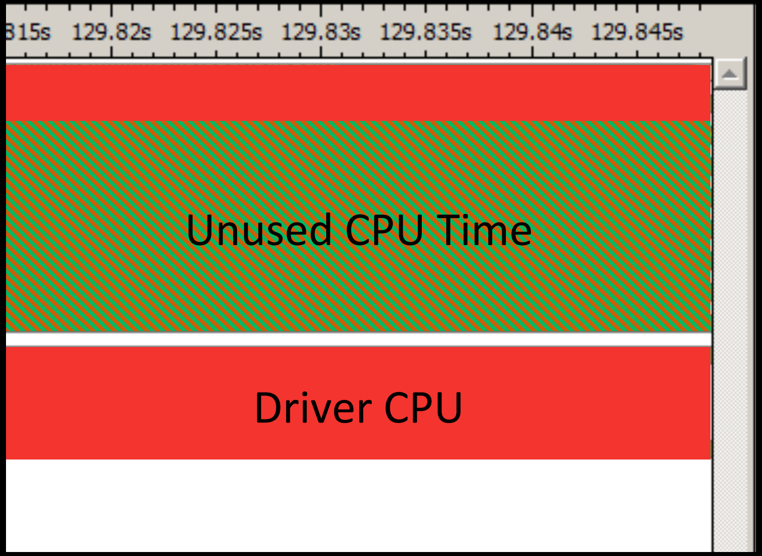
Enough talk, show & tell:

- StarSwarm: Pre-Alpha Nitrous Engine Demo
- Based on our internal engine test,
  - Only a few months of art assets and a little spit shine polish (excuse our mess)
- Demonstrates what a 100k batch game might look like:
  - Not a precanned demo, a game with 2 AI opponents, representative of Nitrous running a game
- Plan to publically release in Q1, 2014
  - Content is modifiable, so public can experiment for themselves

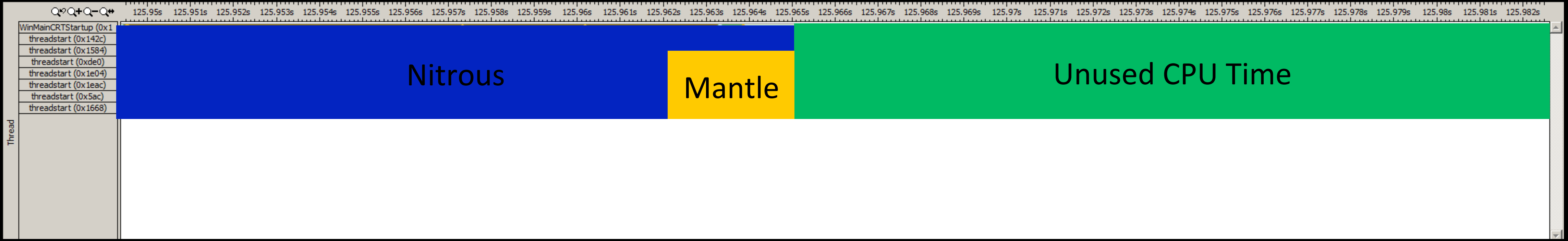
Review: Before Mantle



...



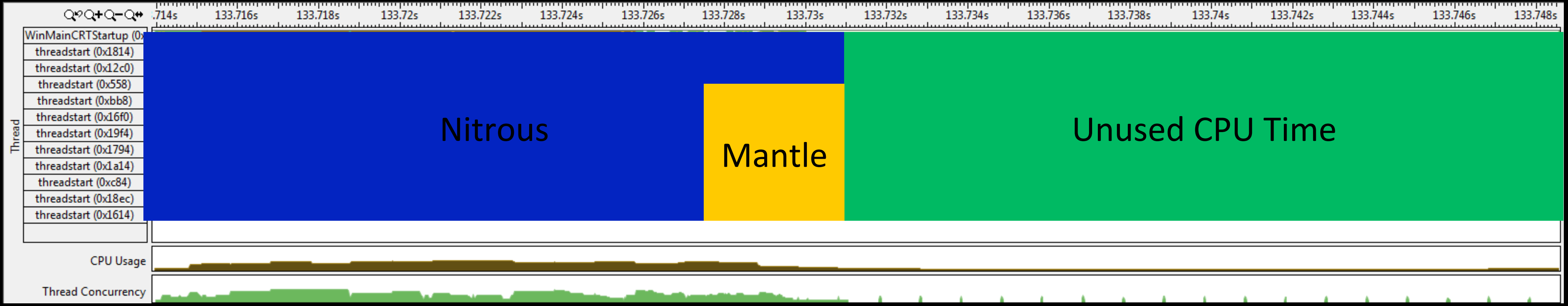
A frame with Mantle



125.95s 125.951s 125.952s 125.953s 125.954s 125.955s 125.956s 125.957s 125.958s 125.959s 125.96s 125.961s 125.962s 125.963s 125.964s 125.965s 125.966s 125.967s 125.968s 125.969s 125.97s 125.971s 125.972s 125.973s 125.974s 125.975s 125.976s 125.977s 125.978s 125.979s 125.98s 125.981s 125.982s
Thread
WinMainCRTStartup (0x1
threadstart (0x142c)
threadstart (0x1584)
threadstart (0xde0)
threadstart (0x1e04)
threadstart (0x1eac)
threadstart (0x5ac)
threadstart (0x1668)
Nitrous
Mantle
Unused CPU Time

- 4 Core 8 Thread CPU
- ~10k units, ~50k batches
- ~18ms CPU execution + ~15ms waiting for GPU = ~33ms a frame.
- No Driver thread at ALL!
- Lots of clear green!

Mantle: 12 threads



- 6 Core 12 Thread CPU
- ~10k units, ~80k batches
- ~18ms CPU execution + ~15ms waiting for GPU = ~33ms a frame.

Results

- API overhead is reduced by at least 10x
- Mantle scales much better on CPU cores – so true performance is much higher than 10x when measuring total time spent in driver
- At an application level, StarSwarm is 3 times faster in some cases
- Completely changes the landscape for multi-core
- CPU benchmarks indicate that AMD 8350 is comparable to a Core I7 4770k for performance. Work can now scale across more cores
- On a RX290, Still GPU bound even when 8350fx is underclocked to 2 ghz

Point of no Return

- Mantle Proves “It can be done”
- 100k+ batches is a reality, with CPU room to spare
- Just the beginning: Only basic optimizations: Mantle, Nitrous continue to improve
- 2015: 300k batches
- 2018: 1 million batches
    - GPUs and CPUs start to call code on one another
    - GPU used as a co-processor even in non multi-media apps

Acknowledgements

AMD Mantle Team

Ironclad Games

Stardock

DISCLAIMER & ATTRIBUTION