Optimizations Enhance Halo Wars* 2 for PCs with Intel Integrated Graphics

The Mission

When top UK-based studio Creative Assembly* began their ambitious work on *Halo Wars* 2*, they wanted the game to run on a variety of settings supported by DirectX* 12, and to be playable up and down the hardware ladder—including advanced desktop PC configurations and laptops. While many of the optimizations also enhanced the game for high-end systems with discrete graphics cards, this white paper will explore the team's efforts for Intel integrated graphics and multicore processing functions.

Extending a Franchise

First-person shooter *Halo** is one of the most popular franchises in PC gaming history; it started in 2001 with *Halo: Combat Evolved*, one of the original Xbox* launch titles. By late 2015, *Halo* had generated over USD $5 billion in lifetime game and hardware sales. Development work for *Halo Wars 2* was performed by Creative Assembly, veterans of *Alien: Isolation** and *Total War: Warhammer**. They were experienced at producing games for multiple platforms; but, with DirectX 12 still maturing, the team faced challenges with their engine, the DX 12 driver, multicore efficiency, and more.

Figure 1. Halo Wars* 2 is the latest addition to the Halo* universe.

Switching the game to an RTS title meant tracking more units and packing the interface with crucial statistics, while updating the “mini-map” constantly. *Halo Wars 2* also introduces “Blitz” mode—a new, innovative and action-packed twist on RTS gameplay that combines card-based strategy with explosive combat. The new mode also streamlines most of the traditional RTS systems, such as base-building, skills development, and resource management.

Expanding the User Base

Michael Bailey, Lead Engine Programmer at Creative Assembly, was one of the principal developers involved.

“Given the nature of RTS games, unit counts and large-scale battles with plenty of VFX were always a big focus,” he explained. “This, combined with a deterministic networking model common to RTS games, put restrictions on how scalable we could be, so we pushed from the start to ensure we had a good base to make use of multiple CPU cores. Our primary goal here was to ensure the game ran on a wide range of hardware, and the Microsoft Surface* Pro 4 laptop was a natural to target, aiming for 30 fps and still looking great.”

Figure 2. Dazzling particle effects keep the screen display busy.

Excellence across a wide spectrum of hardware was key. There is a temptation to develop early versions of a game to initially fit the specs for a high-end desktop system, taking full advantage of 10 teraflops of GPU potential. But this makes later scaling the game to Xbox One* and Ultrabook™ devices tricky, as early design choices could limit optimization options. With top-end Ultrabooks and laptops nearing the Xbox One in power and performance, those devices are now considered mainstream. That means developers can no longer concentrate on either CPU or GPU optimizations—both are crucial.

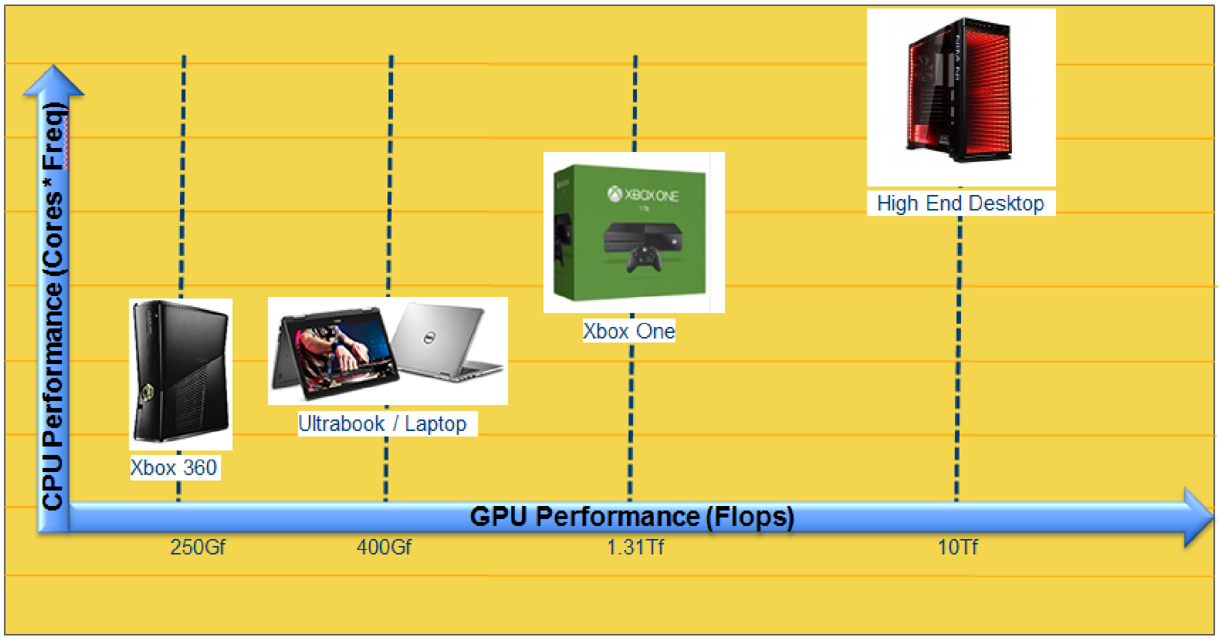

Figure 3. Getting the game to play on Ultrabook™ devices and laptops was a key task.

Bailey had not specifically optimized previous games for Intel® HD graphics, but working directly with Intel allowed him to get up to speed quickly. Fortunately, the graphics systems used by Creative Assembly were designed to be scalable without hurting the overall look and feel. Intel engineers talked Bailey's team through their tools, on site, to diagnose issues and measure performance for added features. As the DX 12 version became more mature, areas for improvement were pinpointed. This included testing the performance of the game across multicore machines to ensure acceptable decent scaling.

The Game is On

In the early stages of the project, the team encountered several issues:

- Instability on the target system
- Speed issues at low settings on a discrete GPU
- Severe I/O lag
- Tools failure on the Universal Windows Platform (UWP) and DirectX 12
- Driver Issues
- Corruption due to buffers being reused before completion

None of the problems turned out to be show-stoppers, but the early screenshots revealed that there was a lot of work to do.

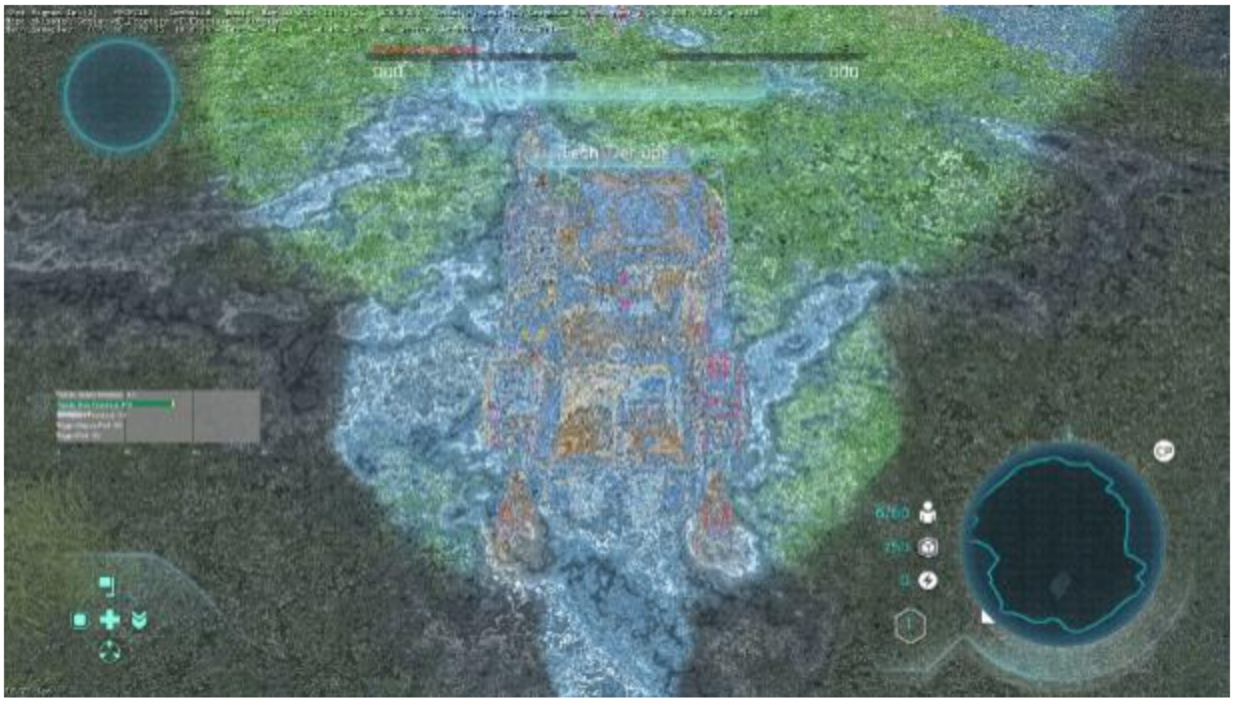

Figure 4. Early screenshots revealed corruption on a wide scale, affecting terrain, units, and the "mini-map."

Multicore Optimizations

The team worked on several different areas to optimize the game's multicore support:

- Improve algorithms
- Reduce memory allocations
- Streamline assets
- Perform low-level optimization
- Enhance parallelization

In multiplayer battles, there were challenges ensuring that the simulation—spread over multiple threads—remained deterministic. On the CPU side, the team concentrated on being able to split the simulation while ensuring determinism. In addition, specific to DirectX 12, they worked to minimize resource barriers and redundant work.

The team found that changing the order of something meant there was a chance that they could end up with each client diverging and having a different representation of the *Halo Wars 2* world. If two or more clients disagreed on a checksum, for example, the result was a “desync” that caused a player to be kicked out. One cause was a race condition where the output is dependent on the sequence or timing of other uncontrollable events. The team soon learned what calculations could put be put onto other threads, when they could run, and when it was safe to do that.

They eventually reached a stage where the CPU side was efficiently running across multiple threads, well apart from the render thread. Their work on multicore optimization will be presented at the 2017 Game Developers Conference (see link, and list below.

GPU Optimizations and Diagnostics

The Creative Assembly team performed multiple GPU optimizations, working closely with Intel® Graphics Performance Analyzers (Intel® GPA). These are powerful, agile tools which enable game developers to utilize the full performance potential of their gaming platform, including Intel® Core™ processors and Intel® HD Graphics, as well as Intel® architecture-based tablets running the Android* operating system. Intel Graphics Performance Analyzers visualize performance data from the application, enabling developers to understand system-level and individual frame performance issues, as well as allowing “what-if” experiments to estimate potential performance gains from optimizations.

The Creative Assembly team used new features of Intel GPA Monitor to launch and profile the game in action. They also examined problematic frames with Intel GPA Frame Analyzer to identify graphics “hot spots” in a particular scene.

Intel GPA tools also assisted investigations into corrupt terrain tiles, which the team tracked down to an error with descriptor tables. In addition, they investigated performance bottlenecks and judged performance versus quality tradeoffs for code that controlled terrain tessellation, as well as texture resolution bandwidth bottlenecks.

Bailey and the team also used Microsoft GPUView and other internal game profilers. The team scaled down their graphical options in the following areas:

- Disabling Async Compute when it ended up using too many synchronization primitives to guard between multiple command lists. (This slowed things down on hardware where Async Compute isn’t natively supported.)

- Reducing the terrain tessellation amounts for quality versus performance scalability reasons; on smaller screens, the extra detail afforded by tessellation wasn't worth the cost.
- Dropping the terrain compositing tiled resource map sizes down to match the resolution the game was running in. This helped to avoid compositing terrain textures at resolutions that were too high for the display.
- Correcting the pixel shader system. The team had assumed that as the pixel shader wasn't outputting anything within their shadow-rendering Pipeline State Objects (PSOs), it would be stripped along with interpolators from the Vertex shader. Intel GPA revealed that wasn't the case.
- Tweaking the small-object culling factor to account for resolution (both in the main scene and in the shadows where the shadow resolution was much smaller).
- Swapping the heavy shadow PCF filter kernel to use GatherCmp to massively reduce the cost of that shader with no visual difference.
- Removing redundant terrain composition layers (a bug was causing them to export all layers, even unnecessary ones).
- Implementing a dynamic particle-reduction system which prioritized battle visual effects and dropped less important "incidental" environmental effects, or heavier "high-end" effects such as particle lights.

Players can identify units by what visual effects they saw, or what they look like against the terrain. The optimizations Bailey and his team implemented had to keep the display readable without decimating the frame and disabling certain effects. "If the scene gets too heavy, we start ramping down the particle effects," he said.

They also ensure that the battle effects look amazing. When there were 20 explosions on top of each other, they started turning off environmental effects and the less important visual effects. "Those are based upon on the current particle system load," he explained. "On the Microsoft Surface Pro 4, they're probably dropped down a bit more than they would drop down on a higher-end system which can handle more of a load. You get the environmental effects staying on, but they're not really a big part of the battle."

Figure 5. Big battles contain multiple units, plenty of particle effects, and lots of information to process.

DirectX 12 Challenges

Creative Assembly's technical base is separate from the *Total War: Warhammer* engine, and the graphics layer is brought over from *Alien: Isolation*. They wrote the DX 12 graphics layer, building on top of existing DX 11 support, to take advantage of newer features and ensure it was more optimal than DX 11.

When they started, the graphics layer was targeting DX 11 for PC, so planning for and moving to DX 12 exclusively gave them plenty of opportunities to maximize performance. Bailey discovered during development that they were "over-cautious" with fences and descriptor table invalidations to ensure correctness and stability. They studied several DX 12 talks at GDC 2016 and gathered more information from online sources, until they understood the best way to take advantage of it in their engine.

As the system became more mature and stable, they started to optimize to ensure they removed redundant descriptor settings and unnecessary resource barriers and fences. This, combined with render pass changes to track the resource dependencies better, meant that they were able to shift some of their command-list building passes, such as shadows, to run in parallel with other areas and make more use of multiple cores. The command lists they moved to run in parallel were heavy on the CPU side due to the number of draw calls. These passes increased the cost with the number of units fighting on-screen, so they benefitted greatly from this optimization during heavy battle sequences. Their DX 12 version soon became faster than DX 11 for CPU-side rendering, even single-threaded.

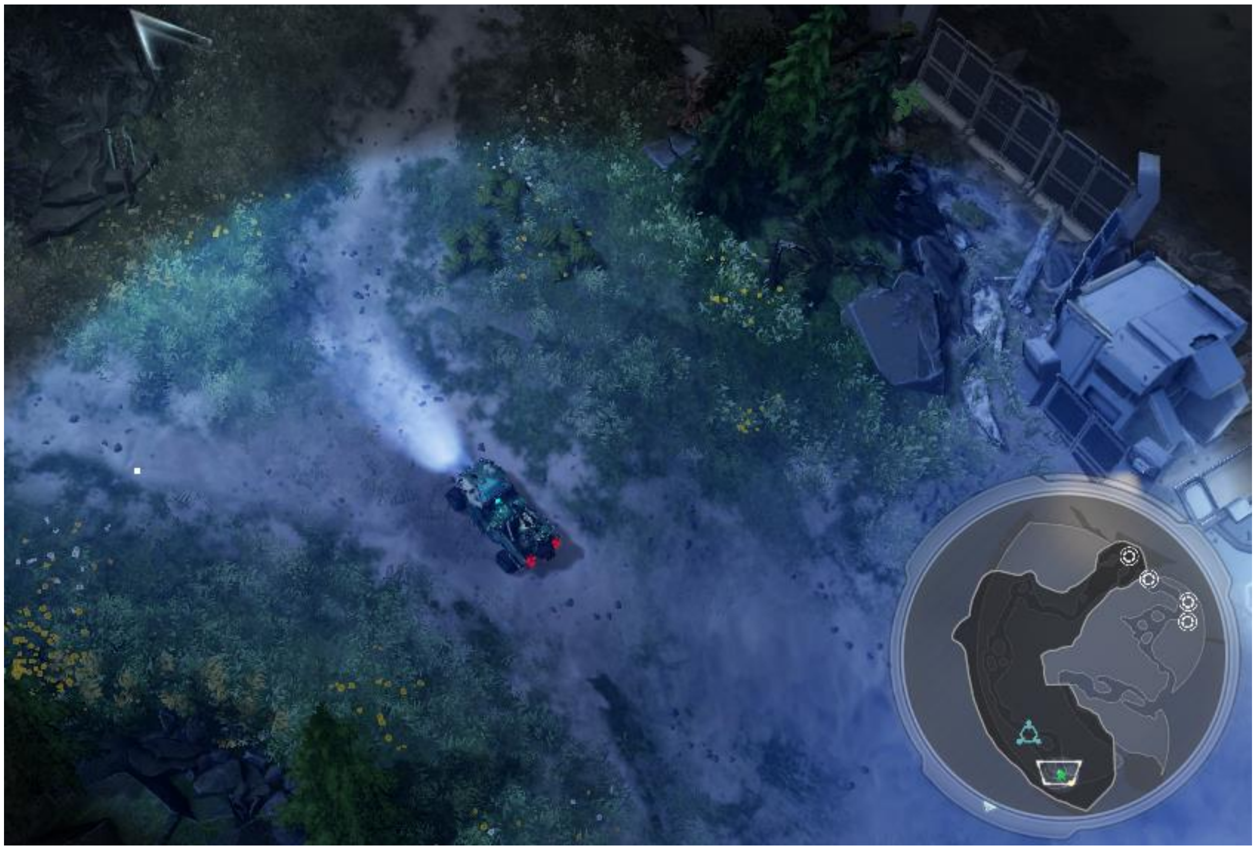

Figure 6. In this screenshot, the "mini-map" in the bottom right corner is fully functional.

The trade-off was that the DX 12 environment was very new: enabling DX 12 validation layers to get diagnostic warnings and errors was not yet mature. The documentation was incomplete, or inaccurate. When things went wrong, Bailey's team wasn't sure if it was their fault, or if the driver was to blame.

Eventually, they were developing only on DX 12 and advancing to a single low-level graphics API gave them more control over the performance trade-offs they could make. This allowed the team to target a large number of Windows* 10 devices with a single code-path. The beta released in January 2017 supports DirectX 12 down to feature level 11.0; the game's terrain texture compositing code requires "Tiled Resources Tier 1" support.

Challenges of Evolving Software

Because there were so many new features they wanted to exploit in the game, the team had to continually update drivers, compilers, and Windows 10 versions. Fortunately, they were in direct contact with various hardware vendors, which meant they had access to beta drivers with specific fixes, as well as beta versions of their internal tools.

Over time, the drivers, tools, and validation layers became more mature, so when errors did occur they were usually from the game code and were easier to track down. The graphics testbed helped debug a wide set of DX 12 API functionality, from developing new features to working in a much more cut-down environment where they were in control of all the variables.

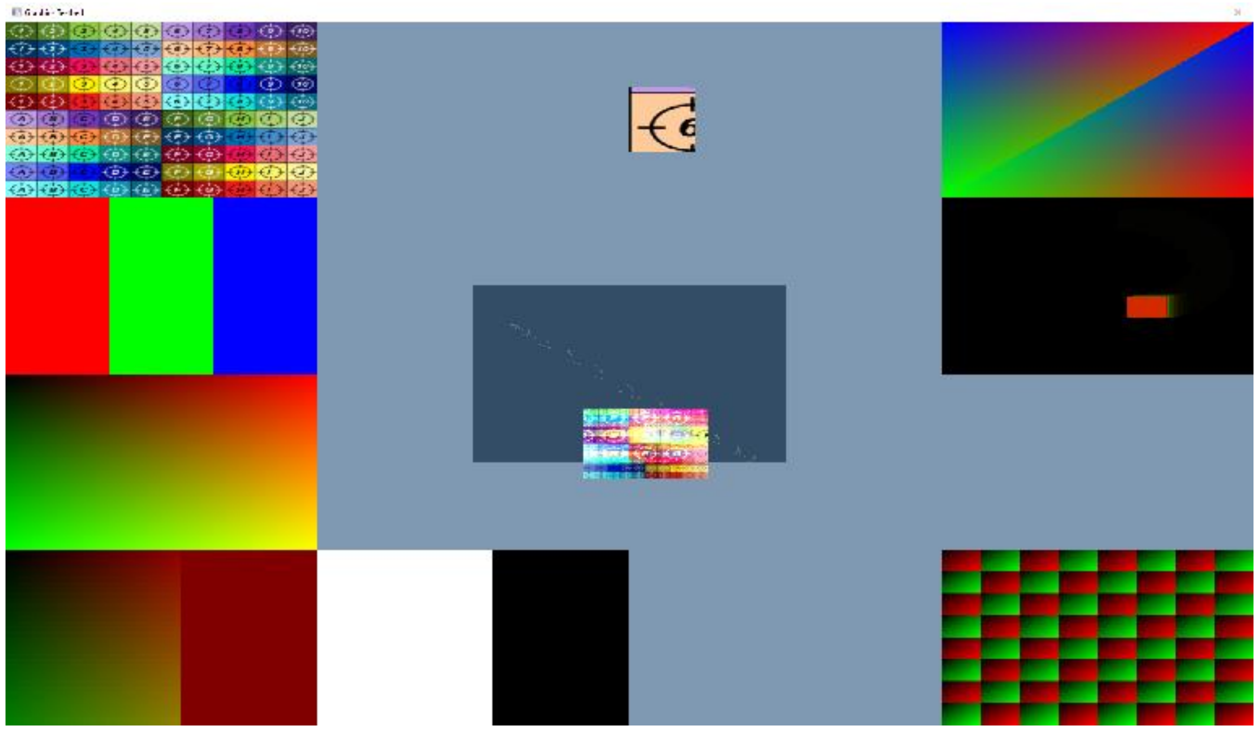

Figure 7. Cut-down environment for testing new features.

Advice for Developers

Reach Out to the Experts

Bailey advises developers to reach out to the Independent Hardware Vendors (IHVs) for key insights on problems. “They obviously want your game working well on their hardware, and they will give you help, as well as tools,” he said. “If you can get them to take an interest in your game, or you're taking interest in their hardware, then it's a mutually beneficial experience." Part of the payoff is access to early fixes and improvements, and optimization feedback, which Bailey's team used extensively.

"I don't think we ever got conflicting feedback," Bailey said. The vendors were helpful in identifying known issues, and, in some places, the game works slightly differently when it identifies hardware from Intel, AMD*, or NVIDIA*. "They've all got their own slight differences, but talking to them is how you find out what they're good at," Bailey said. Overall, the result was a faster game, especially on the lower-end hardware.

Create a Robust Test Lab

The Creative Assembly engine team has six people attacking problems and testing solutions in a variety of ways. They often borrow machines as needed and rely heavily on remote debugging to a second machine or Surface Pro 4 at their desk—this makes working on a touchscreen device much simpler. "I'm not sitting there trying to use the touchscreen to try and get it to fit on the screen," Bailey said. "I can comfortably use my big desktop to remote debug."

Microsoft owns the rights to the *Halo* franchise, so Creative Assembly had access to Microsoft's compact testing lab through that relationship. It was a gateway to all sorts of different varieties of hardware, some below the minimum specs. Being able to regularly test builds increased their coverage and ensured that Creative Assembly's optimizations were thoroughly checked.

Share Knowledge

Bailey encourages developers to reach out and participate within the community, especially by attending events such as the Game Developers Conference (GDC). If that's not possible, Bailey suggests "downloading the talks to catch up on them later."

Previously, there were only a few DirectX 12 talks. This year, there will be more tips and tricks shared by experienced teams. "You want to share your successes as well," Bailey said. "If you share your success, then people will come and talk to you about how they did something differently."

It's like a multi-threaded attack on a problem, but with people instead of processors. "When you share these things, you invite the conversation," Bailey explained. "And inviting that conversation means you might find out a better way of doing something.

Maybe your approach was pretty good already, but things can always get better and better."

Additional Resources

Intel Graphics Performance Analyzers:
https://software.intel.com/en-us/gpa

Intel Multicore FAQ:
https://software.intel.com/en-us/articles/frequently-asked-questions-intel-multi-core-processor-architecture/

Introduction to the Universal Windows Platform:
https://docs.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide

DirectX 12 Installer:
https://www.microsoft.com/en-us/download/details.aspx?id=35

Microsoft Surface Pro 4:
https://www.microsoftstore.com/store/msusa/en_US/pdp/productID.5072641000

CPU and GPU Optimization Talk at GDC 2017:
http://schedule.gdconf.com/session/threading-your-way-through-the-tricks-and-traps-of-making-a-dx12-sequel-to-halo-wars-presented-by-intel