Dan Baker Oxide Games

D3D12 Future VR and beyond

- Learned a lot about the performance of D3D12 engine with Nitrous 1.0
- What does a second gen D3D12/Vulkan engine look like?
  - Direct control of synchronization primitives: gives us control where we need it
  - Multi Core rendering: allow for lower latency
  - More complex multi-engine (aka async compute): allows high efficiency for VR

Dan Baker Oxide Games

New challenges, new terms

- Average FPS not a useful metric
  - Must run at 90, consistently
- How to measure performance of an Engine on a given System?
  - CPU speed – all the stuff that a CPU has to do to run our game scene filled with objects. Physics, AI, skinning, simulation, gameplay etc.
  - GPU speed – what we need to render the scene on a display, VR or otherwise
- App Motion to Photon Latency is known quality bar, but how do we improve?

Dan Baker Oxide Games

A better term for CPU performance

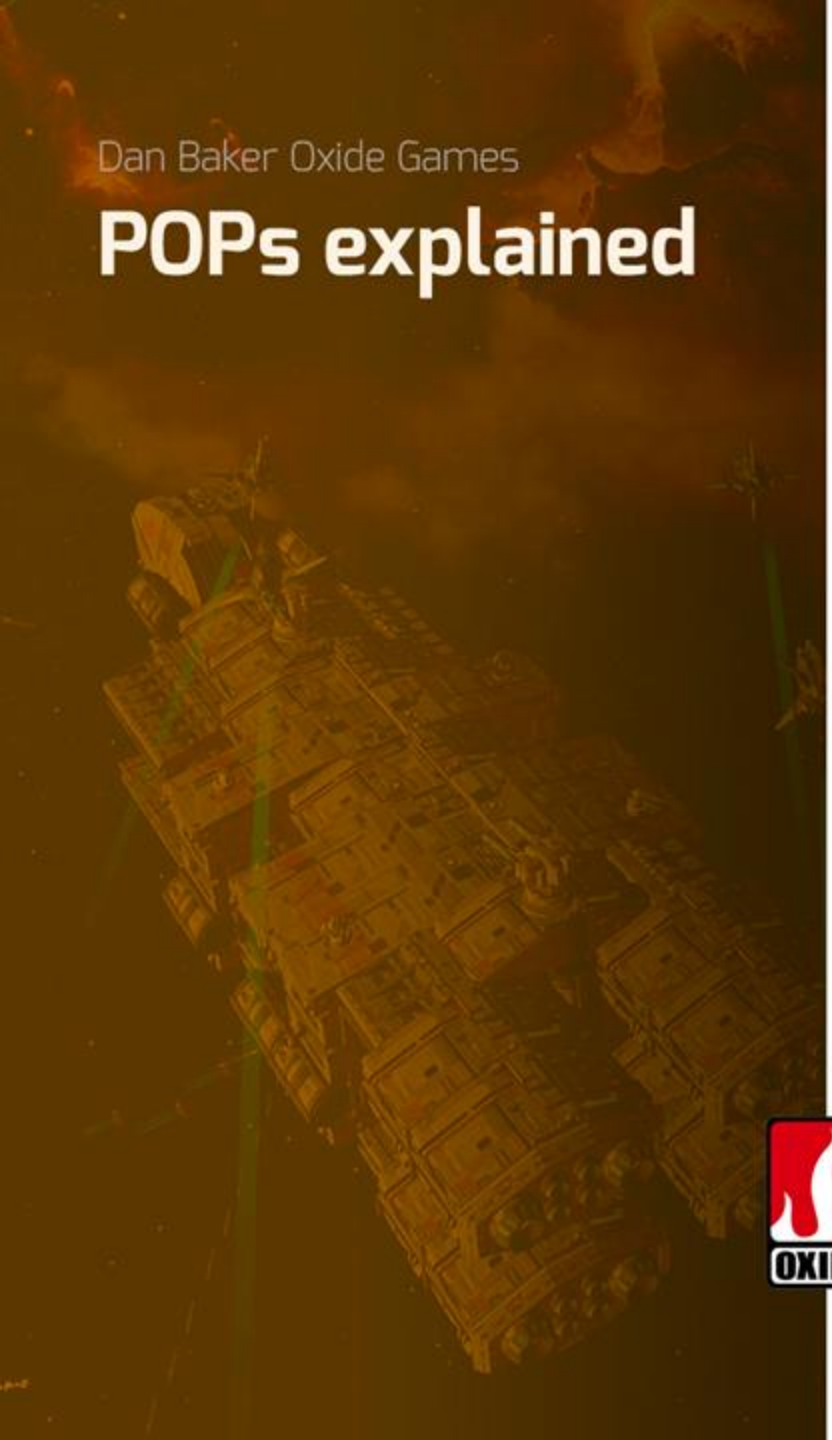
- Need to understand maximum load for a given system e.g. like max towing for a truck, or max loaded weight for a bridge
- Number of Objects per unit of time on a given system

POPS

Processed Objects Per Second

Dan Baker Oxide Games

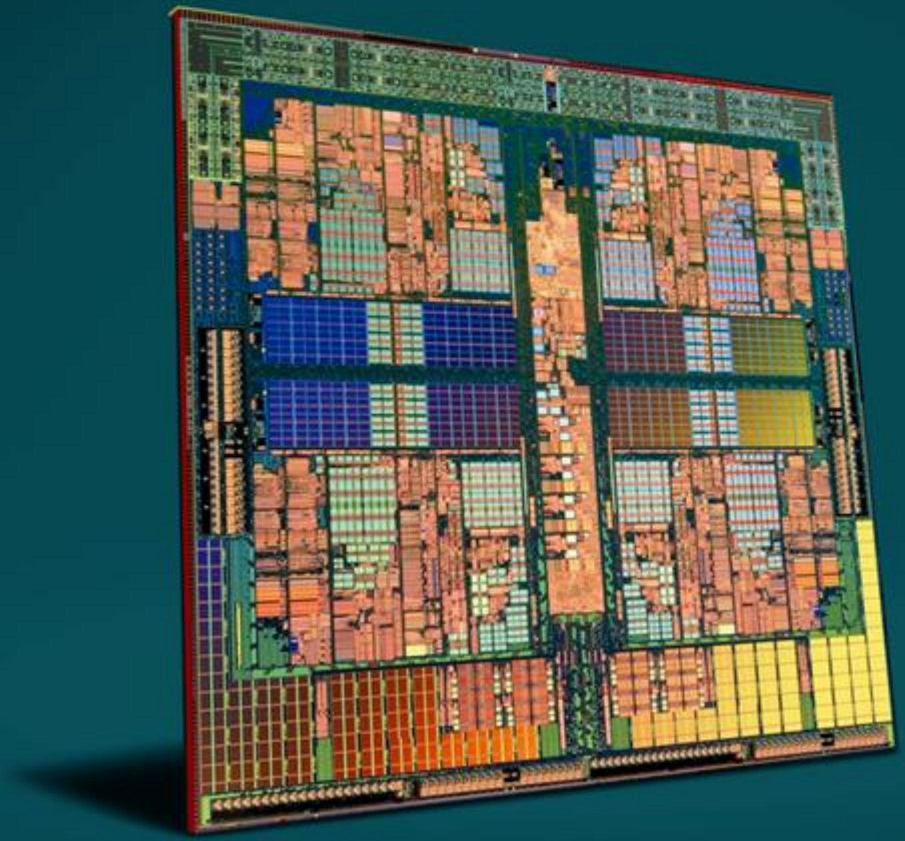
POPs explained



- Simple concept: Many engines can become CPU bound. Becomes increasingly difficult to run simulations at a consistent 90 fps
- 90 fps is misleading, because of time to photon-loop, every millisecond that can be eliminated improves experience
- Even if engine is fast enough to run all CPU side work in 11 ms, a better experience if it can handle it in far less time (e.g. 5 ms)
- TLDR: higher POPS = better experience. But how do we get a higher POPS?

Dan Baker Oxide Games

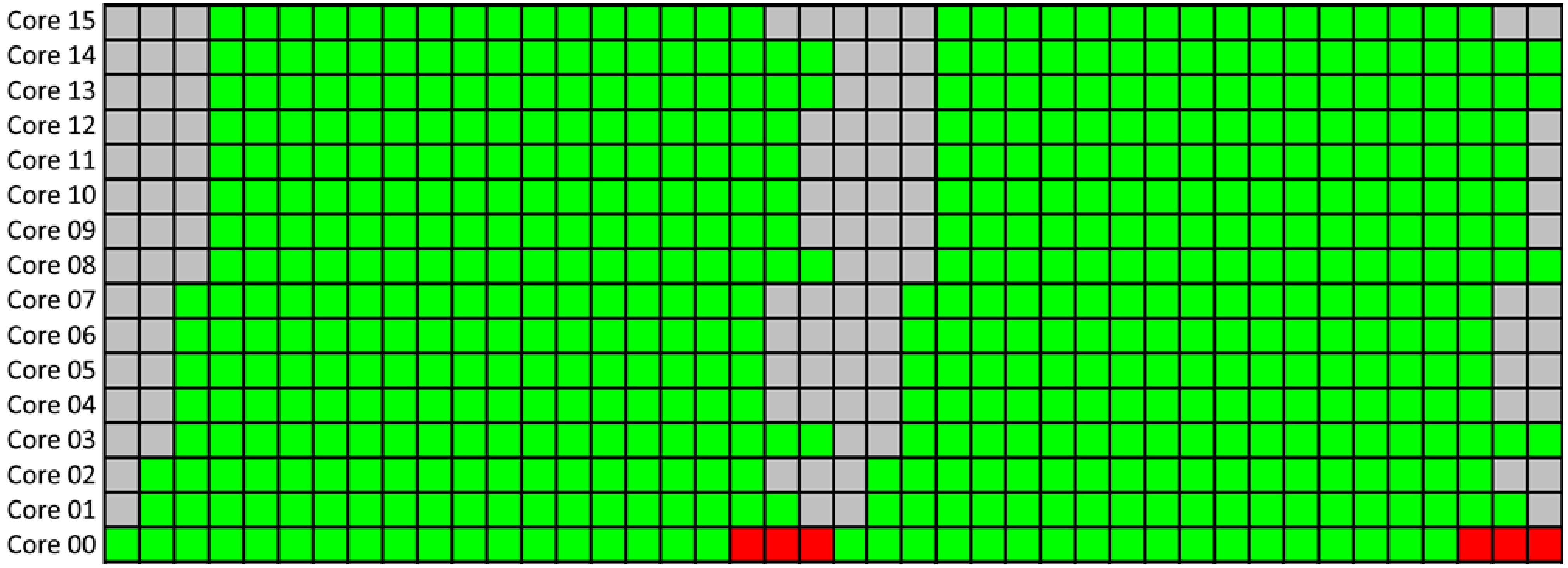
Efficient use of modern CPU – single core



- Modern CPUs are fast
- Eliminate:
  - Arbitrary branches
  - Deep call stacks
  - Poor cache use
- Do
  - SSE instructions
  - Vector Math
- Not small gains, most code could be 10x faster!

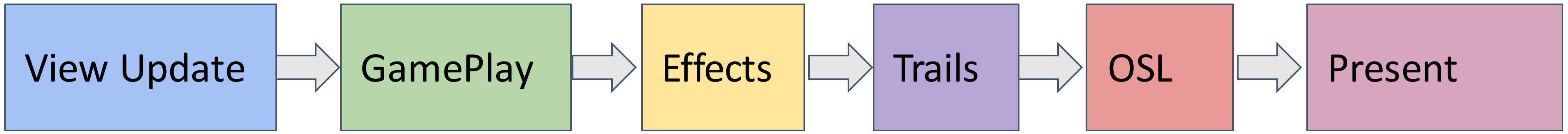
Architecture: App: Starswarm

Execution gaps due to warm up and imperfect execution of workload



Architecture: Challenge

Chains of dependent systems can cause system level serialization.

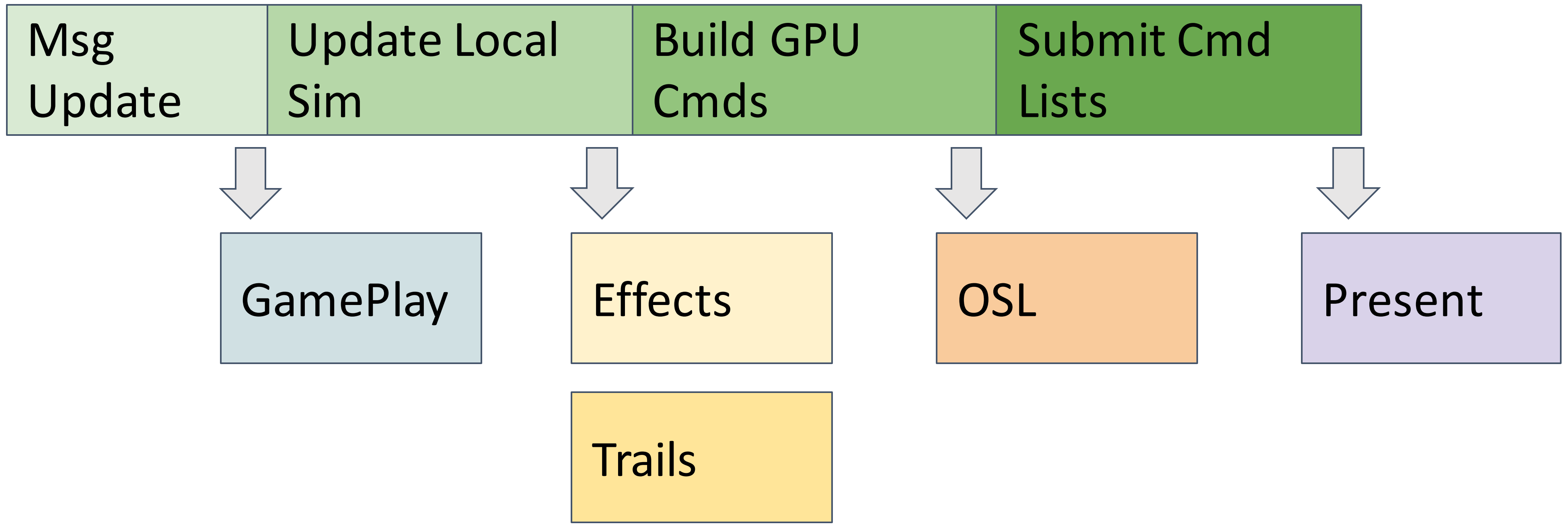


Delayed processing (double/triple buffering) can help address this, but at the cost of simulation and visual fidelity. Fast moving objects, fast camera can make this problematic.

Architecture: Ashes: Systems Multi-Stage

Design systems to have multiple stages, useful to satisfy dependencies as quickly as possible, as well as organize the frame better for performance.

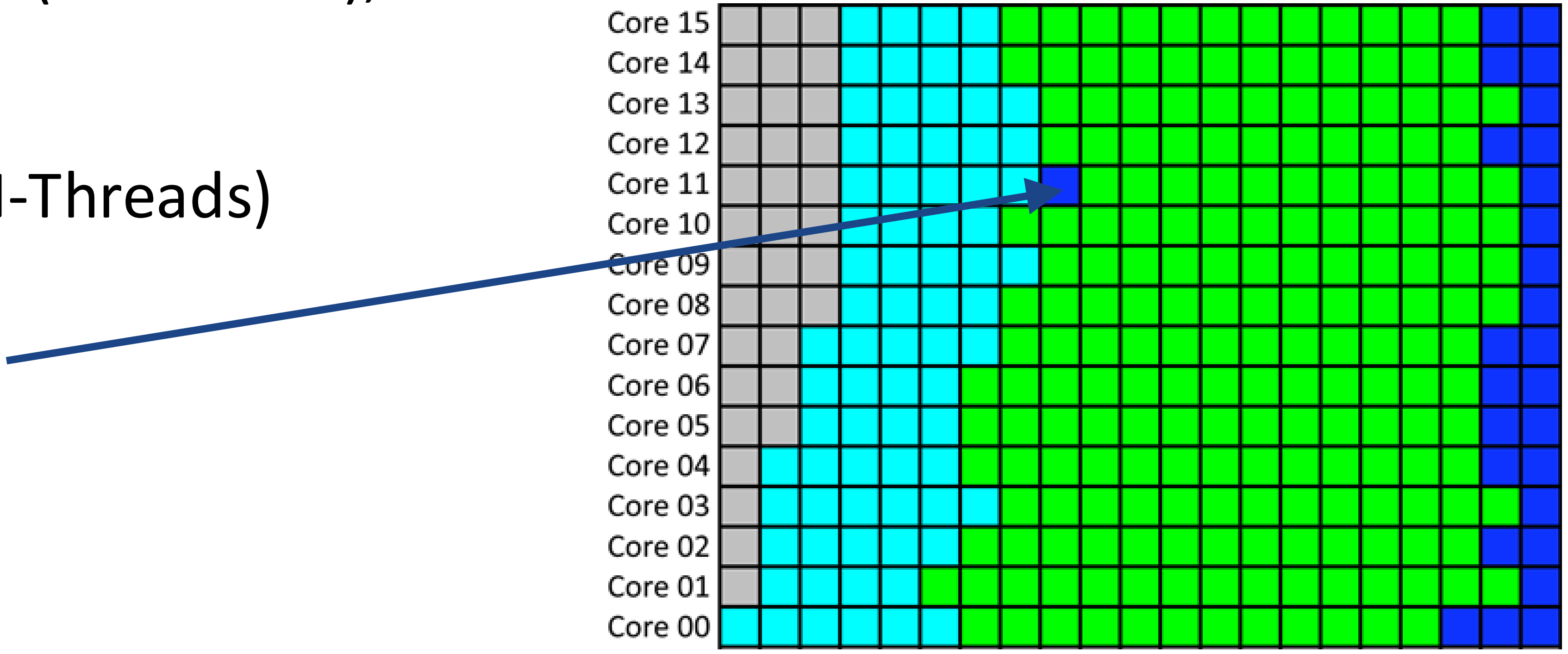
Model Views: Multiple Phases



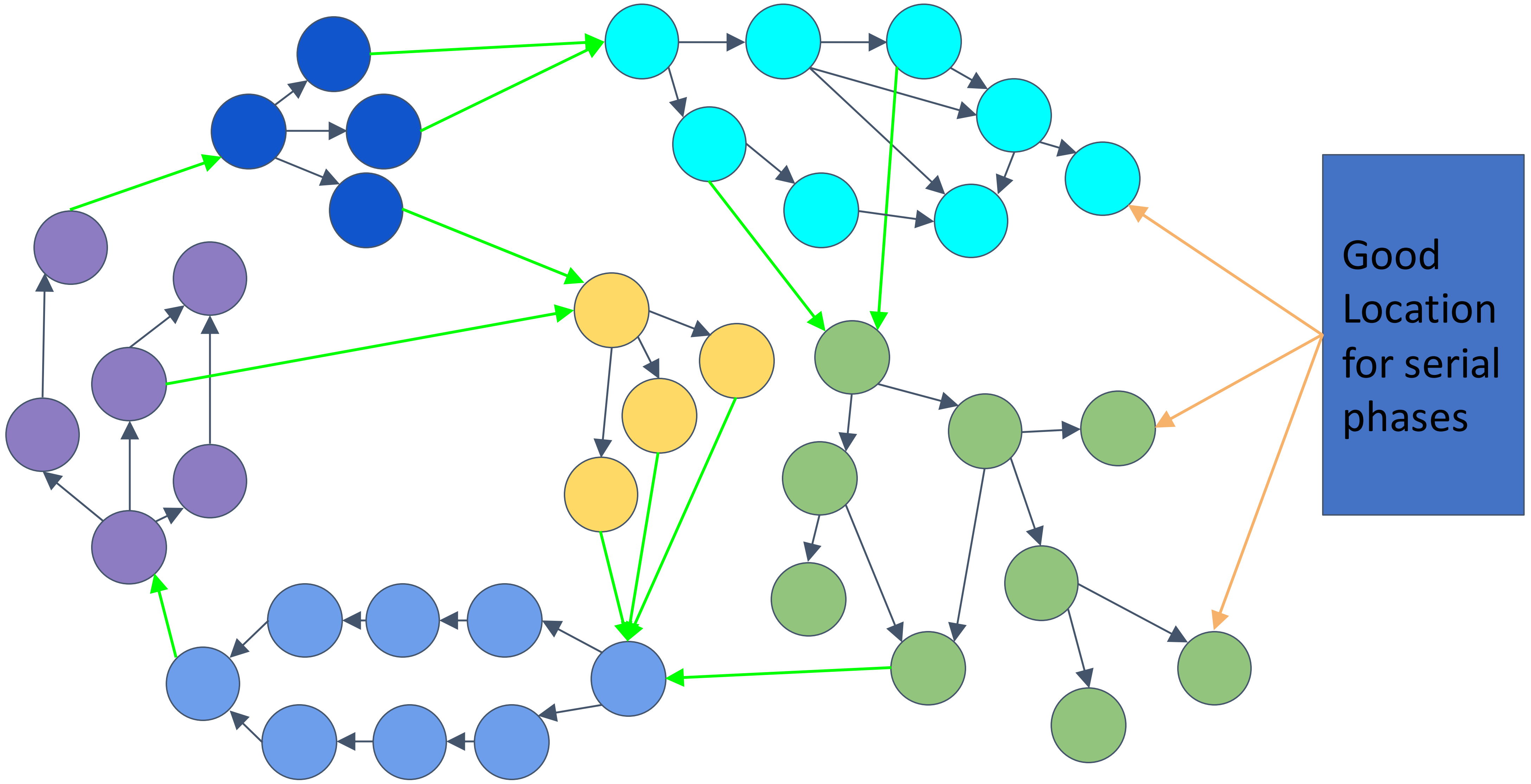
Architecture: Ashes: 14k Avg Num Tasks

Manual Priming for multi-parallel execution with some signals
Modules->Update(N-Threads, &PhysicsSignal);
Projectiles->Update(N-Threads);

On PhysicsSignal()
Physics->Update(N-Threads)



Nitrous 2.0: App as Collection of DAGs

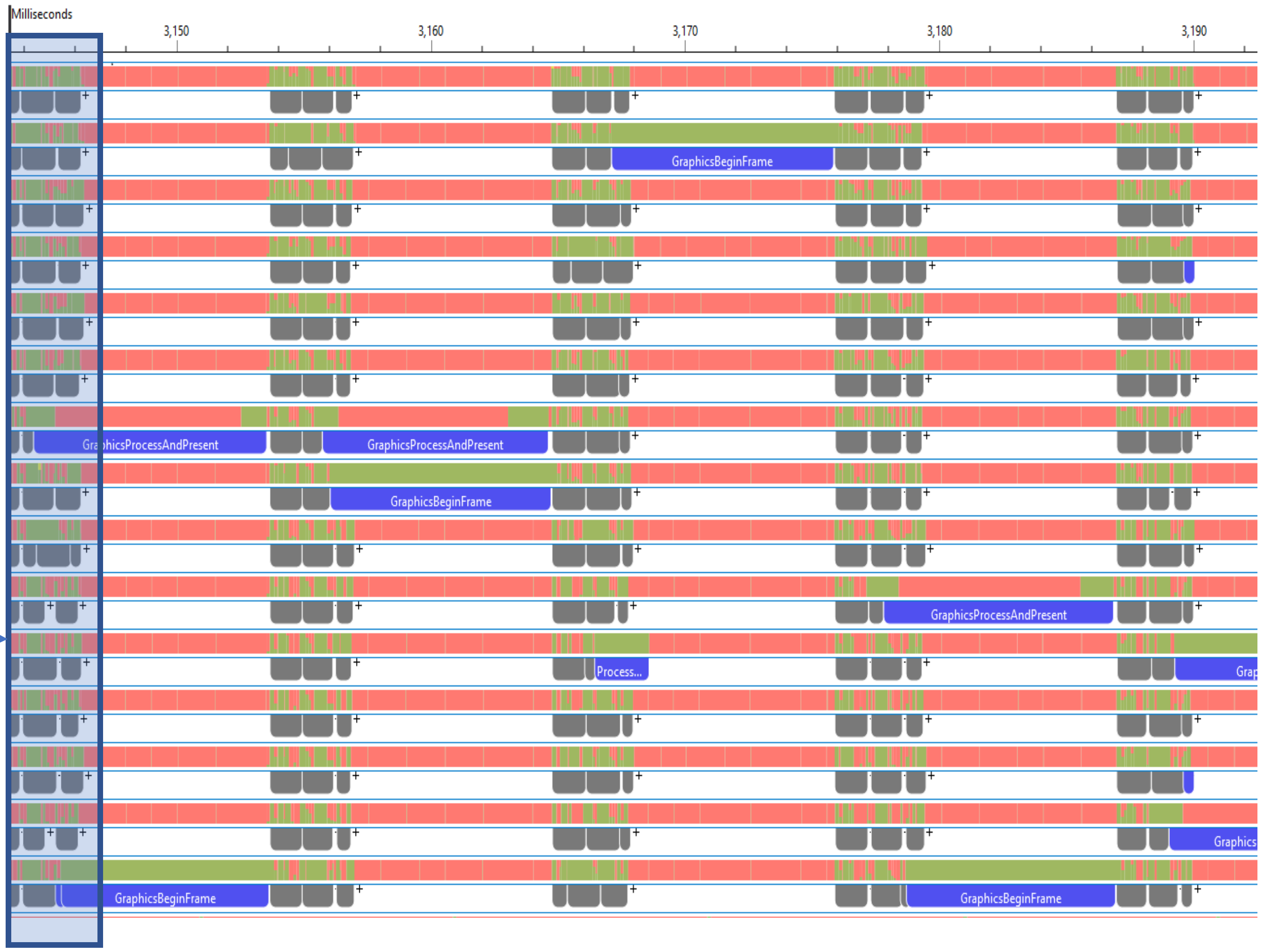


Dan Baker Oxide Games

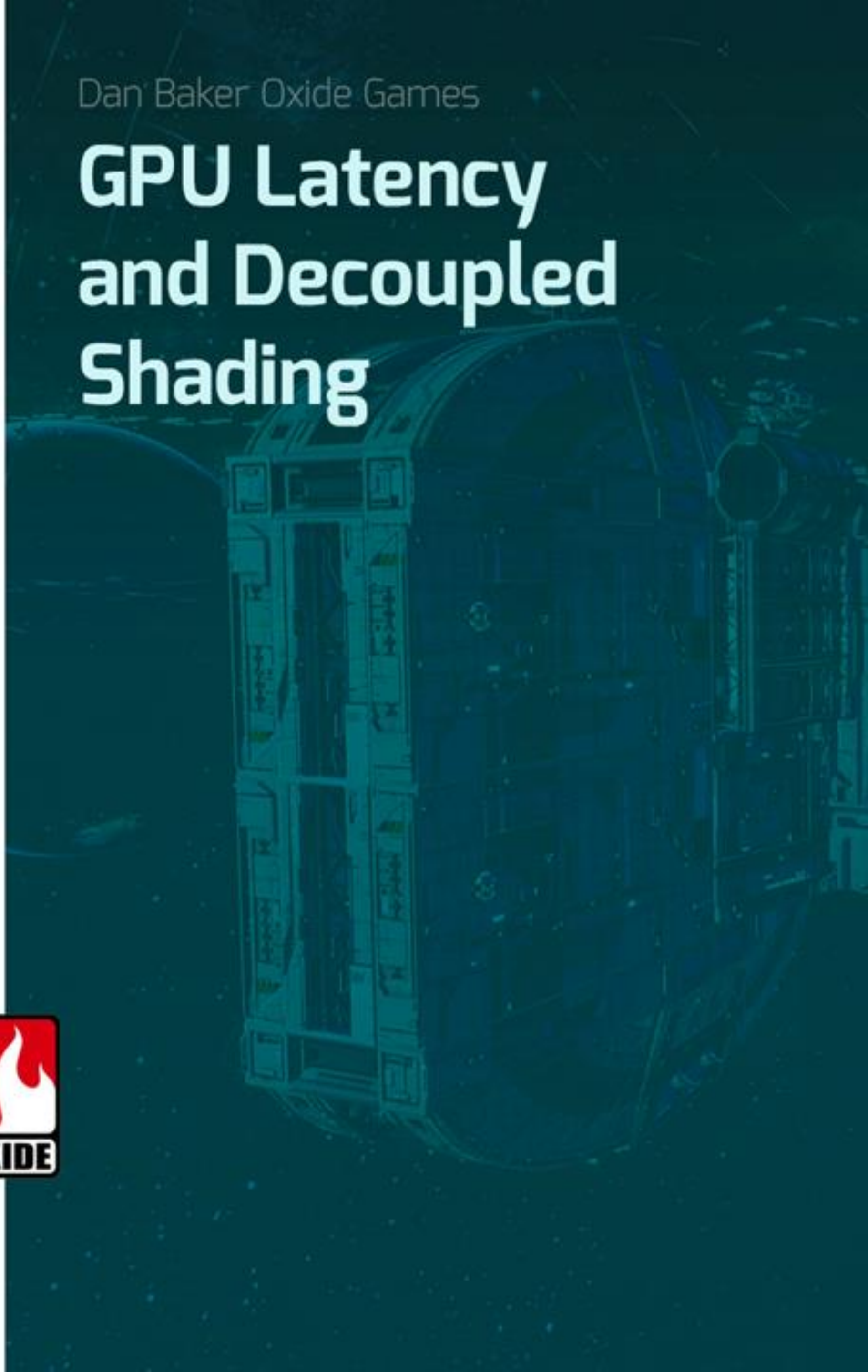
Efficient use of modern CPU - multicore

- The more cores you have, the faster a frame can be made
- Latency is reduced = super critical for VR

On 16 cores, entire Frame can be processed in just a few MS



- The current way: Generate 2 eyes, 90 fps
  - Lots of waste, lots of pixels to shade
  - Techniques get complex trying to reduce shading, e.g. foveated rendering
  - Must be very careful about all sorts of aliasing, especially shader aliasing and eye to eye ‘exactness
  - If intend to use whole GPU, end up adding 11 ms of latency
- Is there a better way?
  - Can we shade less frequently?
  - Can we share shading work between the eyes?
  - Can we guarantee that each eye has the same shading data?
  - Can we do better about not dropping frames?



Dan Baker Oxide Games

GPU Latency and Decoupled Shading

Dan Baker Oxide Games

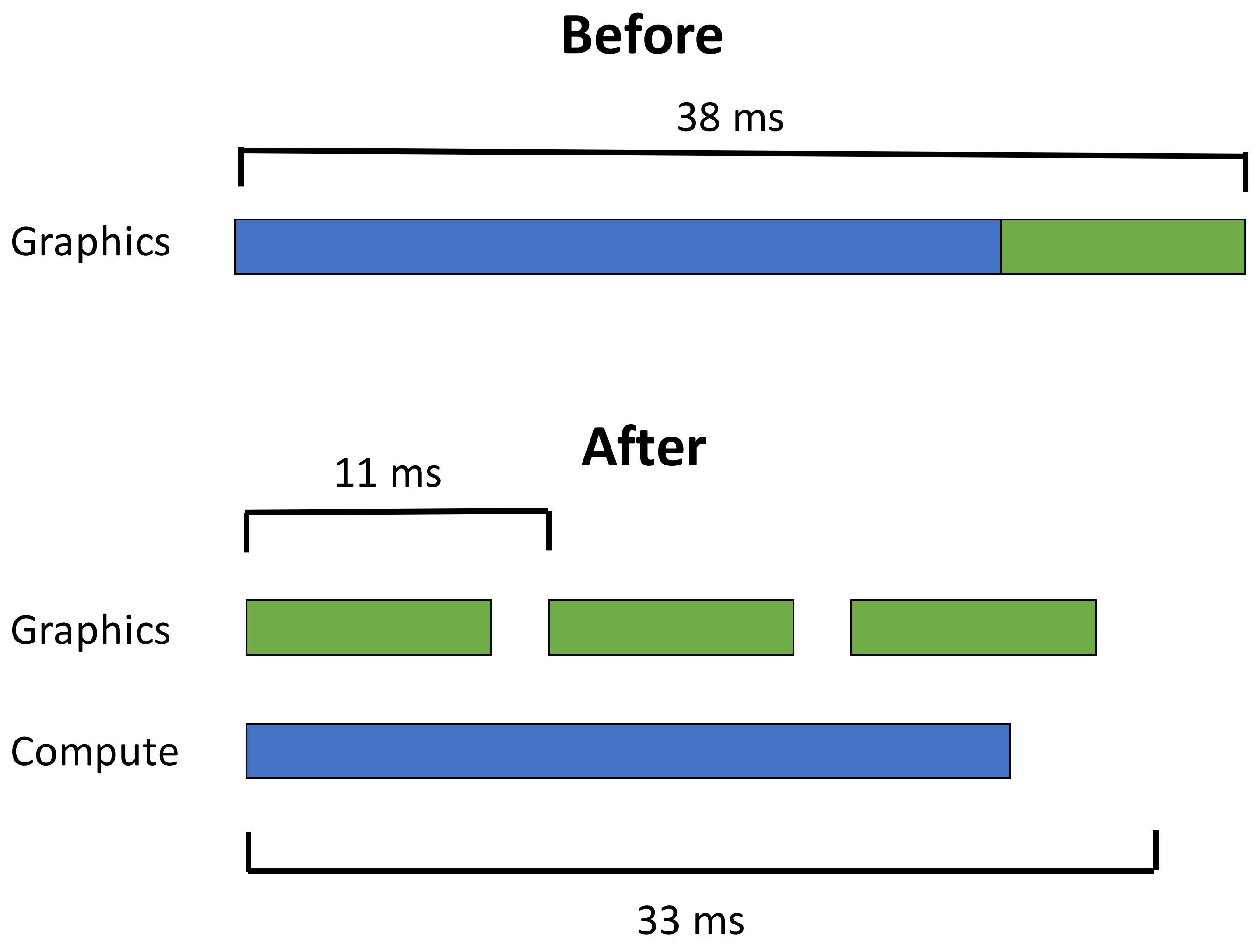
Object Space, a better way of doing VR

- Core concept – shade once, at reduced FPS (e.g. 30 fps) and share data between the eyes
- Aliasing, performance, eye coherency, all better

Dan Baker Oxide Games

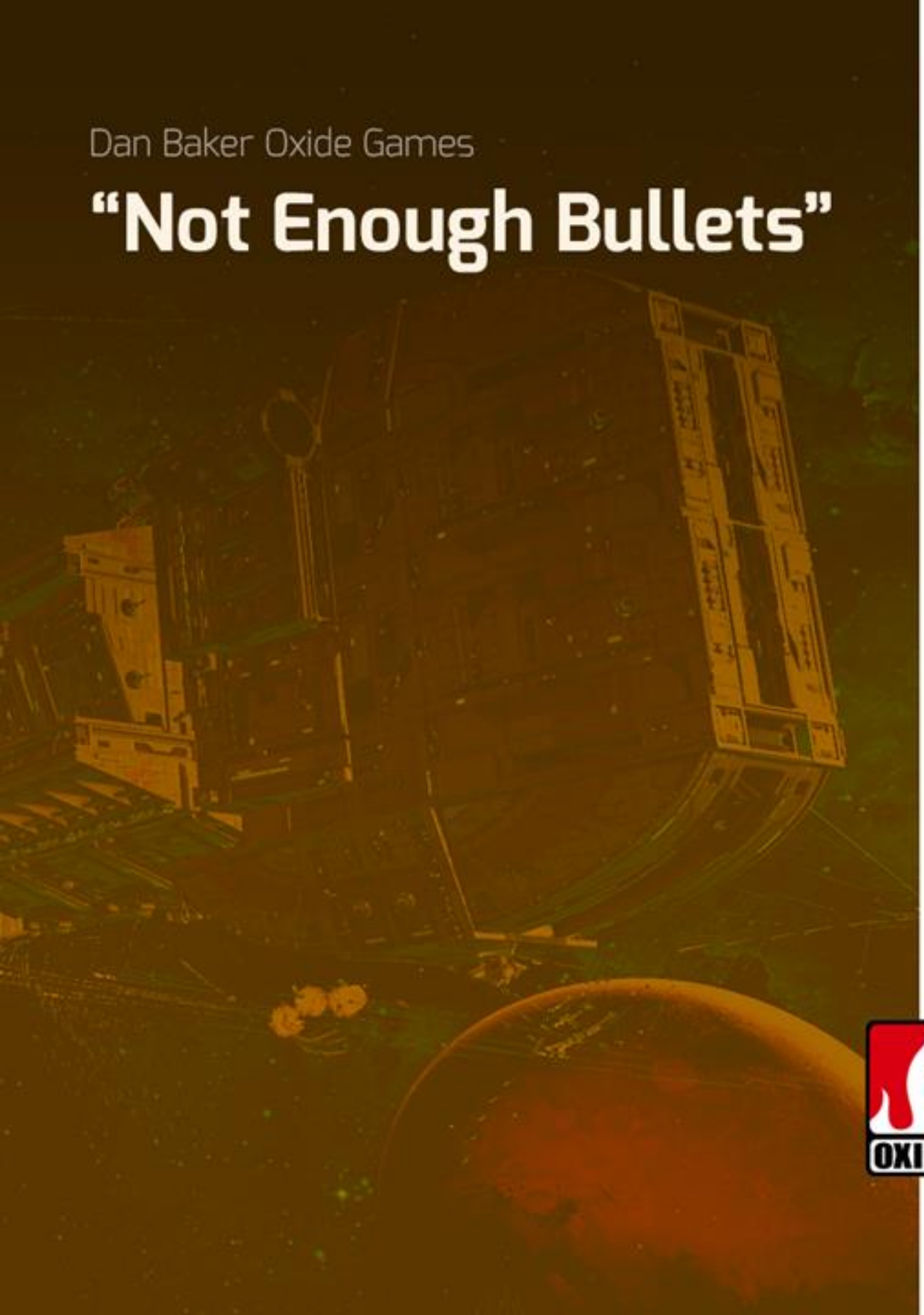
Async compute to the rescue

3X Performance Increase



Dan Baker Oxide Games

“Not Enough Bullets”



- Small VR game/demo based on Nitrous 2.0
- Used as our prototype for Nitrous 2.0 concepts
- Space VR game with thousands of star fighters and huge capital ships
- Called Not Enough Bullets in reaction to the sheer chaos of space battle!

Dan Baker Oxide Games

Improved Latency

- VR tracking reduced to only the rasterization portion - typically < 50% of GPU resource
- Thus, can shave off ~5-6 ms latency
- High POPS + Decoupled Shading = App Motion to Photon Latency

Dan Baker Oxide Games

Conclusion

- Next gen APIs benefits:
  - Decoupled shading can be supported natively
  - CPU overhead reduced
  - Multiple cores can be effectively used
  - Strict scheduling can guarantee when work will be done
- “Not Enough Bullets” demo shows all this in action!