Compute Shader

Optimize your game using compute shader

凯奥斯

Compute Shader

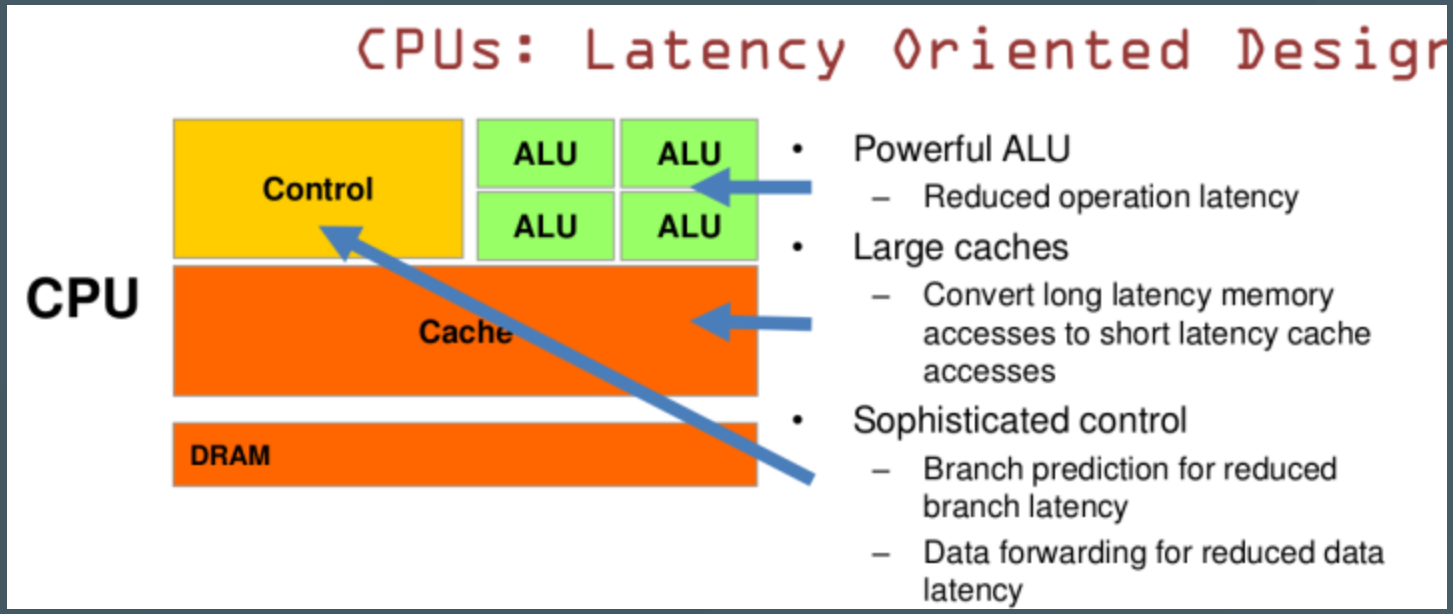
- 概念
- 语法
- 用途

概念

介绍一下背景知识

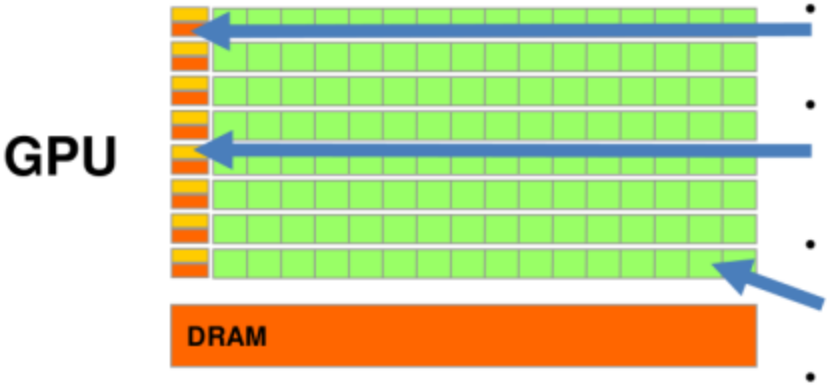
GPGPU

CPU是基于低延迟的设计



GPU是基于大吞吐量的设计

GPUs: Throughput Oriented Design



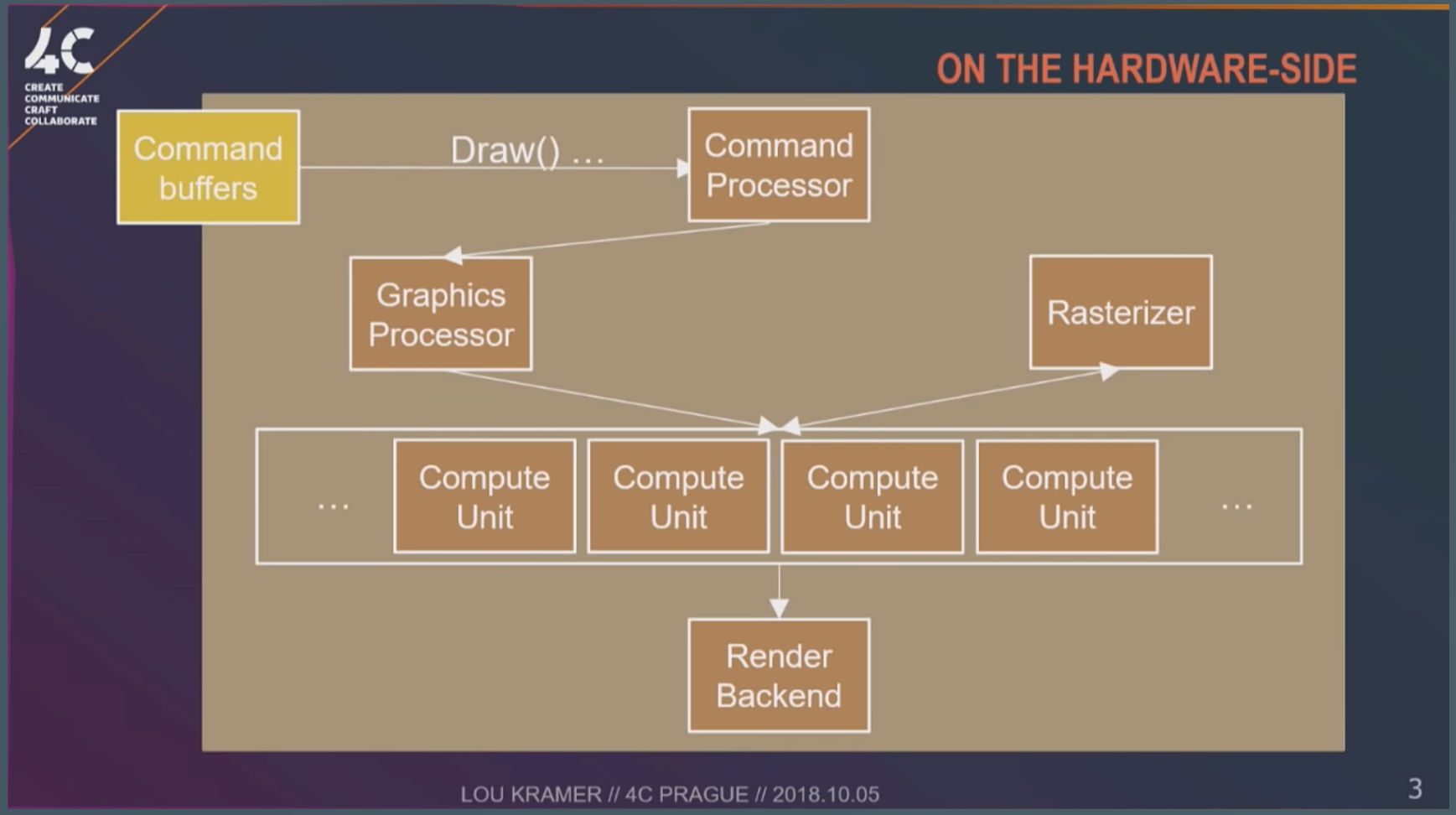
- Small caches
  - To boost memory throughput
- Simple control
  - No branch prediction
  - No data forwarding
- Energy efficient ALUs
  - Many, long latency but heavily pipelined for high throughput
- Require massive number of threads to tolerate latencies

支持Compute Shader的图形API

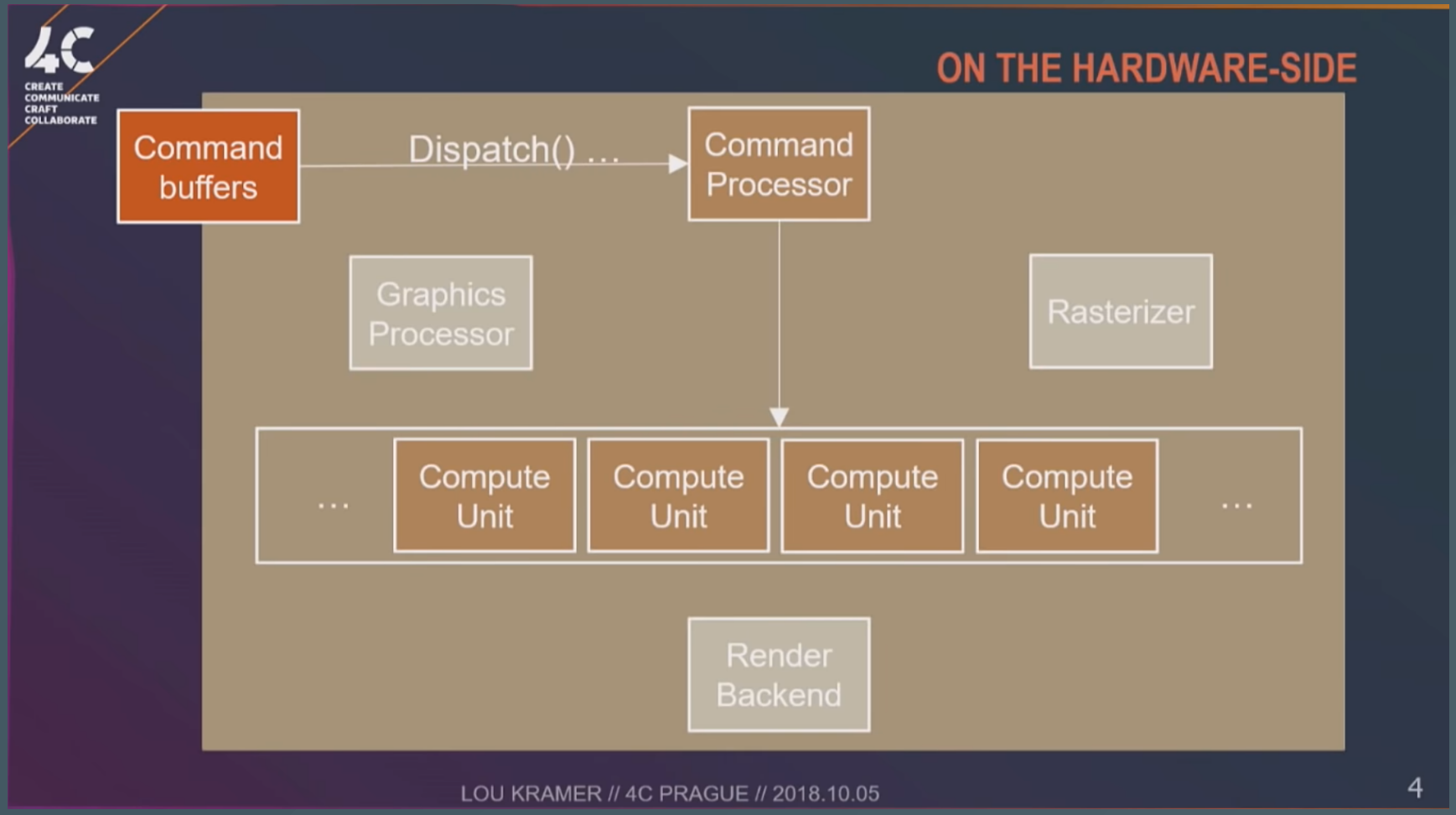
Compute管线与图像管线的对比

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

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

计算管线（硬件端）


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

如何在Unity里使用Compute Shader

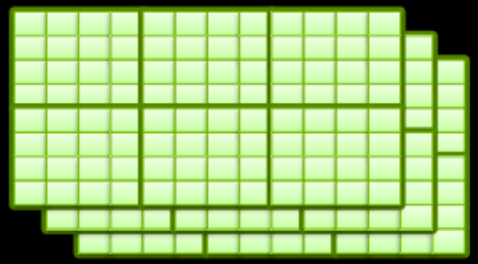
kernel

```
// 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);
}
```

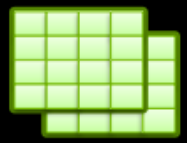
Dispatch

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

numthreads



Dispatch: 3D grid of thread groups. Hundreds of thousands of threads.



Thread Group: 3D grid of threads. Tens or hundreds of threads.

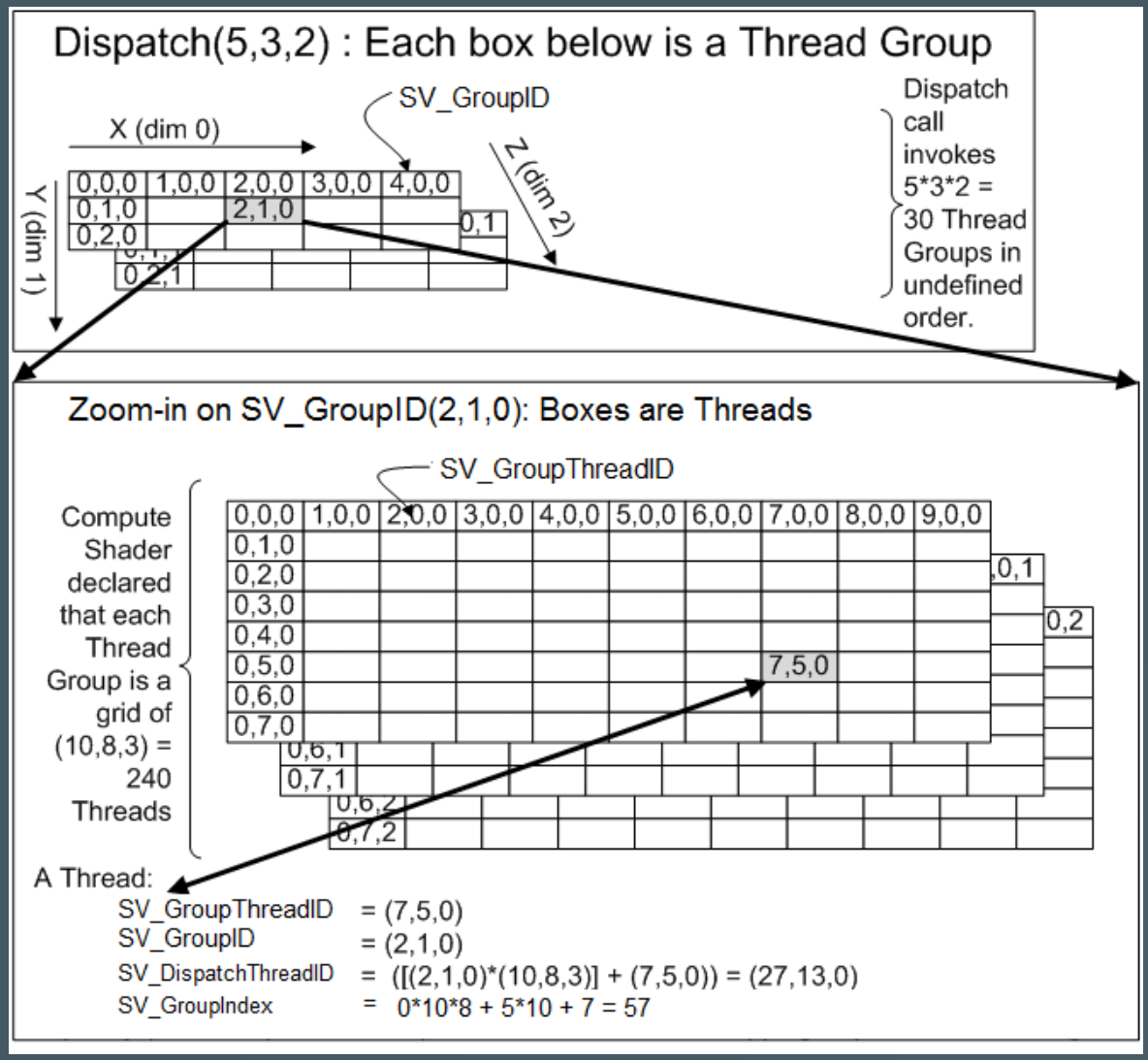
numThreads nX, nY, nZ

Thread: One invocation of a shader.

SV_DispatchThreadID,
SV_Group**ThreadID**,
SV_GroupID

PRESENTED BY NVIDIA.

thread groups



Buffer & Texture

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

groupshared

使用groupshared可以将一个变量标记为组内共享。（又叫TGSM）

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
```

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

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SBuffer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SBuffer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SBuffer调用**GetDimensions**。
- ES 3.1在一个CS里至少支持**4个SBuffer**（所以，我们需要将相关联的数据定义为struct）。

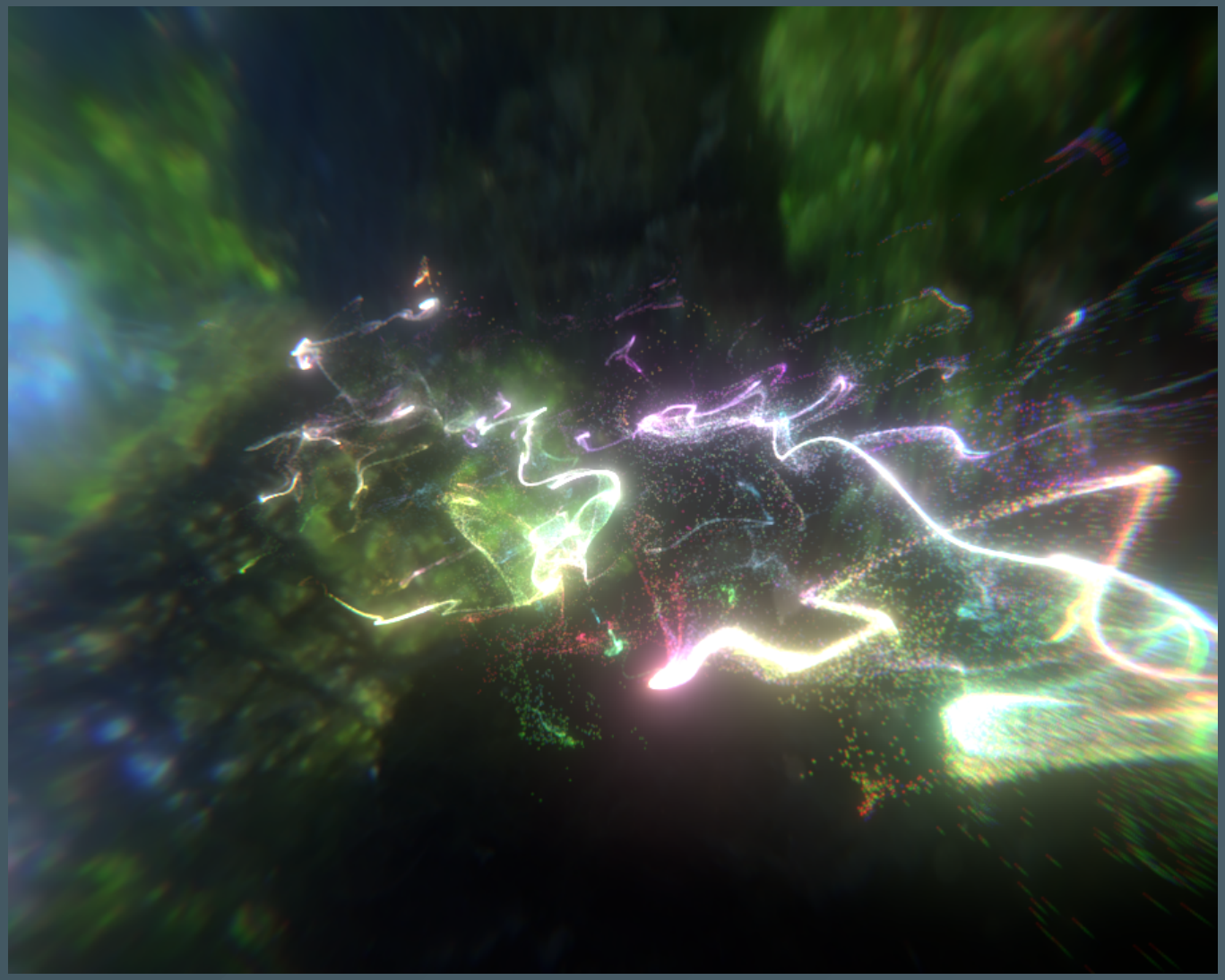
性能

- 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。
- 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*

应用

目前有哪些应用

GPU Particle System



GPU Simulation

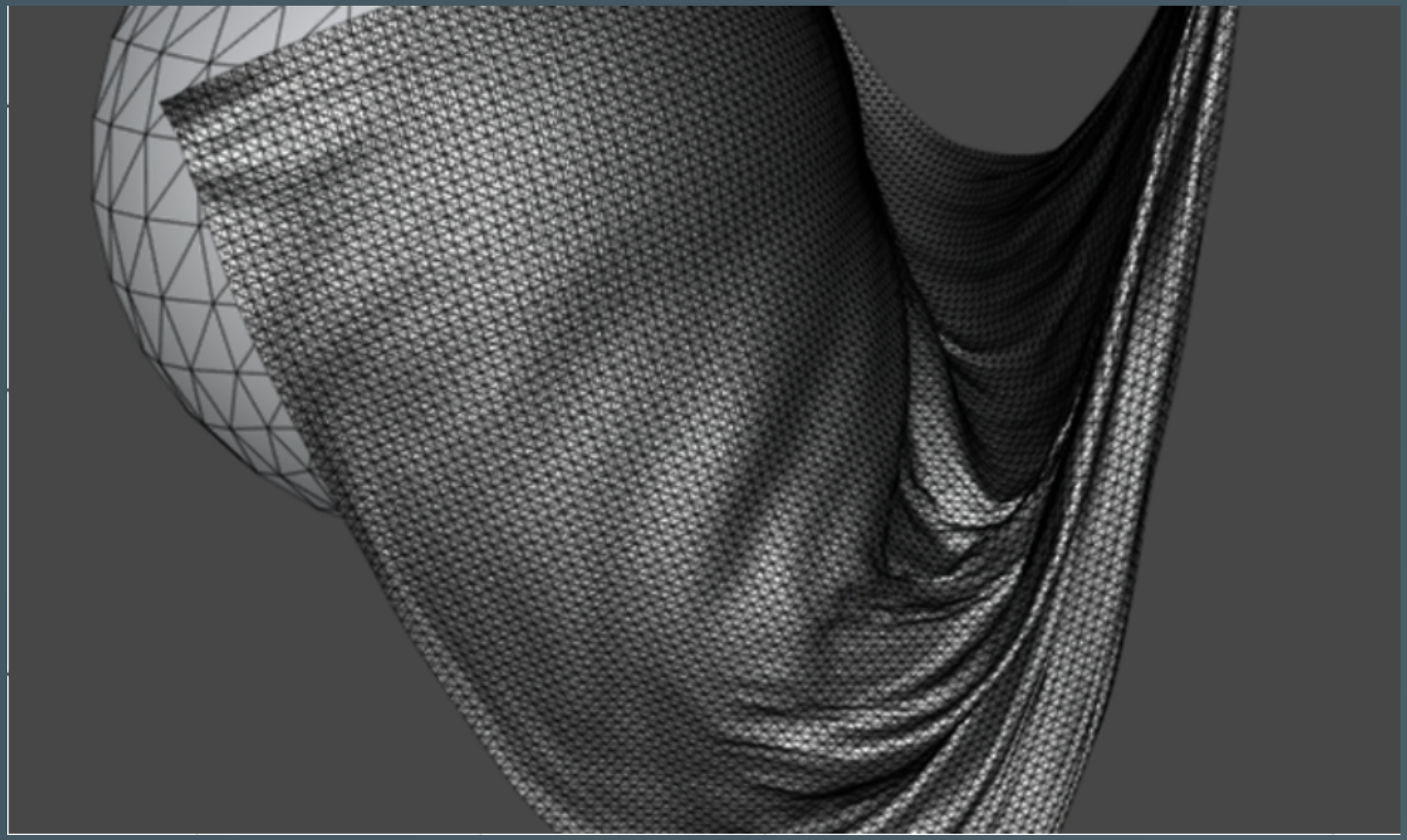
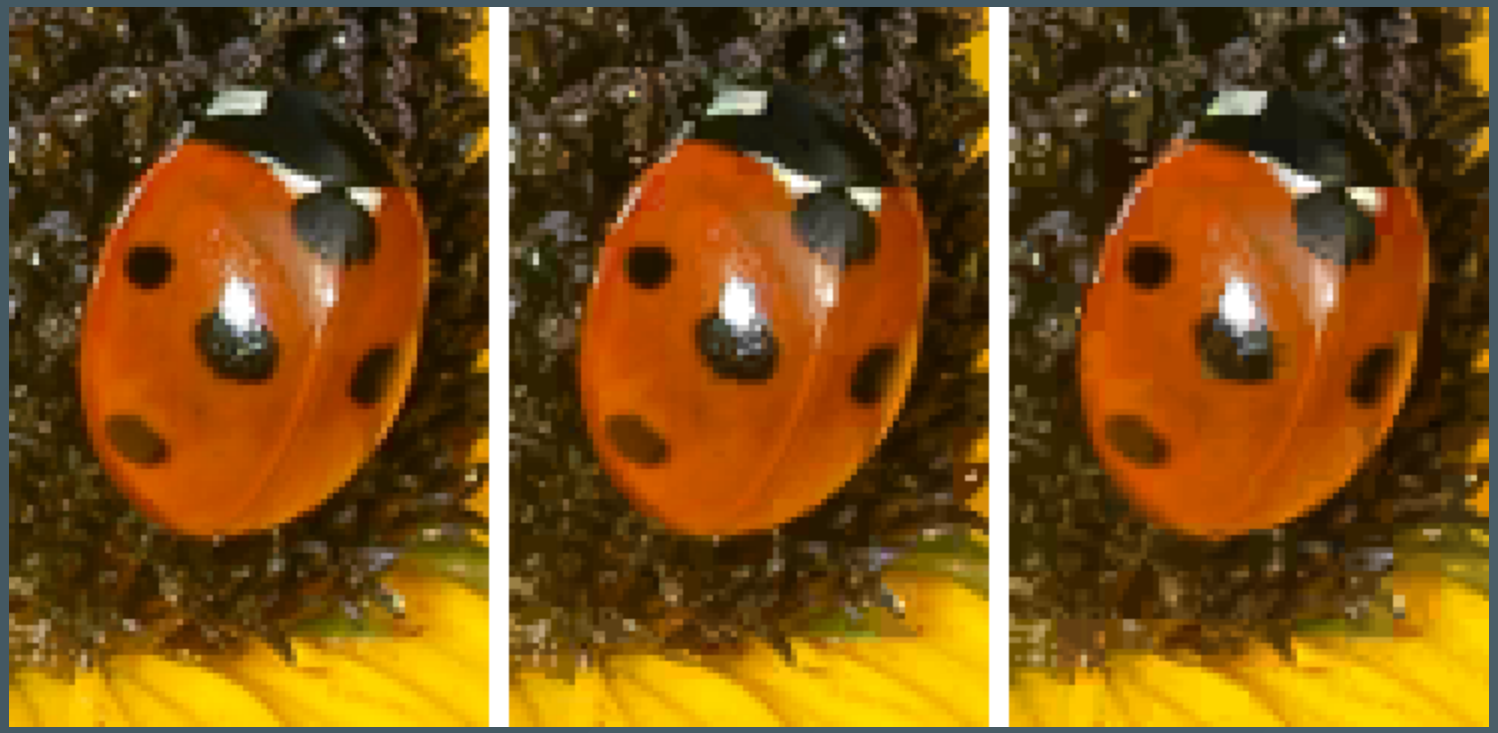
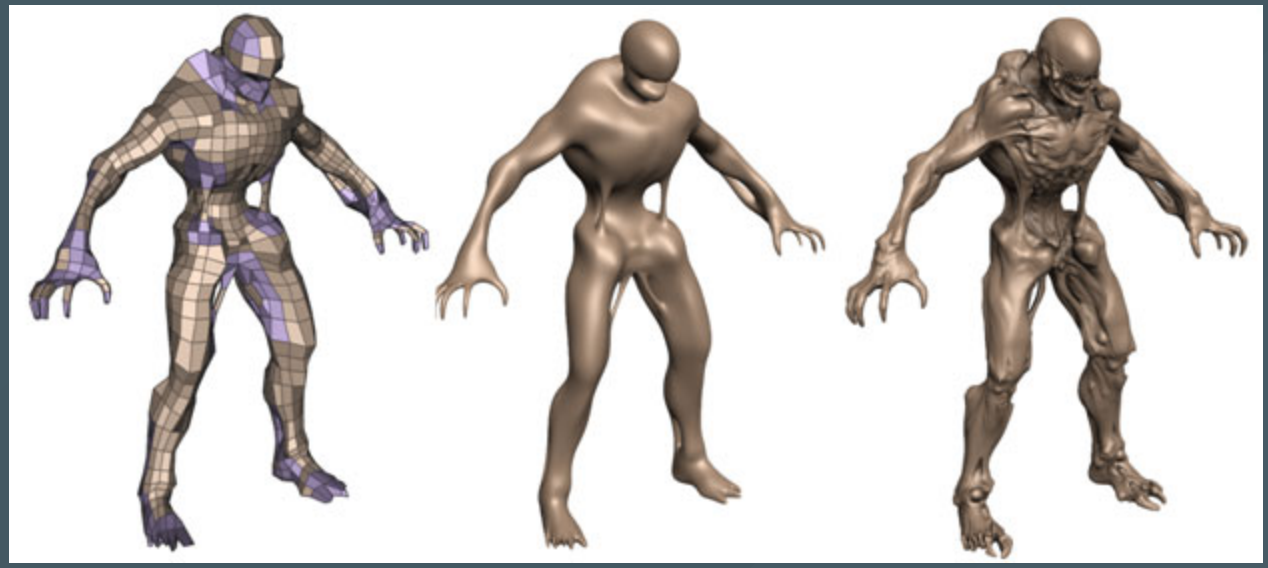


Image Processing

Image Compression



Tessellation



Local lights culling

Occlusion culling

GPU Driven Rendering Pipeline

还有很多很多.....

引用

1. Graphic Processing Processors (GPUs) Parallel Programming
2. DirectCompute Optimizations and Best Practices
3. Compute Shaders: Optimize your engine using compute / Lou Kramer, AMD (video)
4. Introduction to Compute Shaders in Vulkan
5. Compute Shader(OpenGL)
6. Compute Shader Overview(Direct3D 11)
7. About Threads and Threadgroups(Metal)

引用

8. ARM® Mali™ GPU OpenCL Developer Guide(Version 3.2)
9. Real-Time Rendering 3rd Edition. Chapter 18
10. GPU Particles (Github)
11. GPU Cloth Tool
12. Compute Shader Filters
13. ASTC
14. Introduction to 3D Game Programming with DirectX 11
15. DirectX 11 Tessellation (NVIDIA)

引用

16. DirectX 11 Rendering in Battlefield 3
17. Hi-Z GPU Occlusion Culling
18. GPU-Driven Rendering Pipelines
19. Compute shaders (Unity3D)
20. Problems with ComputeBuffer Readback
21. ComputeShader.Dispatch (Unity3D)
22. Low-level Shader Optimization for Next-Gen and DX11 (ppt)
23. Low-Level Shader Optimization for Next-Gen and DX11 (video)

引用

24. 数字图像处理（冈萨雷斯）
25. General-purpose computing on graphics processing units (Wikipedia)
26. Volume Tiled Forward Shading (Github)
27. Unity3D AssetPackages
28. Mythbusters Demo GPU versus CPU (NVIDIA)