Developing for Intel® Graphics: Today and Into the Future

Kyle Grau
Agenda

- Current Intel® Graphics and Trends
- General Detection of Features for DirectX® 12 and Vulkan®
- SIMD on Intel
- Prepare for Upcoming Features

* other names and brands may be claimed as the property of others
Intel® Graphics Increasing in Performance and Capabilities
GPU Detection for Features

- GPU hardware is constantly changing with each generation
- One generation of hardware from a vendor may now support new features

- For DirectX* 12 and Vulkan*, query the hardware for feature support using defined APIs
  - For DirectX* 12, check feature support with ID3D12Device::CheckFeatureSupport
  - For Vulkan* use vkPhysicalDeaviceFeatures and check for proper extension support with vkEnumerateInstanceExtensio
    nProperties

- Avoid using vendor IDs to disable features, use slower execution paths, or defaulting to low settings.

- Always check to see if the hardware detected supports the features needed and meets the technical requirements for your game

- Do use features keeping in mind architectural differences
Understanding Intel’s Driver Versioning

- Our driver build number is the last 7 digits of the driver version. Check these numbers if there is a reason you need specific driver support from Intel.

- If you have legacy code that is only checking the last 4 digits, please update your code to check the last 7 digits to ensure your game will run on Intel.

EU SIMD Explained

Support for 1/2/4/8/16 or 32-wide instructions

- Higher than SIMD8 instructions pair adjacent registers
- SIMD16 would pair 2 physical registers to a single logical 64B register

Compiler makes the decision:

- VS/DS/HS/GS: SIMD8
- PS/CS: SIMD8/16/32

Performance tip – Reducing register pressure allows:

- Higher SIMD
- Better latency hiding
- Better instruction pipelining

- Reduced spills
- Better codegen
How To Reduce Register Pressure

Don’t:
- Branch on constant buffer conditions
- Non uniform access to buffer data
- Excessive variable decl. (esp. arrays)

Do’s:
- Use partial precision
- Move common code outside branches
- Specialization constants / #define

GRF Usage:
- <64: SIMD16
- 64-128: SIMD8
- >128: SIMD8 with Spills
SIMD Key Takeaways

- Reduce register pressure whenever possible
  - Better SIMD width
  - Better latency hiding
  - Better instruction pipelining
  - Reduced spills and fills
  - Better codegen

- Do not make assumptions about SIMD lane counts
  - Use GetWaveSize() and similar wave intrinsics to get wave count. Swizzle operations on one hardware vendor may fail on another
  - Race conditions can happen when SIMD is different than thread group size. Use barriers to ensure proper read/write access to memory
  - If thread groups are independent and do not rely on other thread groups, avoid barriers as they introduce unnecessary waiting conditions
Adaptive Sync

- Supported since Gen11 (Ice Lake graphics)
- Enable to relieve screen tearing and stuttering on displays that support it

Requirements
- Monitor that supports VESA adaptive sync display
- User also has to enable it with Intel Graphics Control Panel

For DX12
- Use DXGI_SWAP_CHAIN_ALLOW_TEARING and DXGI_PRESENT_ALLOW_TEARING

For Vulkan*
- Use VK_PRESENT_MODE_IMMEDIATE_KHR or VK_PRESENT_MODE_FIFO_KHR
High Dynamic Range Support

**DirectX* 12**

- Swap chain must use `DXGI_SWAP_EFFECT_FLIP_SEQUENTIAL` or `DXGI_SWAP_EFFECT_FLIP_DISCARD` and recommended to use `DXGI_FORMAT_R10G10B10A2_UNORM`
- Must explicitly use `IDXGISwapChain3::SetColorSpace1` method to set color space to `DXGI_COLOR_SPACE_RGB_FULL_G2084_NONE_P2020`
- Use `DXGI_OUTPUT_DESC1` to get information about supported color spaces, color information, and luminance values to adjust tone mapping in post processing
Queue Support

**Multiple queues with hardware support can support asynchronous compute on GPU.**

- Allows the creation of separate command lists for different tasks.
  - One queue for render work, another for compute shader tasks, and another for copy operations.
  - Still require necessary synchronization if there is a dependency across queues. (semaphore)

**For DX12:** If hardware has queue support, creating queues for compute and submitting compute command lists on that will enable async compute.

**Copy Engine**
- For Vulkan*: Use `vkGetPhysicalDeviceQueueFamilyProperties` to enumerate queue families and create `vkQueue` on appropriate queue family.

**Media Engine**
- For compute-only work that would benefit from async compute, create on non-graphics work queue. Always profile to see if there is benefit using async compute.

- Avoid overlapping compute work in both the graphics and compute queues.
Ray Tracing Support

- Supported with dedicated hardware via DirectX* 12 and Vulkan*

- Early Guidance
  - Use TraceRay over inline ray queries
  - Use indexed meshes for BVH builds
  - Batch acceleration structure build operations
    - Do not interleave barriers, do them all in one command list and barrier at the end
Mesh Shading

2 shader stages to replace legacy geometry pipeline for a compute-shader-like approach for generating geometry.

- Allows for transformation, culling, and generating geometry in small batches without fixed functions.
- Run in SIMD8/16 by default.
- Hardware allocates for the worst-case scenario.
- Big meshlets can lead to lower efficiency.
Variable Rate Shading

- Allows developers to increase visual quality while maintaining frame rate
  - Pixels not adding to visual fidelity can have reduced shading rate

- DirectX* 12:
  - Tier 1: Per draw/per primitive
  - Tier 2: Allow control of shading rate based on image

- Vulkan*:
  - Supported via VK_KHR_fragment_shading_rate
    - For features, check feature support for per draw, per primitive, and for image based with VkPhysicalDeviceFeatures
Call to Action

- Use GPU detection code to help guide enabling of features for Intel
  - Try to avoid disabling features based on vendor ID; future hardware may support these capabilities

- Variable SIMD means lane count can vary based on graphics compiler choice
  - Use GetWaveSize() and similar wave intrinsics to get wave size. Swizzle operations on one hardware vendor may fail on another
  - Design your shader algorithms to work with any SIMD width

- Check available command queues

- Be aware of the new guidance from Intel on checking Intel Graphics Driver versions
  - Current guidance is to check last 7 digits of the driver version to get full build number

- For Vulkan* KHR extensions, check for supported sizes and limits

- Use DirectX* and Vulkan* APIs for:
  - Adaptive sync
  - HDR
  - Ray Tracing
  - Mesh Shading
  - Variable Rate Shading

- Ensure middleware is using right features as well
Legal Notices and Disclaimers

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim hereafter drafted which includes subject matter disclosed herein.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at intel.com.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Performance varies by use, configuration and other factors. Learn more at www.intel.com/PerformanceIndex.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates.

Your costs and results may vary.

Optimization Notice: Intel's compilers may or may not optimize to the same degree as competitors' compilers. Optimization depends on the compiler settings and the characteristics of the input and output files, and is not intended for use in production environments. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

* other names and brands may be claimed as the property of others