TURN YOUR FRAME RATE UP TO 60 WITH VULKAN ON THE NINTENDO SWITCH

HOW WE BROUGHT GALAXY ON FIRE TO THE SWITCH AND MADE IT THE BEST VERSION
TARGET PLATFORM SWITCH

- ✔ OS stuff (threading, memory, file IO, etc.)
- ✔ Input + rumble
- ✔ Savegames
- ✔ Player profiles
- ✔ Networking
- ✔ Video playback
- ✔ 3rd party SDKs
VULKAN 1.1 SUPPORTED
PLAY MODES

vkDeviceWaitIdle();
vkDestroySwapchainKHR();
vkCreateSwapchainKHR();

for () {
    vkDestroyImageView();
    vkDestroyFrameBuffer();
}
// Create image
imageCreateInfo.samples = VK_SAMPLE_COUNT_4_BIT;

// Create render target
renderPassCreateInfo.attachmentDescription.samples = VK_SAMPLE_COUNT_4_BIT;

// Pipeline creation
pipelineMultisampleStateCreateInfo.rasterizationSamples = VK_SAMPLE_COUNT_4_BIT;

// Have to resolve at some point
vkCmdResolveImage();

// Or via
subpassDescription.pResolveAttachments = ...;

// Sample shading
pipelineMultisampleStateCreateInfo.sampleShadingEnable = VK_TRUE;
pipelineMultisampleStateCreateInfo.minSampleShading = 0.0f - 1.0f;
ANISOTROPIC FILTERING

samplerCreateInfo.anisotropyEnable = VK_TRUE;
samplerCreateInfo.maxAnisotropy = 1.0f - 16.0f;
GETTING STARTED

- Compare
- Profile (before)
- Profile (after)
- Change
NON-VULKAN

• Data-oriented design
• Not everything must be turned up to 60
• Reduce quality based on distance
• Auto reduce quality
DATA-ORIENTED DESIGN
NOT EVERYTHING MUST BE TURNED UP TO 60
AUTO REDUCE QUALITY

100 Drawcalls/culled: 1007/291 Frame: 43791
6 Drawcalls/culled: 870/289 Frame: 44519

lod_bias=0
Display: 0.079 ms

lod_bias=21
Display: 0.077 ms

GPU -2 ms
Submit 17 → 4 ms
VULKAN

• Vulkan Performance
• Basics
• Measuring
• DescriptorSet Caching
• VK_NV_dedicated_allocation
• Sorting draw calls
• Mo' parallelization, mo' problems
VULKAN PERFORMANCE

~1400 draw calls [ms]

- Nexus 5X: 71.4 ms (OpenGL ES 3), 26.4 ms (Vulkan)
- Galaxy S7: 32.1 ms (OpenGL ES 3), 11.3 ms (Vulkan)
- Google Pixel XL: 33.1 ms (OpenGL ES 3), 8.2 ms (Vulkan)
VULKAN PERFORMANCE BASICS CHECKLIST

✅ No vkDeviceWaitIdle() or vkQueueWaitIdle()

✅Textures in

    VK_IMAGE_TILING_OPTIMAL

    VK_MEMORY_PROPERTY_DEVICE_LOCAL_BIT

✅ Group DescriptorSets by frequency
MEASURING

// 1. Setup
vkCreateQueryPool(...);

// 2. At beginning of command buffer
vkCmdResetQueryPool(commandBuffer, queryPool, firstQuery, 2);
vkCmdWriteTimestamp(commandBuffer, VK_PIPELINE_STAGE_TOP_OF_PIPE_BIT, queryPool, query);

// 3. At end of command buffer
vkCmdWriteTimestamp(commandBuffer, VK_PIPELINE_STAGE_BOTTOM_OF_PIPE_BIT, queryPool, query + 1);

// 4. Get results
uint64_t timestamps[2];
res = vkGetQueryPoolResults(device, queryPool, firstQuery, 2,
   sizeof(uint64_t) * 2, timestamps, sizeof(uint64_t), VK_QUERY_RESULT_64_BIT);

// in ms
float timeInMs = (timestamps[1] - timestamps[0]) / 1000.0f;
// Uniforms:
vkCmdBindDescriptorSets(commandBuffer, VK_PIPELINE_BIND_POINT_GRAPHICS, layout, firstSet, 1, &uniformSet, 1, &dynamicOffset);

struct DescriptorSetCacheEntry
{
    VkDescriptorSet set;
    uint32_t lastFrameUsed;
};

HashMap<Hasher::Value, DescriptorSetCacheEntry> descriptorSetCache_;

// If found in cache:
vkCmdBindDescriptorSets(commandBuffer, VK_PIPELINE_BIND_POINT_GRAPHICS, layout, firstSet, 1, &textureSet, 0, nullptr);

// If not found in cache:
vkAllocateDescriptorSets(device, &allocateInfo, &textureSet);
vkUpdateDescriptorSets(device, writeCount, writeTextures, 0, nullptr);
vkCmdBindDescriptorSets(commandBuffer, VK_PIPELINE_BIND_POINT_GRAPHICS, layout, firstSet, 1, &textureSet, 0, nullptr);
// Enable extension on vkCreateDevice()
VK_NV_DEDICATED_ALLOCATION_EXTENSION_NAME

// Use it for an allocation
VkDedicatedAllocationMemoryAllocateInfoNV dedicatedInfo = {};
dedicatedInfo.sType = VK_STRUCTURE_TYPE_DEDICATED_ALLOCATION_MEMORY_ALLOCATE_INFO_NV;
dedicatedInfo.pNext = nullptr;
dedicatedInfo.image = image;
dedicatedInfo.buffer = VK_NULL_HANDLE;

VkMemoryAllocateInfo allocateInfo = {};
allocateInfo.pNext = &dedicatedInfo;
SORTING DRAW CALLS
MO’ PARALLELIZATION, MO’ PROBLEMS
MO’ PARALLELIZATION, MO’ PROBLEMS

Gameplay → Render prep → Render
MO’ PARALLELIZATION, MO’ PROBLEMS
MO' PARALLELIZATION, MO' PROBLEMS

- CPU 0: Gameplay
- CPU 1: Render prep
- CPU 2: Render
- GPU: Swapchain 0, Swapchain 1, Swapchain 2

Timing:
- 16 ms each step
MO’ PARALLELIZATION, MO’ PROBLEMS
MO’ PARALLELIZATION, MO’ PROBLEMS

<table>
<thead>
<tr>
<th>CPU 0</th>
<th>Gameplay</th>
<th>Render prep</th>
<th>Render</th>
<th>Swapchain 0</th>
<th>Swapchain 1</th>
<th>Swapchain 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU 1</td>
<td>Gameplay</td>
<td>Render prep</td>
<td>Render</td>
<td>Swapchain 0</td>
<td>Swapchain 1</td>
<td>Swapchain 2</td>
</tr>
<tr>
<td>CPU 2</td>
<td>Gameplay</td>
<td>Render prep</td>
<td>Render</td>
<td>Swapchain 0</td>
<td>Swapchain 1</td>
<td>Swapchain 2</td>
</tr>
<tr>
<td>GPU</td>
<td>Swapchain 0</td>
<td>Swapchain 0</td>
<td>Swapchain 0</td>
<td>Swapchain 0</td>
<td>Swapchain 1</td>
<td>Swapchain 2</td>
</tr>
<tr>
<td></td>
<td>Swapchain 1</td>
<td>Swapchain 1</td>
<td>Swapchain 1</td>
<td>Swapchain 1</td>
<td>Swapchain 1</td>
<td>Swapchain 2</td>
</tr>
<tr>
<td></td>
<td>Swapchain 2</td>
<td>Swapchain 2</td>
<td>Swapchain 2</td>
<td>Swapchain 2</td>
<td>Swapchain 2</td>
<td>Swapchain 2</td>
</tr>
</tbody>
</table>

16 ms  16 ms  16 ms  16 ms  16 ms  16 ms
MO’ PARALLELIZATION, MO’ PROBLEMS
MO’ PARALLELIZATION, MO’ PROBLEMS
LATENCY?

3 frames multi-threaded on CPU
+ 3 frames on GPU (swapchain)
= 6 frames latency

=> \(6 \times \left(\frac{1000}{60}\text{ ms}\right) \approx 100\text{ ms latency}\)
BENEFITS & DRAWBACKS
OF PARALLELIZATION

👍
• More time to do stuff
• Buffers for varying loads

⚠️
• Multiple copies of data
• Cannot delete data immediately
• Async feedback
• Increased latency
AND THEN THE WORST CASE HAPPENED

Kyyrk. He's really quite pleasant in his own way.
1. Switch is an easy target platform.
2. Vulkan fulfills the cross-platform promise.
3. Synchronize properly.
THANK YOU!

Questions?

Contact: j.kuhlmann@dsfishlabs.com
@j66k

Website: https://dsfishlabs.com/

© 2019 Deep Silver FISHLABS. Deep Silver FISHLABS is a wholly owned development studio of Koch Media GmbH / Planegg, which is a wholly owned subsidiary of Koch Media GmbH, Austria