Compute Shader 入门精要

Optimize your game using compute shader

凯奥斯

Compute Shader 入门精要

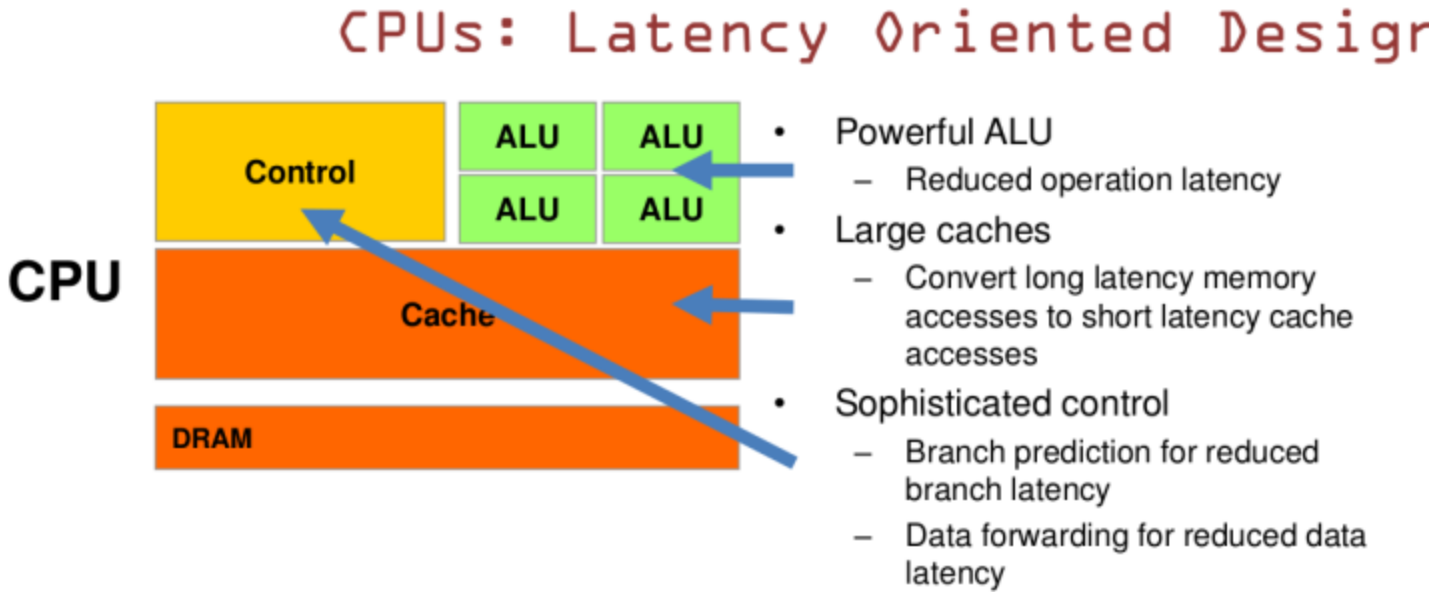
- 概念
- 语法
- 用途

概念

GPGPU（General Purpose Computing on GPU）

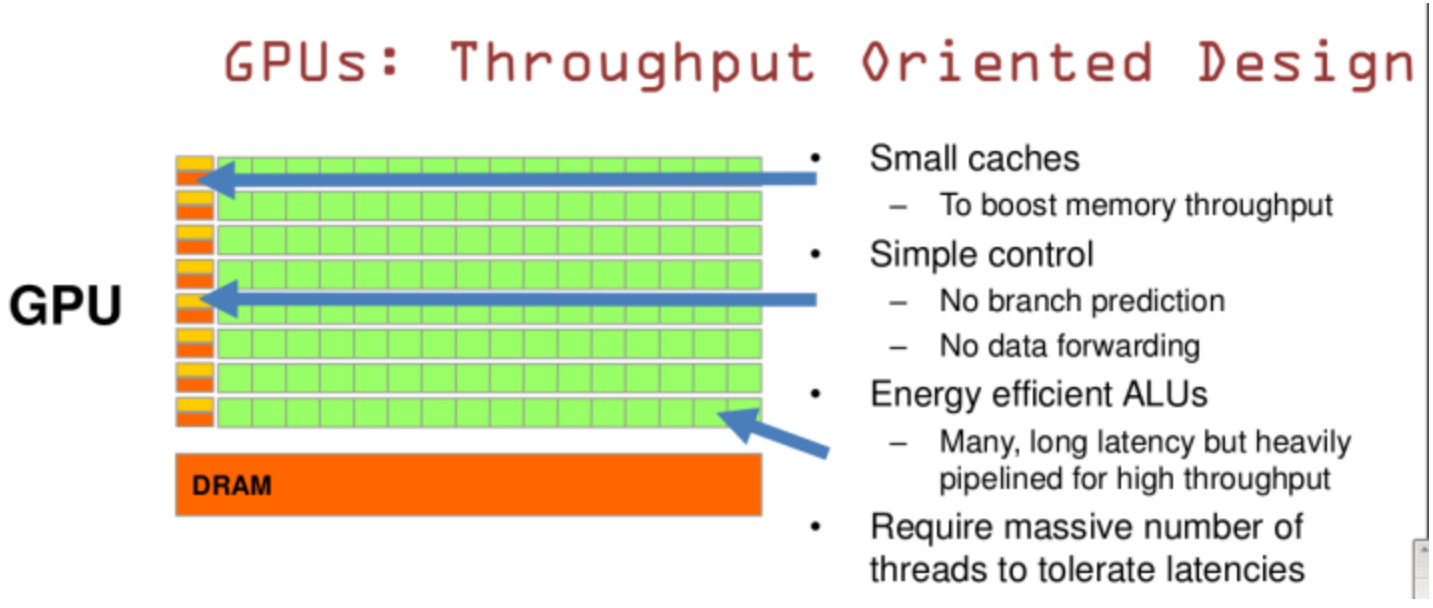
概念

CPU是基于低延迟的设计



概念

GPU是基于大吞吐量的设计



概念

支持Compute Shader的图形API

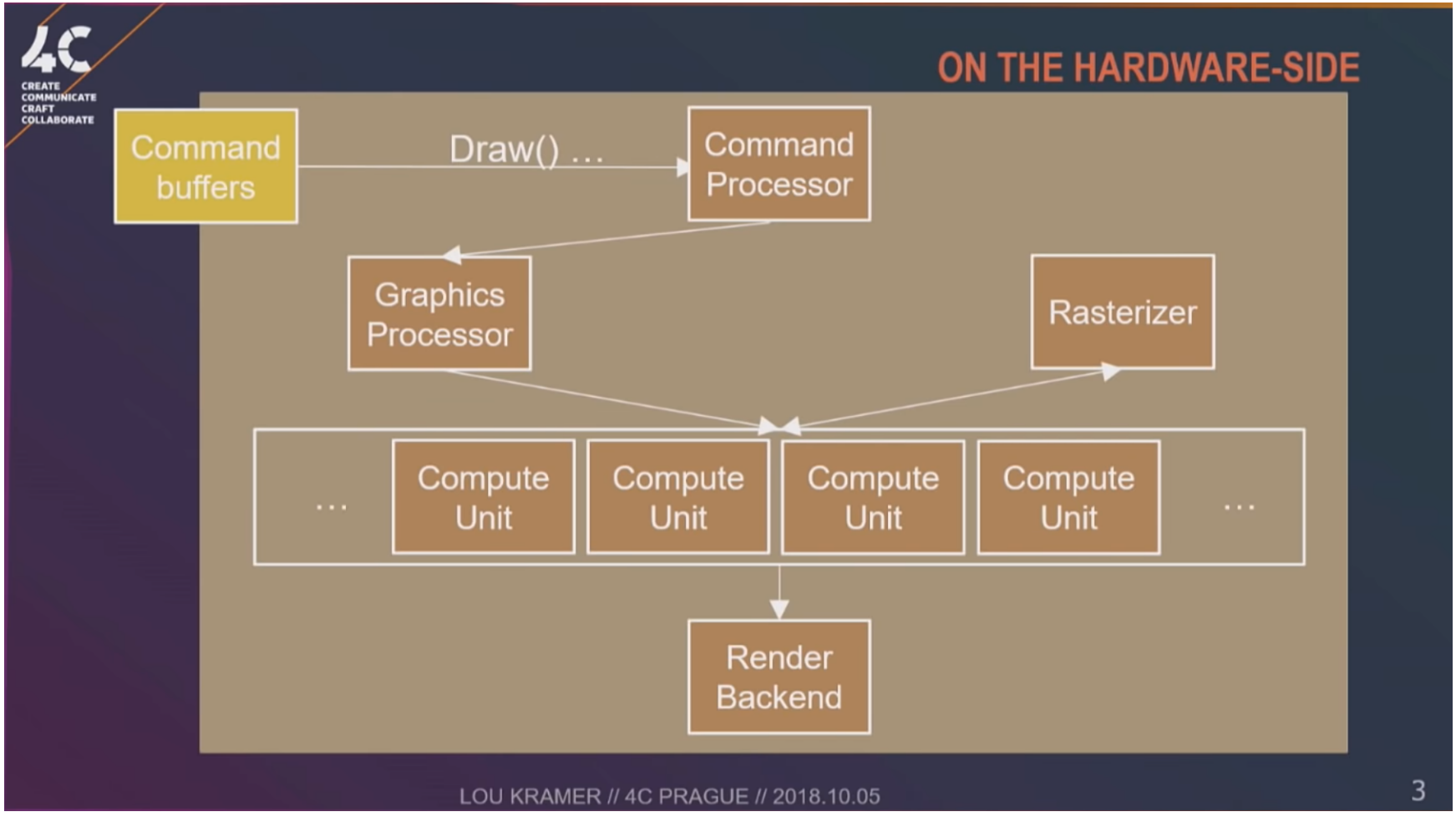
概念

Compute管线与图像管线的对比

Graphics pipeline	Compute pipeline
One to several shader stages (VS, HS, DS, GS, PS) Input assembler Tessellation Rasterizer …	CS shader stage

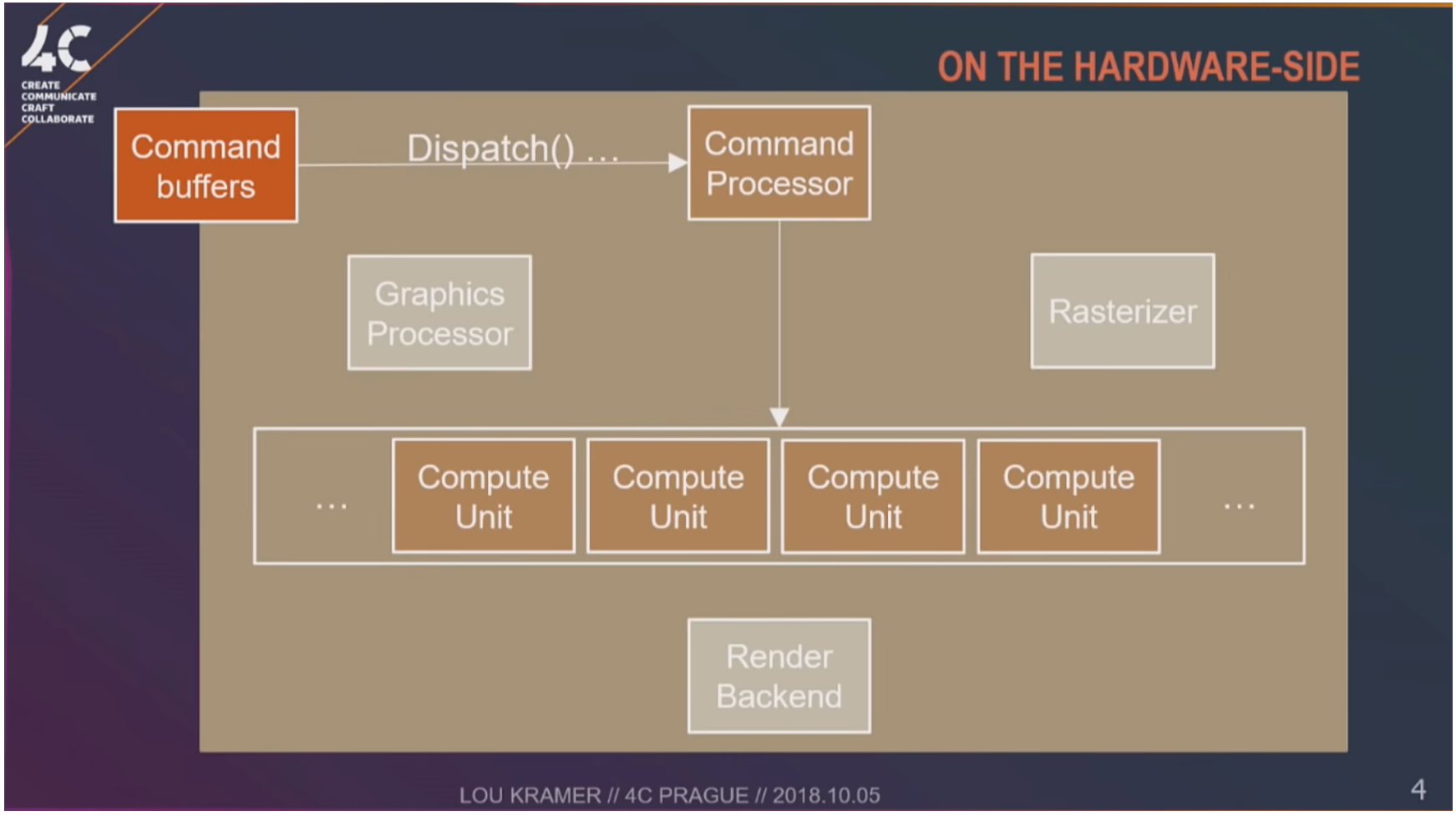
概念

渲染管线（硬件端）


CREATE
COMMUNICATE
CRAFT
COLLABORATE
ON THE HARDWARE-SIDE
Command buffers
Draw() …
Command Processor
Graphics Processor
Rasterizer
…
Compute Unit
Compute Unit
Compute Unit
Compute Unit
…
Render Backend
LOU KRAMER // 4C PRAGUE // 2018.10.05
3

概念

计算管线（硬件端）


4C
CREATE
COMMUNICATE
CRAFT
COLLABORATE
ON THE HARDWARE-SIDE
Command buffers
Dispatch() …
Command Processor
Graphics Processor
Rasterizer
…
Compute Unit
Compute Unit
Compute Unit
Compute Unit
…
Render Backend
LOU KRAMER // 4C PRAGUE // 2018.10.05
4

语法

kernel

```
// test.compute
#pragma kernel FillWithRed

RWTexture2D<float4> res;

[numthreads(8,8,1)]
void FillWithRed (uint3 dtid : SV_DispatchThreadID)
{
    res[dtid.xy] = float4(1,0,0,1);
}
```

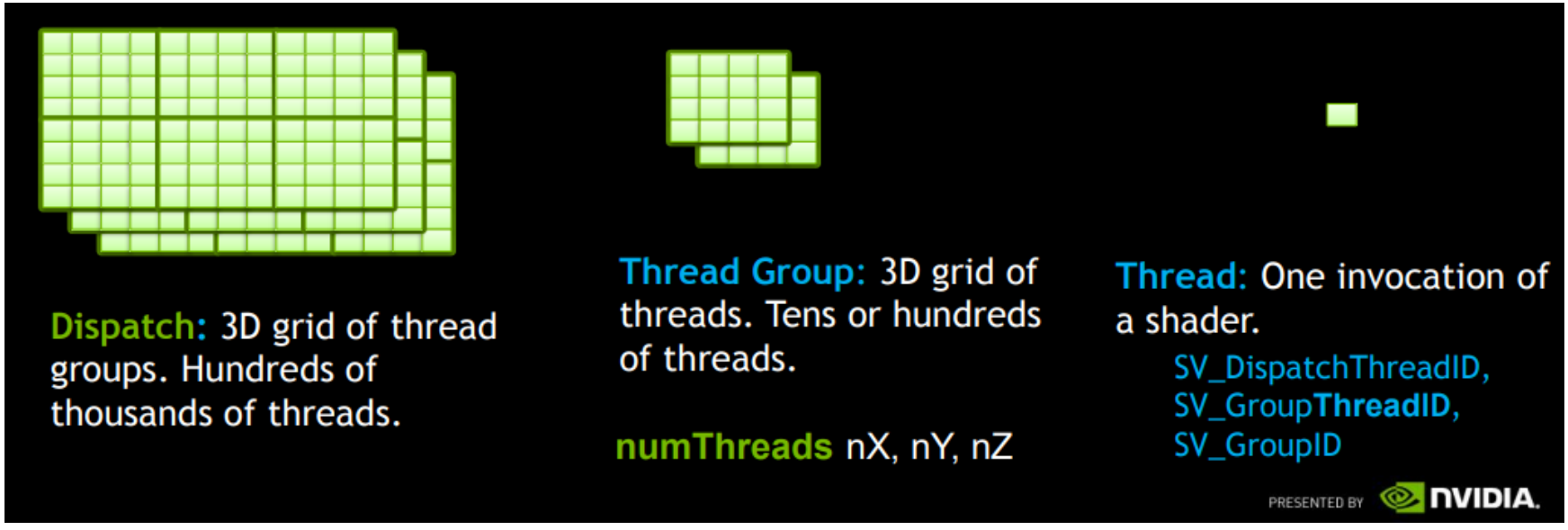
语法

Dispatch

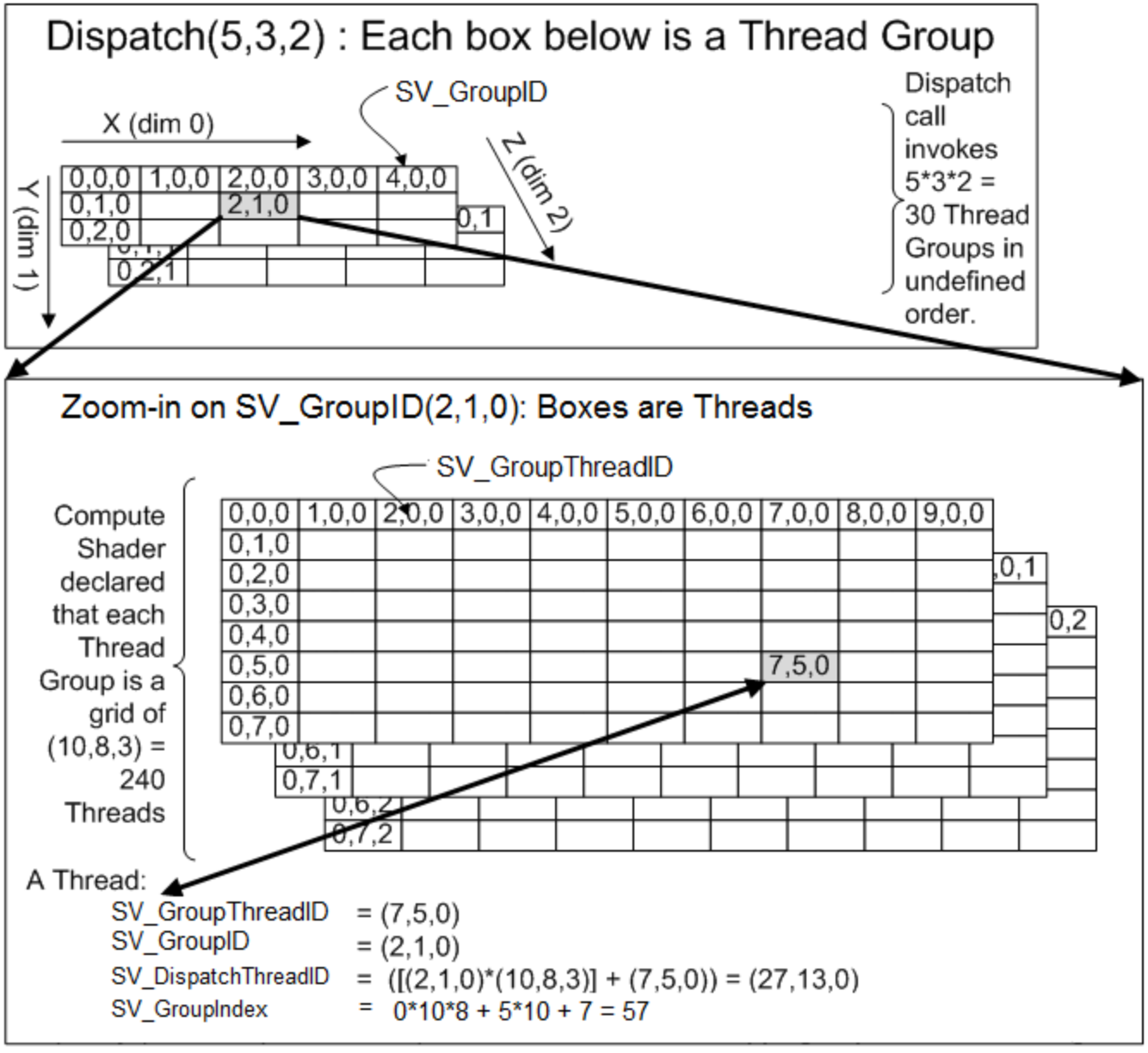
```
public void Dispatch(int kernelIndex,
        int threadGroupsX,
        int threadGroupsY,
    int threadGroupsZ);
```

语法

numthreads



语法


Dispatch(5,3,2) : Each box below is a Thread Group
SV_GroupID
X (dim 0)
Z (dim 2)
Y (dim 1)
Dispatch call invokes 5*3*2 = 30 Thread Groups in undefined order.
Zoom-in on SV_GroupID(2,1,0): Boxes are Threads
SV_GroupThreadID
Compute Shader declared that each Thread Group is a grid of (10,8,3) = 240 Threads
A Thread:
SV_GroupThreadID = (7,5,0)
SV_GroupID = (2,1,0)
SV_DispatchThreadID = ([(2,1,0)*(10,8,3)] + (7,5,0)) = (27,13,0)
SV_GroupIndex = 0*10*8 + 5*10 + 7 = 57

语法

Buffer & Texture

GPU Side	CPU Side
*StructuredBuffer	ComputeBuffer
Texture*D	Texture
RWTexture*D	RenderTexture

语法

groupshared

使用groupshared可以将一个变量标记为组内共享。（又叫TGSM）

语法

Barrier

当我们在不同线程访问同一个资源的时候，我们需要使用barrier来进行阻塞。

```
GroupMemoryBarrier
GroupMemoryBarrierWithGroupSync
DeviceMemoryBarrier
DeviceMemoryBarrierWithGroupSync
AllMemoryBarrier
AllMemoryBarrierWithGroupSync
```

语法

Interlocked

原子操作，不会被线程调度机制打断。

```
InterlockedAdd
InterlockedAnd
InterlockedCompareExchange
InterlockedCompareStore
InterlockedExchange
InterlockedMax
InterlockedMin
InterlockedOr
InterlockedXor
```

但是只能用于int/uint

语法

平台差异

- **数组越界**，DX上会返回0，其它平台会出错。
- 变量名与关键字/内置库函数**重名**，DX无影响，其他平台会出错。
- 如果StructuredBuffer内结构的显存布局要与**内存布局不一致**，DX可能会转换，其他平台会出错。
- **未初始化**的Buffer或Texture，在某些平台上会全部是0，但是另外一些可能是任意值，甚至是NaN。
- Metal不支持**对纹理的原子操作**，不支持对buffer调用**GetDimensions**。
- OpenGL ES 3.1在一个ComputeShader里**至少支持4个buffer**（所以，我们需要将相关联的数据定义为结构体）。

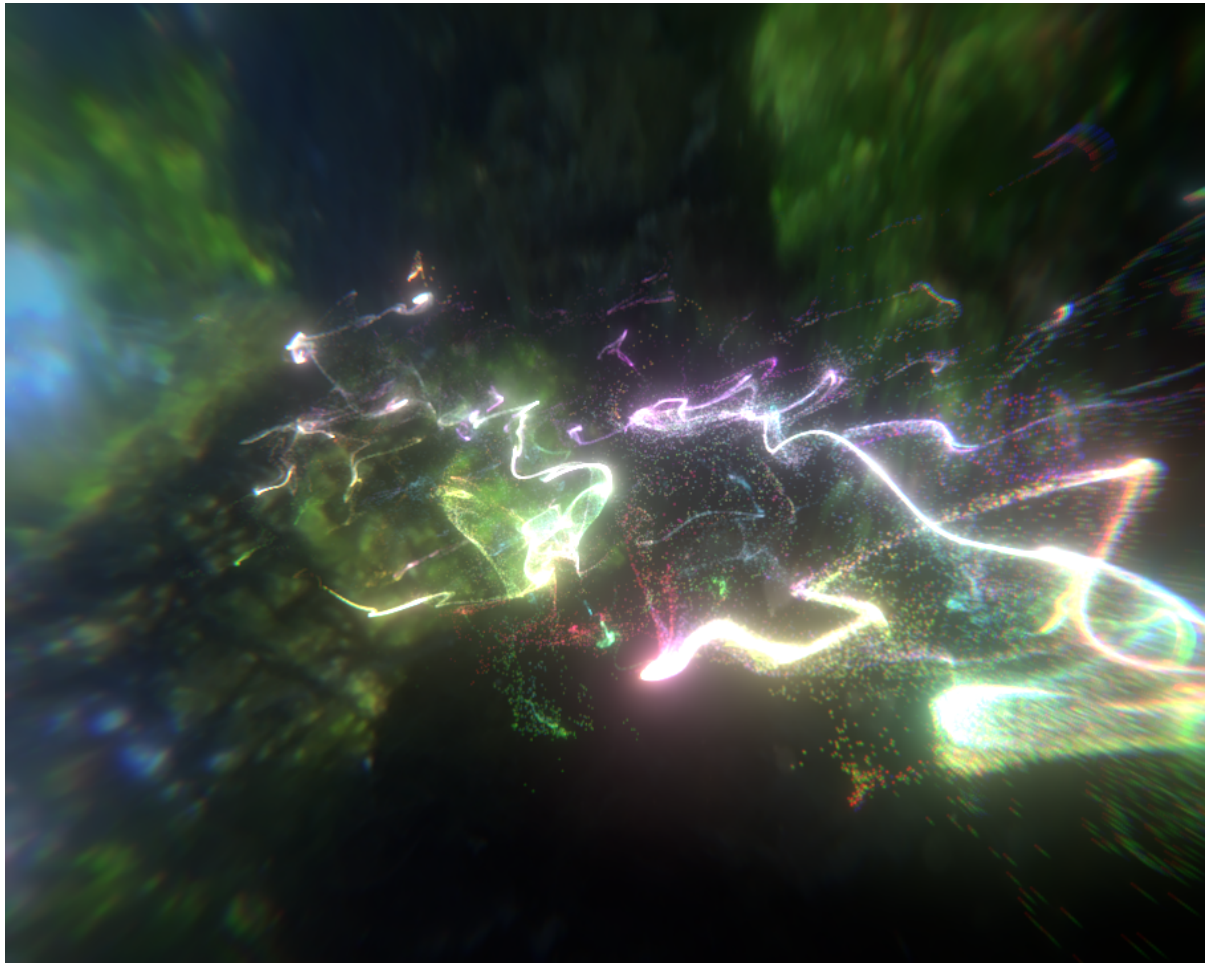
语法

性能

- 尽量减少Group之间的交互
- GPU一次调用64（AMD）或32（NVIDIA）个线程，所以，尽量使numthreads的乘积是这个值的整数倍。（但是Mali不需要这种优化，Metal可以通过api获取这个值）
- 避免回读
- 避免分支，重点避免在thread group中间的分支
- *尽量保证内存连续性*
- *使用[unroll]来打开循环，有些时候需要手动unroll*

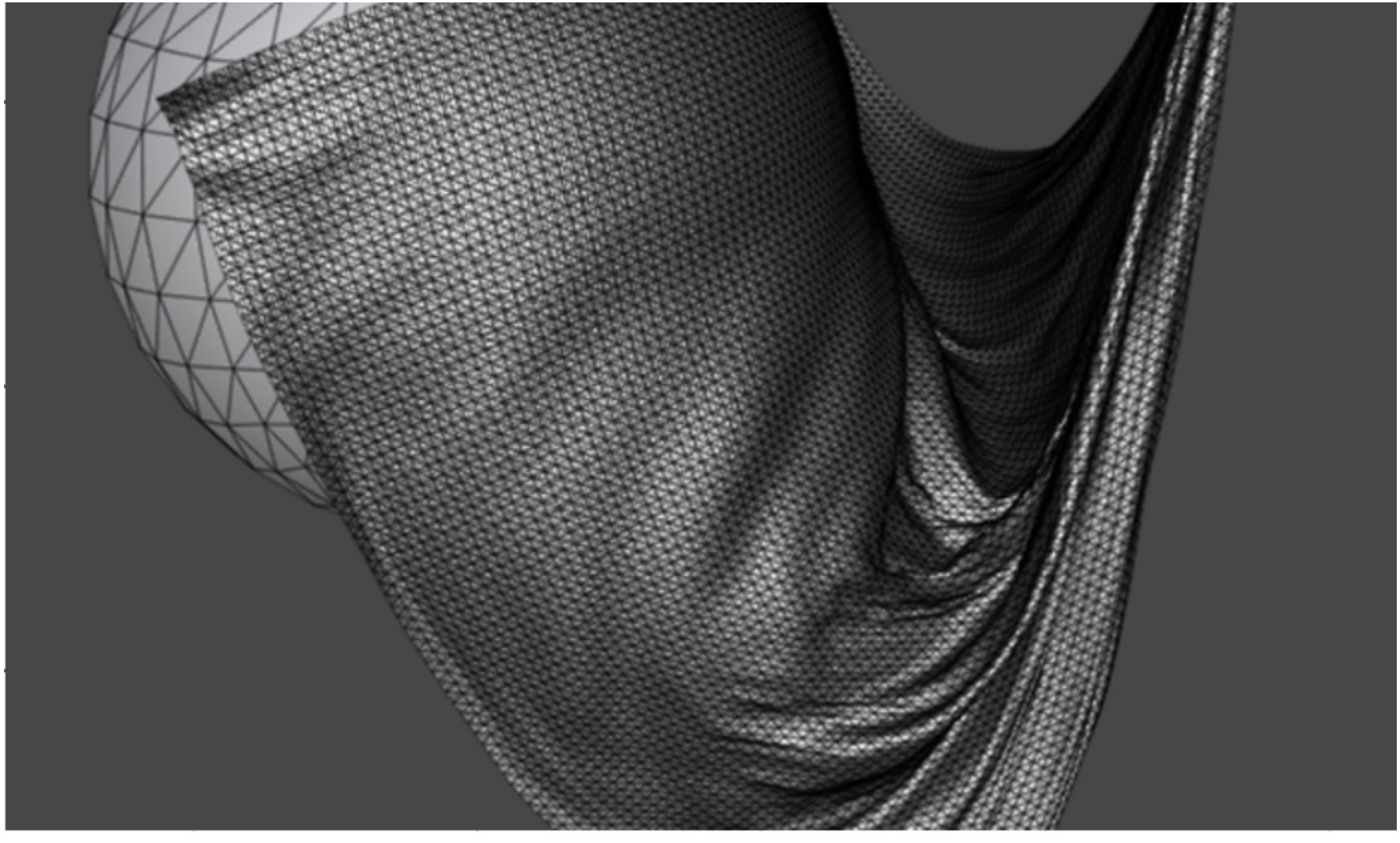
用途

GPU Particle System



用途

GPU Simulation



用途

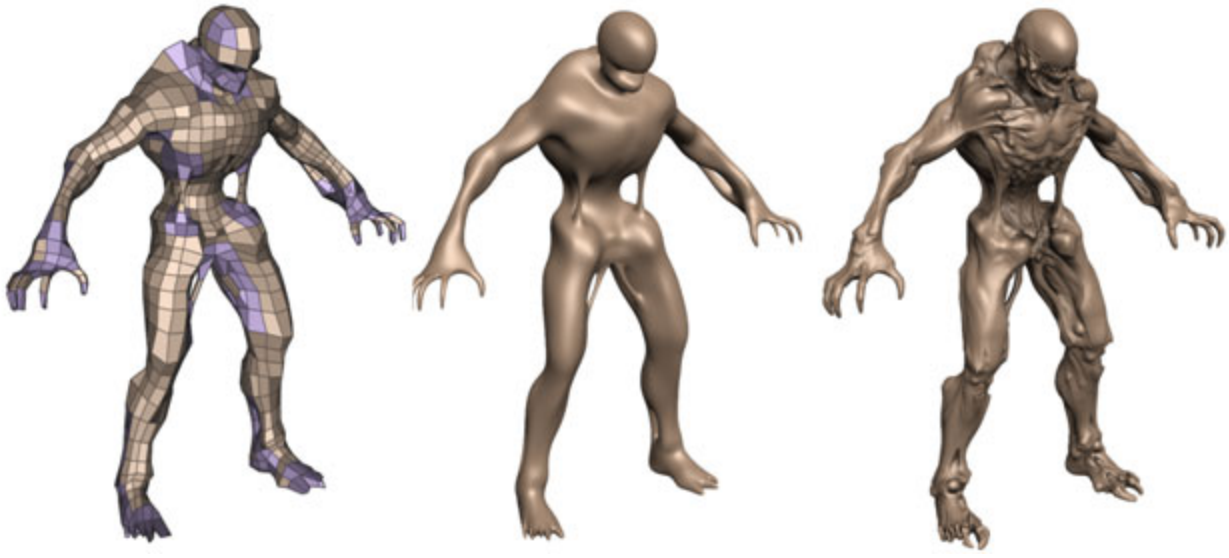
Image Processing

用途

Image Compression

用途

Tessellation



用途

Local lights culling

用途

Occlusion culling

用途

GPU Driven Rendering Pipeline

用途

还有很多很多.....

引用

1. https://slideplayer.com/slide/9636742/
2. http://on-demand.gputechconf.com/gtc/2010/presentations/S12312-DirectCompute-Pre-Conference-Tutorial.pdf
3. https://www.youtube.com/watch?v=0DLOJPSxJEg
4. https://arm-software.github.io/vulkan-sdk/basic_compute.html
5. https://www.khronos.org/opengl/wiki/Compute_Shader
6. https://docs.microsoft.com/en-us/windows/desktop/direct3d11/direct3d-11-advanced-stages-compute-shader
7. https://developer.apple.com/metal/
8. https://static.docs.arm.com/100614/0302/arm_mali_gpu_opencl_developer_guide_100614_0302_00_en.pdf
9. Real-Time Rendering 3rd Edition. Chapter 18

引用

10. https://github.com/Robert-K/gpu-particles
11. https://www.shpakivnia.com/cloth-tool
12. http://www.codinglabs.net/tutorial_compute_shaders_filters.aspx
13. https://en.wikipedia.org/wiki/Adaptive_Scalable_Texture_Compression
14. Introduction to 3D Game Programming with DirectX 11
15. https://www.nvidia.com/object/tessellation.html
16. https://www.slideshare.net/DICEStudio/directx-11-rendering-in-battlefield-3
17. https://zhuanlan.zhihu.com/p/47615677
18. http://advances.realtimerendering.com/s2015/aaltonenhaar_siggraph2015_combined_final_footer_220dpi.pdf

引用

19. https://docs.unity3d.com/Manual/class-ComputeShader.html
20. https://forum.unity.com/threads/compute-shaders.148874/
21. https://docs.unity3d.com/ScriptReference/ComputeShader.Dispatch.html
22. http://www.humus.name/Articles/Persson_LowlevelShaderOptimization.pptx
23. https://www.gdcvault.com/play/1020352/Low-Level-Shader-Optimization-for
24. 数字图像处理（冈萨雷斯）