中国游戏开发者大会

CHINA GAME DEVELOPERS CONFERENCE

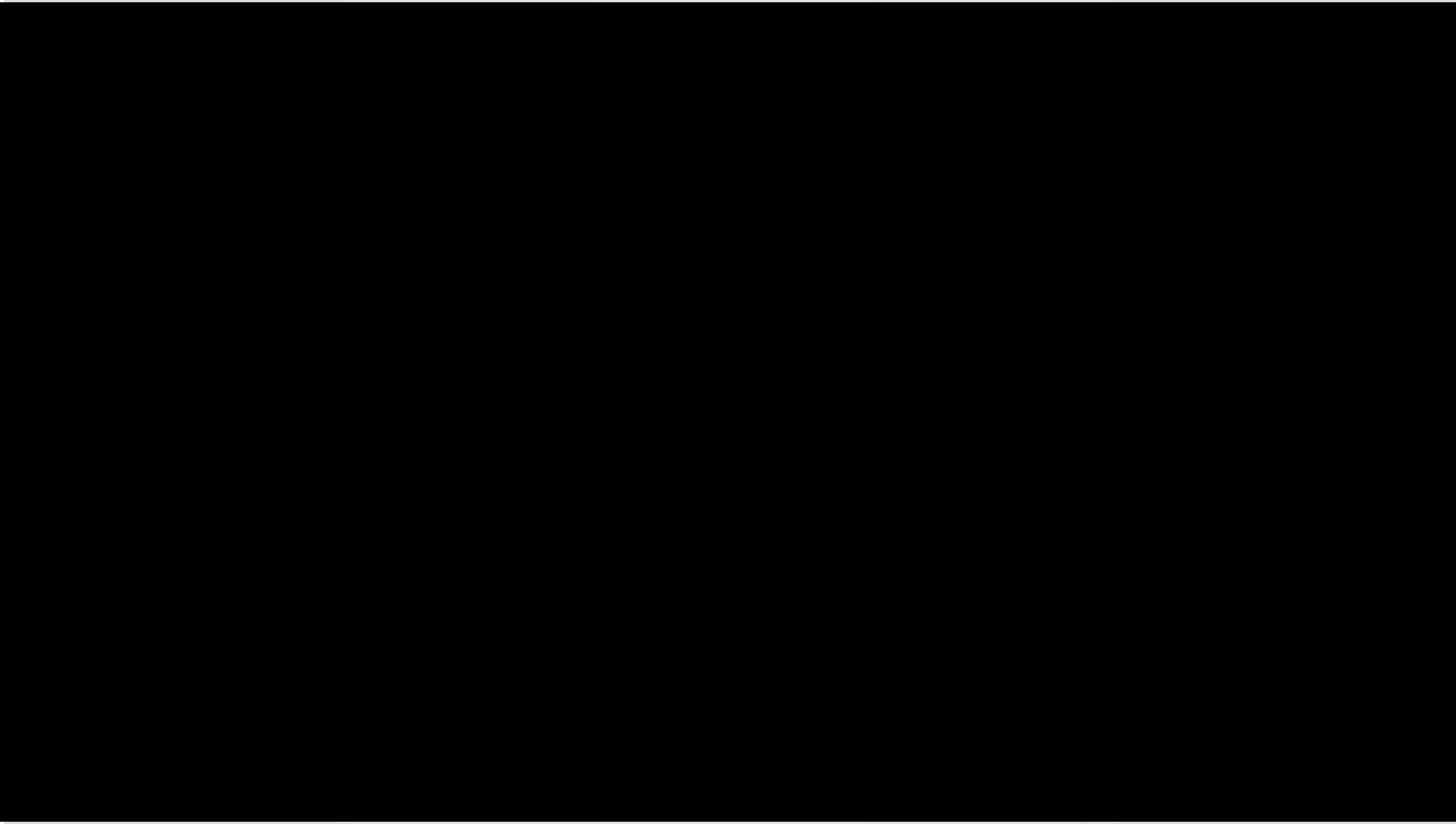
九阳神功的DX12版本开发及优化

吕文伟
引擎技术专家
蜗牛游戏(Snail Game)

杨雪青
内容技术开发工程师
英伟达(NVIDIA)

九阳神功

- 首款3D武侠类英雄团战网游
- 业界顶级的实时图形渲染效果
- 国内首款基于DirectX12的跨平台网络游戏
- 使用了最新的NVIDIA GameWorks特效和GPU特性
- Xbox One和PlayStation 4版本已经在国内上市，PC版也将在年内上市



DX12 移植

从DirectX11移植到DirectX12

- 两个资深图形程序员花费了6个星期时间为CryEngine3开发DX12版本的渲染器
- 使用DX12的API重新封装DX11的接口，在最大程度上保留原有的上层逻辑
- 使用HashMap管理所有的渲染状态，在运行时动态的去保存和查询
- 给每一个资源和状态分配一个全局唯一的顺序号，用它来拼接生成Hash值。如果该资源被释放，那么它的顺序号还可以回收再利用
- 每个资源在Hash值中安排一个合理位宽，并提供顺序号的上限保护
- SDKLayer(Debug Runtime)会让你及早的发现潜在的渲染错误

PipelineStateObject 管理

- 所有基于DX11的渲染状态设置都被缓冲并延迟到DrawCall调用处执行
  1. RasterizerState
  2. BlendState
  3. DepthStencilState
  4. InputLayout
  5. Shader

- 通过GetCachedBlob获取驱动处理过PSO的硬件Cache数据，加快PSO的创建过程

Sampler 管理

- Sampler管理

1. 一个Sampler Heap最多只能存放2048个Sampler
2. 把每一个Shader所使用的16个Sampler编为一组进行生成Hash值。因此一帧中最多出现128种Sampler组合
3. 上层渲染逻辑尽量安排相同的Sampler在固定的Slot上，例如NormalMap，ShadowMap等
4. 想要更多的Sampler组合，只能切换Heap

View 管理

- Shader Resource View管理

1. View Heap最多能够存放1M个View Descriptor
2. 采用与Sampler一样的数量进行管理，保证上层逻辑每个DrawCall只会使用最多16个SRV

- Contant Buffer View管理

1. 不用Descriptor Table，而是使用Root Descriptor，因为Constant Buffer属于动态数据，它的GPU Address(即资源的内偏移)会频繁发生变化
2. 如果使用的是静态的CB可用Descriptor Table方式来管理。但是Slot要固定下来
3. 每个Shader Stage引用的Constant Buffer都是256字节对齐的

Command List

- Command List的API调用不是线程安全的，不能跨线程调用
- Command Queue的Submit是线程安全的
- 为了避免渲染命令间歇性堵塞，可以创建多个Command List对命令进行缓冲
- 使用Fence同步多个Command List，用帧号作为同步完成的期待值
- 驱动和运行时都没有额外的线程帮助应用程序异步处理渲染命令的Build和Commit操作
- 将DX11的命令全部缓冲在命令队列里，然后在帧结束时，提交到另一个工作线程里延迟执行，这时需要在这个工作线程里把DX11的渲染命令解析成DX12的命令

小结

- 开放的内存管理模型给开发者更多的技术选择
- 灵活的资源绑定模式，降低了带宽的需求
- 强大且低开销的命令提交管线，让并行渲染逻辑的设计变得更加可定制化

NVIDIA GameWorks及GPU新特性的集成

TXAA的集成

- 在CE3 MSAA模式的基础上做集成
- 由于要做分时 AA，所以Resolve时需要向量贴图
  1. 直接利用原有MotionBlur的渲染结果
  2. 需要把Pack的数据解压成Float16的格式
- 需要记录前一帧TXAA的结果
- 使用NVAPI 设置光栅化阶段的子像素位置模式

HBAO+的集成

- 需要提供世界空间的法线图，引擎中原本的摄像机空间的法线图需要进行转换
- 需要将引擎中的深度图转换成相机空间的深度图
- 需要注意MSAA开启时所对应纹理的格式

Fast Geometry Shader的集成

- 渲染一遍场景就可以完成各层级阴影贴图的渲染
- 将各层阴影贴图可见的物体合在一个可见列表中
- 选用包含各层阴影贴图视野范围的视锥体进行渲染
- 阴影贴图存放在贴图数组中
  1. 作为Render Target数组进行渲染
  2. 为每个层级的阴影贴图设置不同Viewport
  3. 调用Fast GS完成对Render Target数组的渲染

小结

- 除此之外，早前我们还集成了Hair和Clothing的特效
- 使用成熟的GameWorks特效库，节省了很多研发成本
- GameWorks 特效给予游戏产品更强的市场竞争力
- NVIDIA提供的周到可靠的技术支持
- 能够充分利用最新的GPU特性
  - 毕竟只有硬件厂商才最了解自身的产品

NVIDIA

九阳神功中应用的NVIDIA GameWorks及GPU新特性

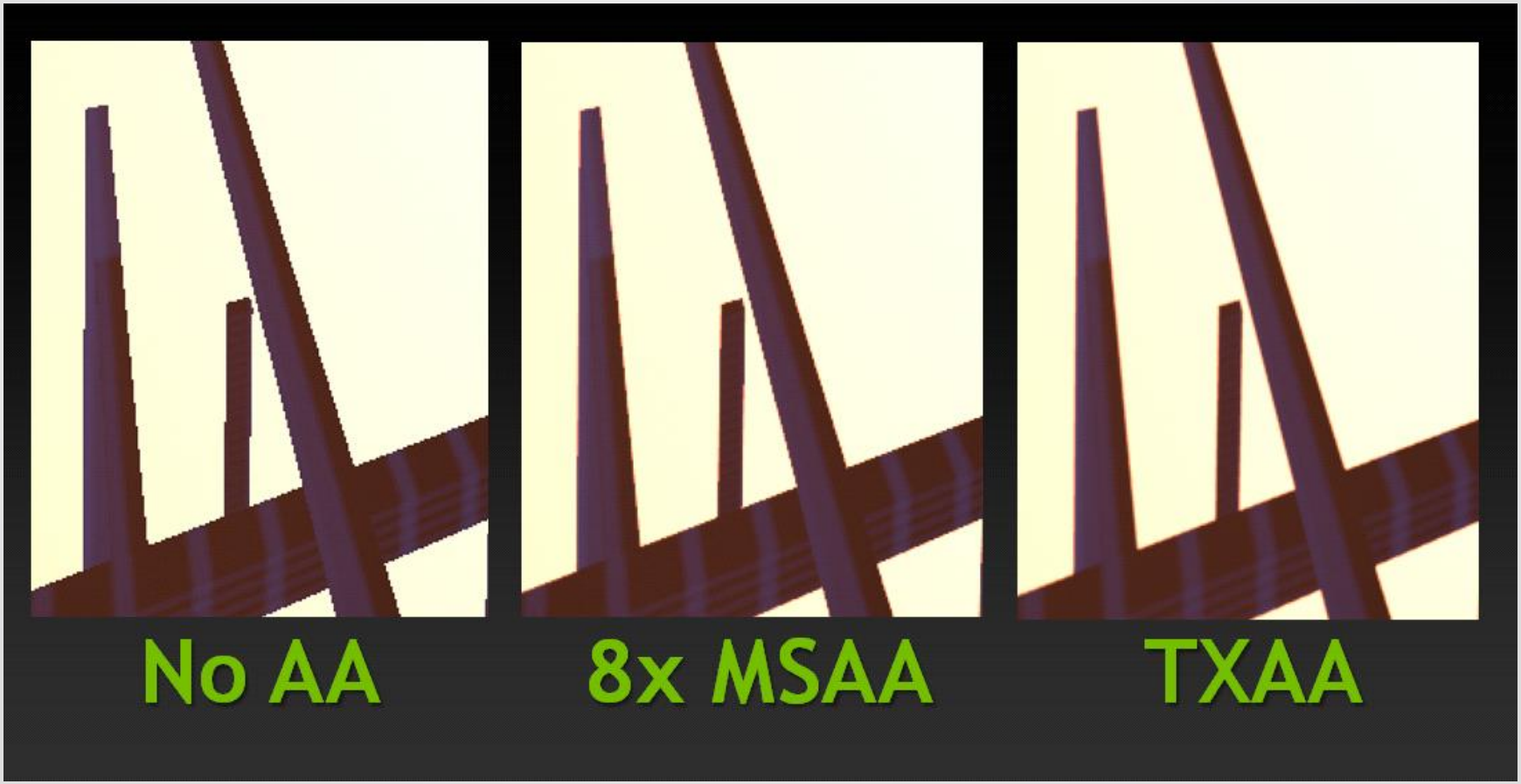
NVIDIA

概要

- GameWorks - VisualFX
  - PostWorks : TXAA3.0
  - ShadowWorks : HBAO+
  - HairWorks
- GameWorks – PhysX
  - 布料(Clothing)
- Maxwell的新特性
  - 多投影加速(Multi-Projection Acceleration)
    - 快速几何着色器(Fast Geometry Shader or Fast GS)
    - 快速多视口投射(Fast Viewport Multi-casting)

NVIDIA

TXAA 3.0



NVIDIA

TXAA简介

- 在MSAA的基础上增加了分时抗锯齿(Temporal AA)
- 替换了MSAA中的降解(Resolve)方法
- 提供比MSAA更高质量的滤镜
  - MSAA的滤镜仅使用简单的平均值

NVIDIA

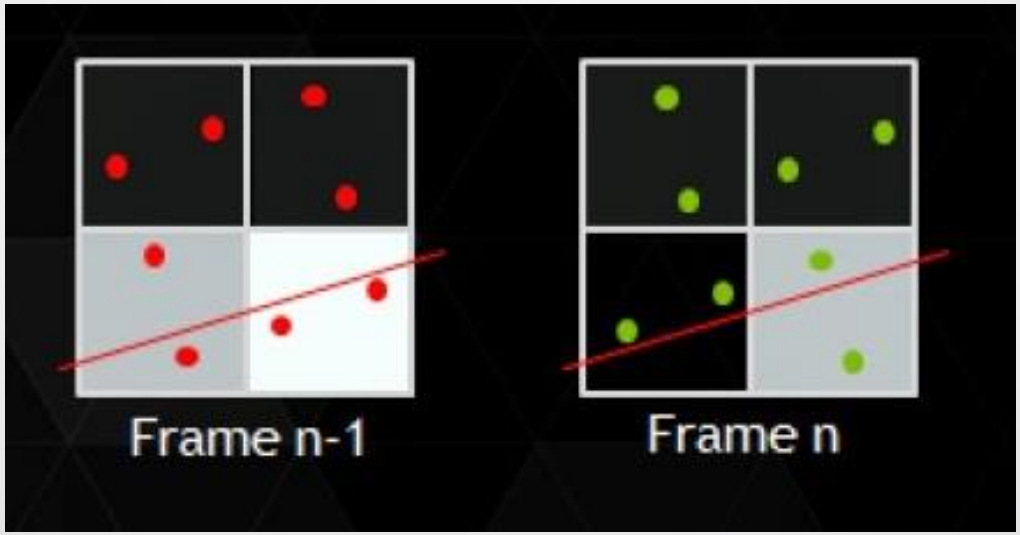
TXAA 3.0中的新特性

- 增加了更多的用户控制
  - 控制重构滤镜的使用
  - 逐像素控制AA的应用
- 提高AA的质量和效率
  - Maxwell新特性 : 可设计子像素位置(Programmable Sample Locations, PSL)

NVIDIA

可设计子像素位置(Programmable sample locations)

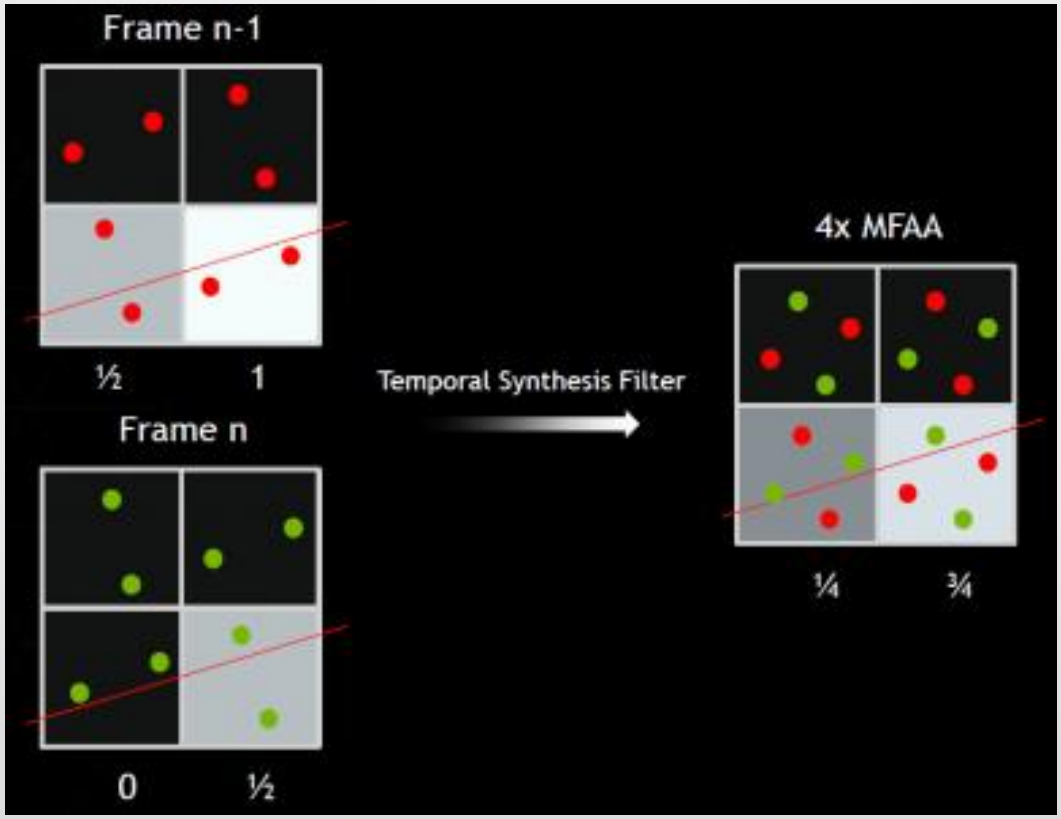
- 子像素位置是可以自己设计定义



NVIDIA

可设计子像素位置(Programmable sample locations)

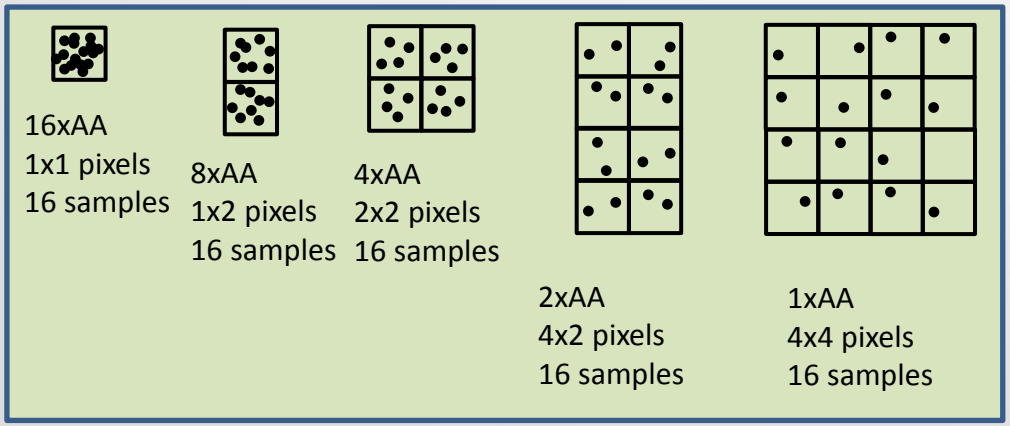
- 使用低级别的MSAA的消耗实现高级别MSAA的效果

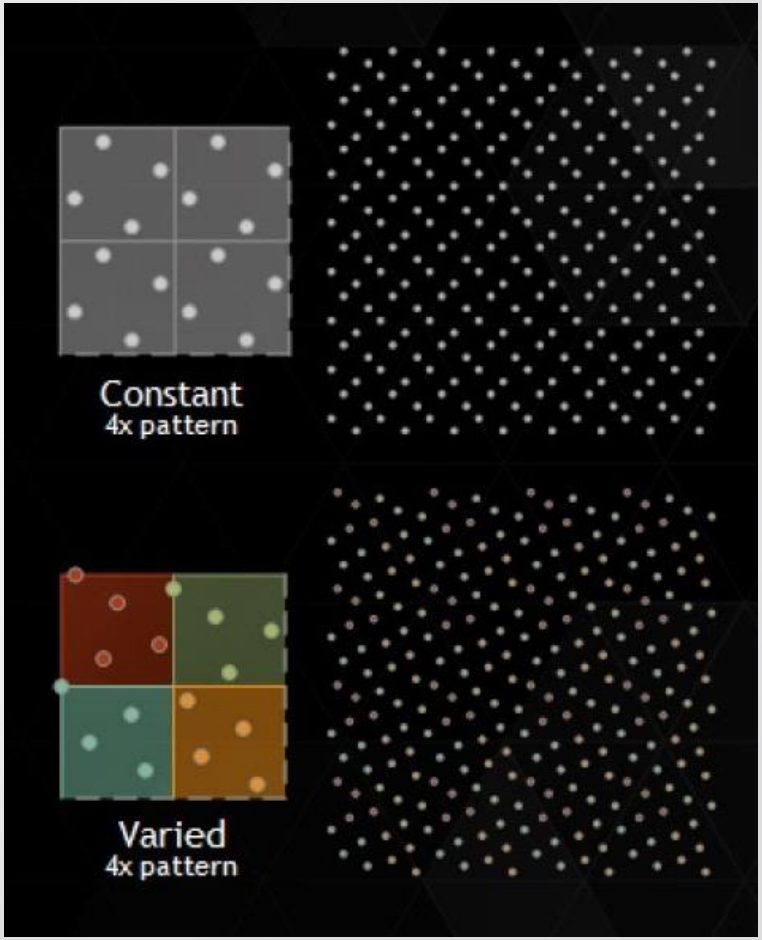


NVIDIA

可设计子像素位置(Programmable sample locations)

- 交错子像素位置(Interleaved sample positions)
  - 16个子像素位置可以平铺到各个像素上
  - 提高AA质量





没有AA

NVIDIA

4xMSAA

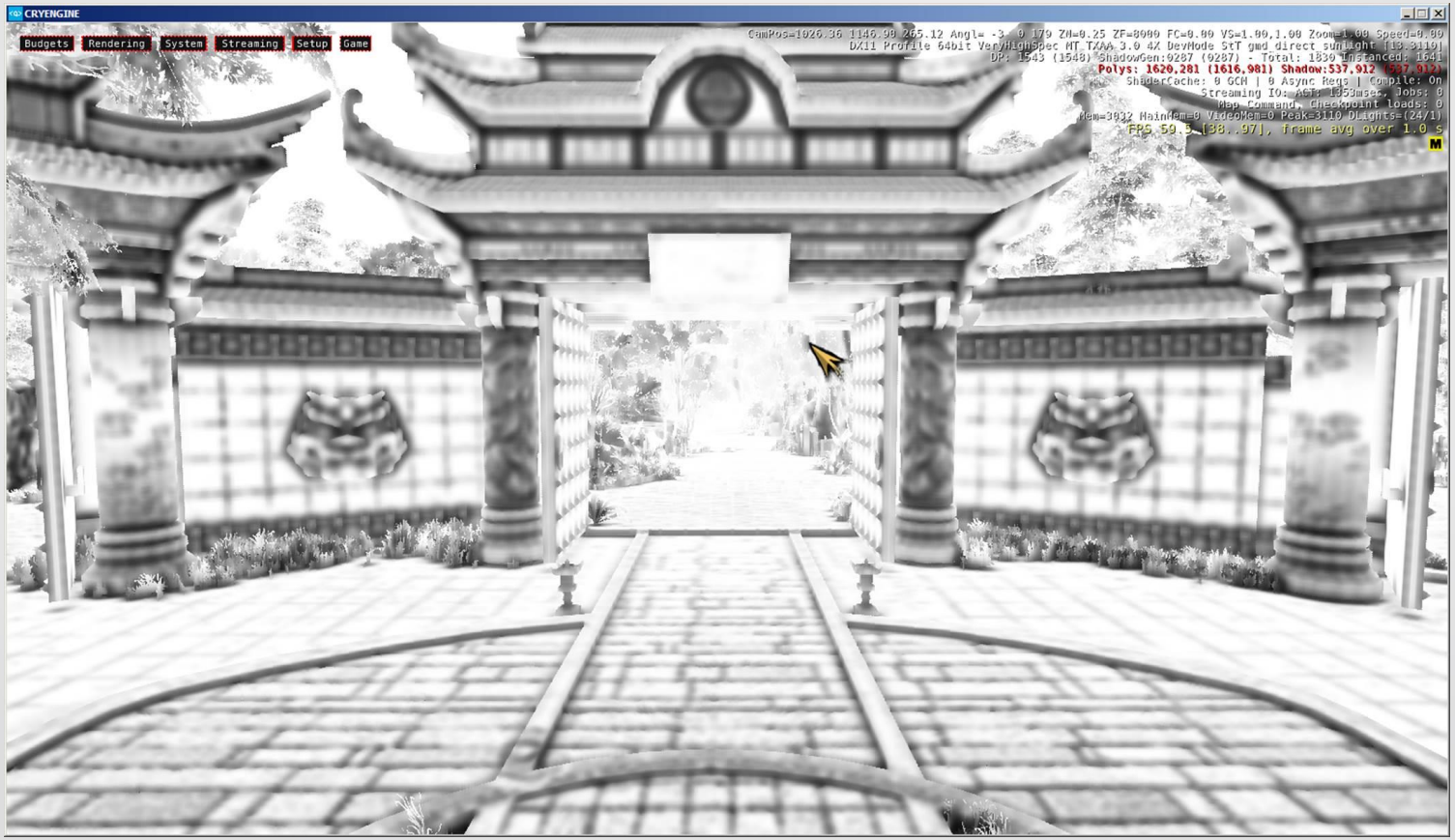
NVIDIA

NVIDIA

TXAA 3.0
(4xTXAA+PSL)

NVIDIA

HBAO+



NVIDIA

HBAO+的特点

- 图形质量比HBAO更好
  - HBAO会产生过度遮挡的问题
- 比HBAO的效率更高
  - 减少了算术指令和贴图指令，
  - 交错渲染，拥有极高的贴图缓存命中率
- 在全屏大小上计算SSAO，而不是半尺寸
  - 在半尺寸上计算SSAO会带来闪烁的问题，特别是较细的物体(如一些带Alpha-test的表面)
- 易于集成

NVIDIA

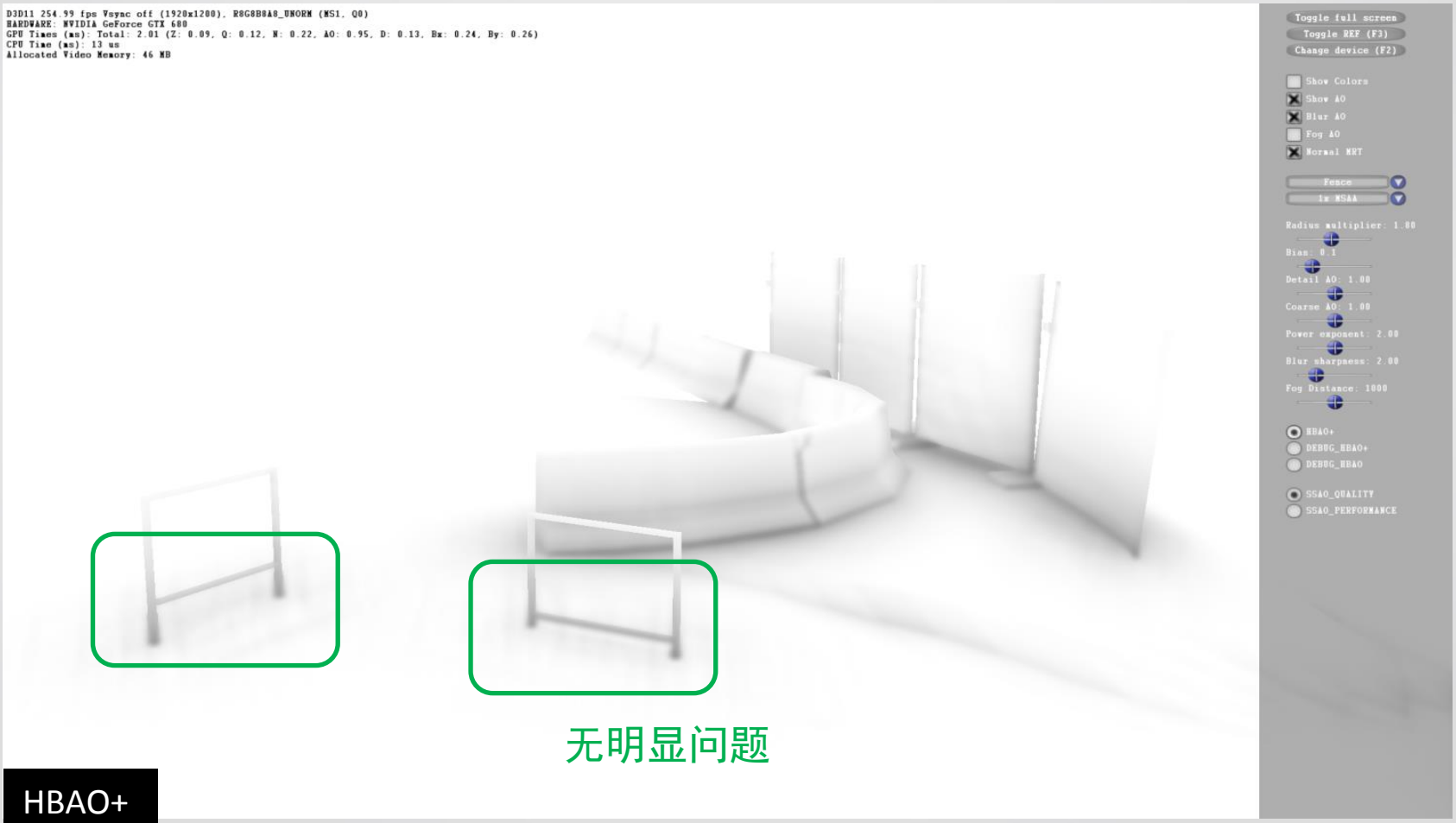
HBAO的图形质量

过度遮挡及闪烁

HBAO

NVIDIA

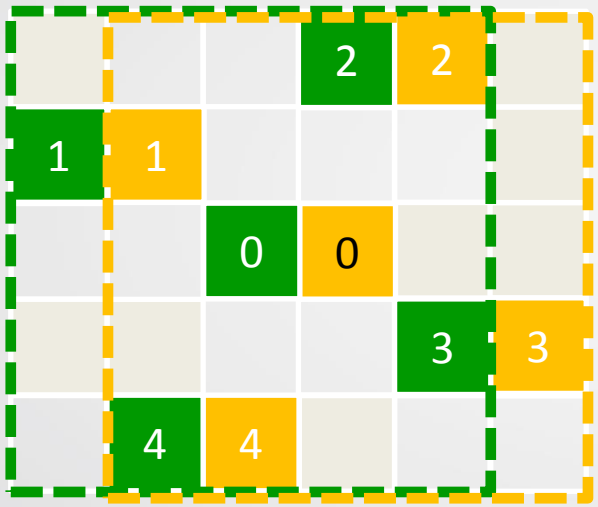
HBAO+的高图形质量



无明显问题

HBAO+

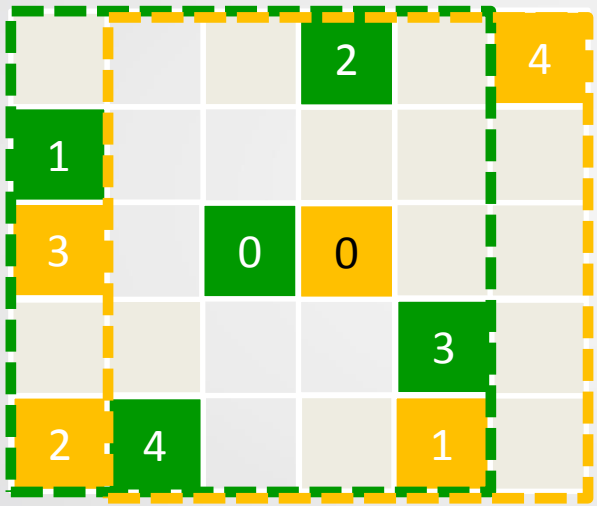
固定采样模式(Fixed Sampling Pattern)



对于每次采样,
相邻的像素读取
相邻的纹理元素(texel)

➔ 良好的空间局部性(Good spatial locality) ☺

随机采样模式(Random Sampling Pattern)

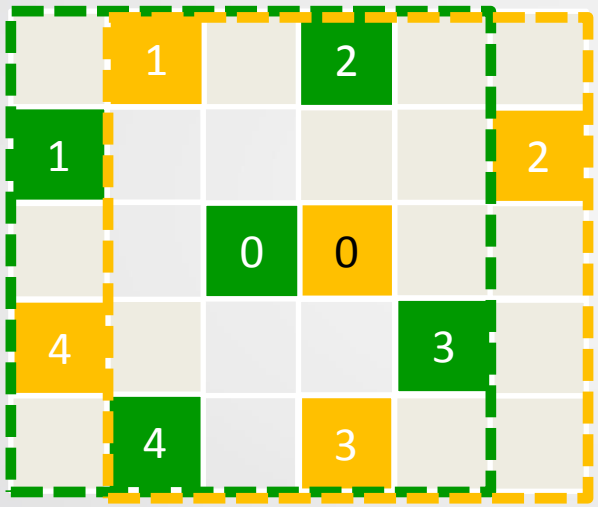


对于每次采样,
相邻的像素读取
相隔遥远的纹理元素(texel)

➔ 空间局部性不佳(Poor spatial locality) ☹

抖动采样模式(Jittered Sampling Pattern)

NVIDIA



对于每次采样，
相邻的像素读取
相同区域中的纹理元素(texel)

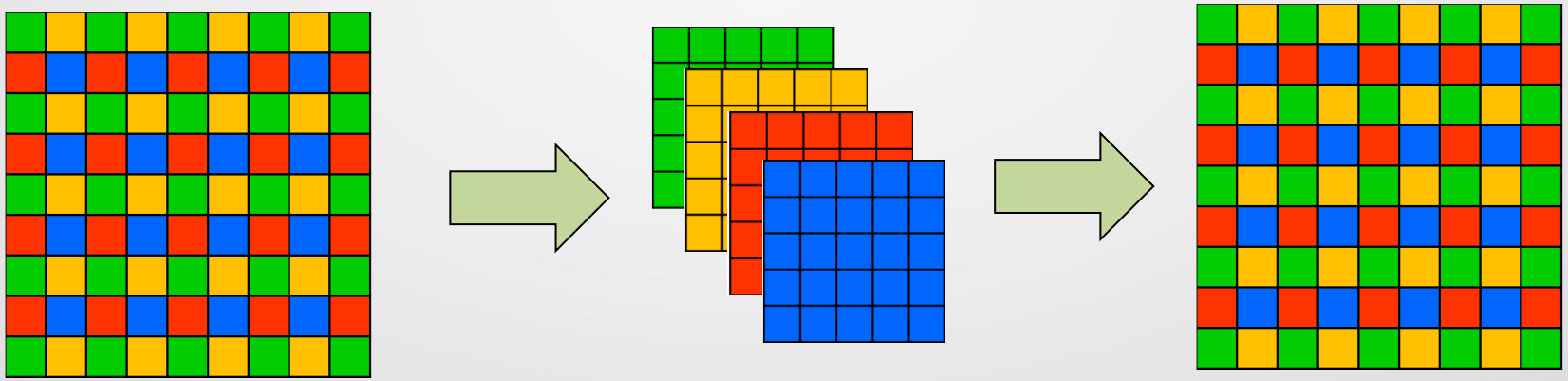
➔ 较好空间局部性(Better spatial locality)

… 但是如果过滤核尺寸变大，区域尺寸也会变大 ☹

交错渲染(Interleaved Rendering)

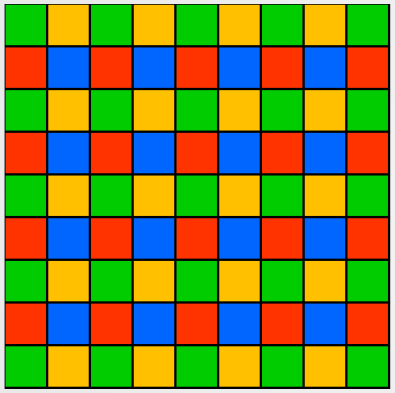
NVIDIA

对降采样后的(downsampled)输入贴图，分别使用各自的采样模式(sampling pattern)进行渲染



NVIDIA

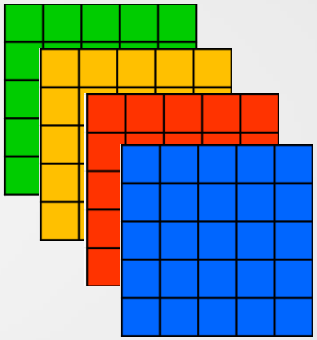
步骤1：将输入数据交错拆分 (Deinterleave Input)



全尺寸的输入贴图

Width = W
Height = H

1 Draw call
with 4xMRTs



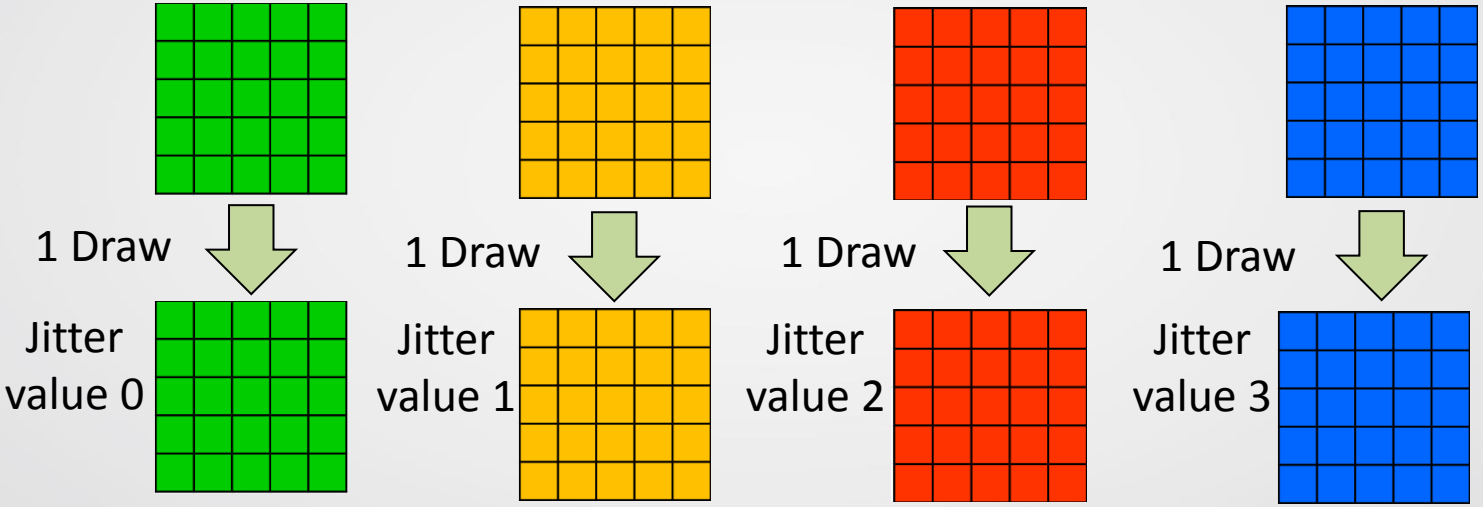
半尺寸2D贴图数组

Width = iDivUp(W,2)
Height = iDivUp(H,2)

NVIDIA

步骤2：无抖动采样

输入: 贴图数组A (slices 0,1,2,3)

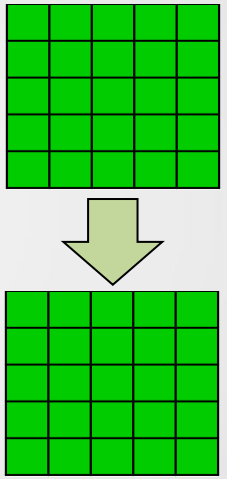


输出: 贴图数组B (slices 0,1,2,3)

步骤2：无抖动采样

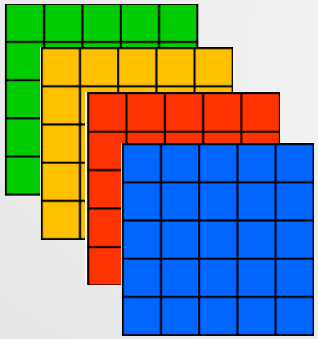
1. 每个Draw Call使用的固定的抖动参数
   ➔ 有利于提高采样点的分布局部性(Locality)

2. 每个Draw Call使用低分辨率的输入贴图
   ➔ 降低了内存带宽需求

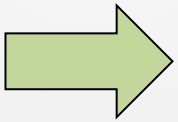


NVIDIA

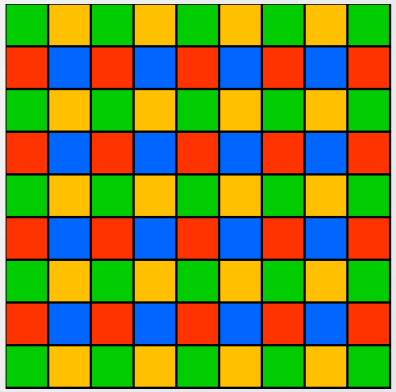
步骤3：把结果交织复原



1 Draw call



With 1 Tex2DArray
fetch per pixel



4x4 交错渲染(Interleaving)

NVIDIA

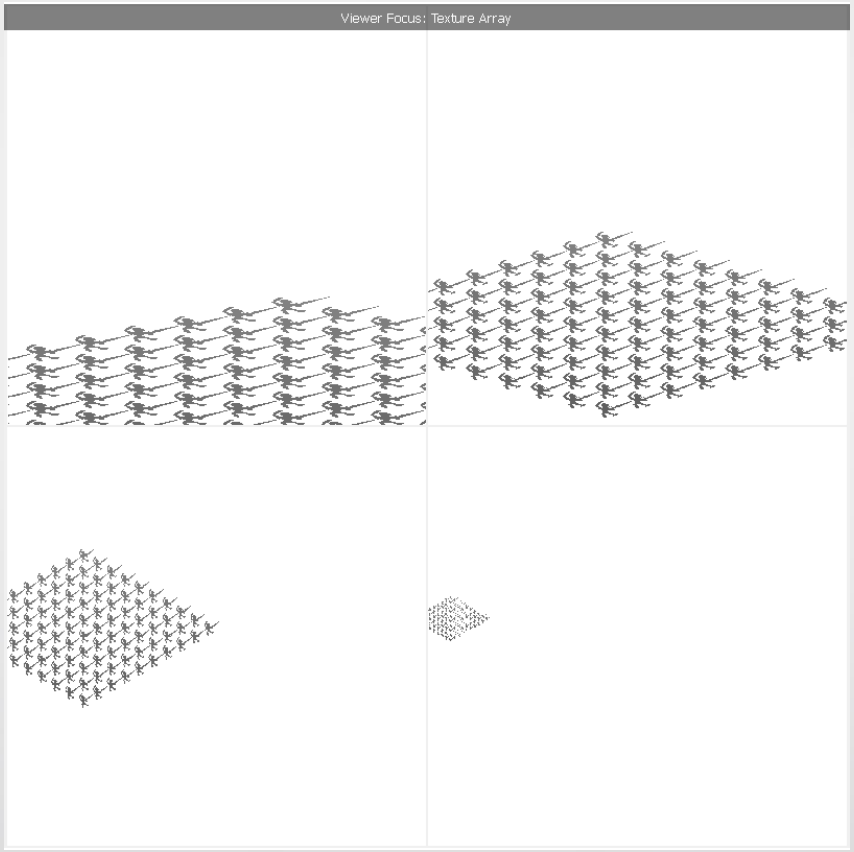
我们经常使用4x4的抖动贴图来抖动较大且分散的过滤核(large sparse filter)

可以使用4x4的交错渲染管线

1. **解交错(Deinterleaving):** 2 Draw calls with 8xMRTs
2. **采样计算(Sampling):** 16 Draw calls
3. **交错复原(Interleaving):** 1 Draw call

NVIDIA

多投影加速(Multi-Projection Acceleration)



NVIDIA

Fast GS vs Regular GS

- Fast GS是一种特殊的geometry shader
- Fast GS不能“生成”新的图元
- Fast GS节省了几何体生成的开销

NVIDIA

快速多视口投射(Fast Viewport Multi-casting)

NVIDIA

使用场合

- 适用于我们仅需通过GS设置图元属性但不需要改变图元拓扑结构的情形
- 渲染Cube-Map
- 体素化
- 多分辨率着色(用于VR渲染)
- Cascaded Shadow Maps

NVIDIA

使用Fast GS实现CSM

- 只需渲染一遍场景即可生成所需的所有阴影贴图，从而节省CPU开销
- 使用一个包含各层阴影贴图渲染范围的大视锥体进行渲染，通过设置不同的viewport来完成各层级阴影贴图的渲染

LOD=2

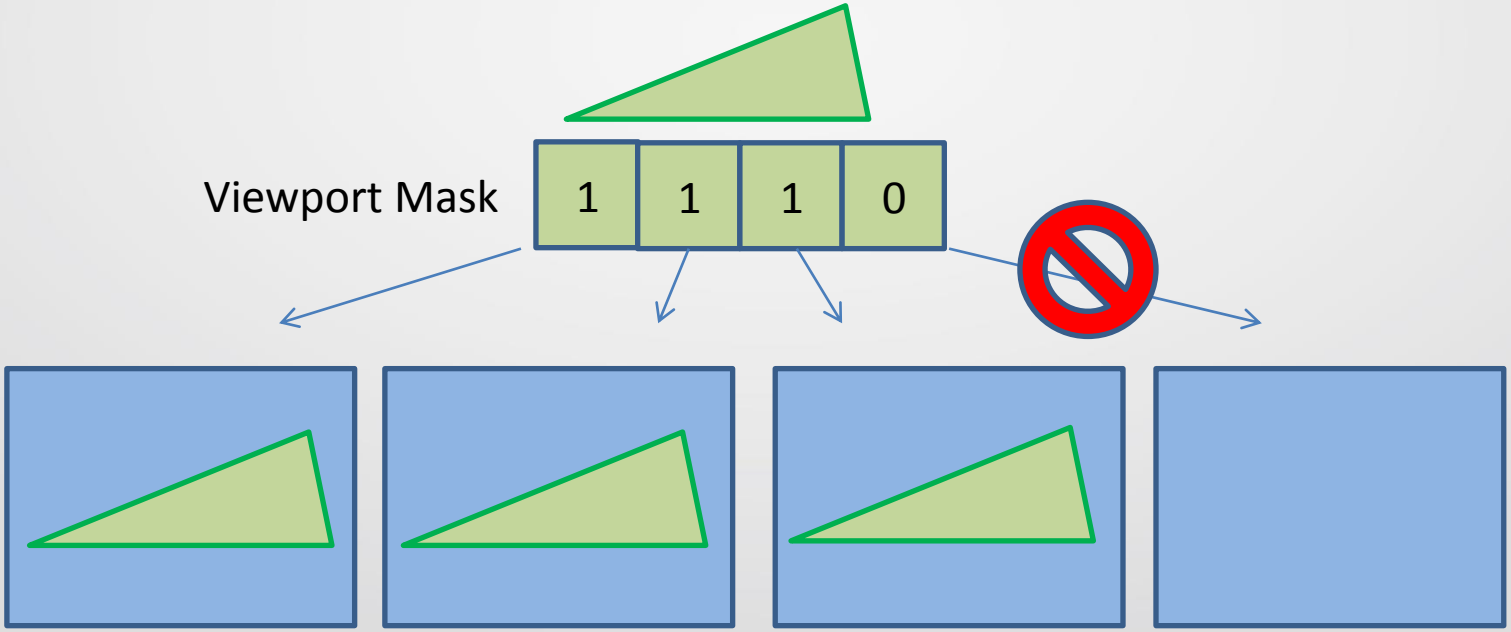
LOD=1

LOD=0

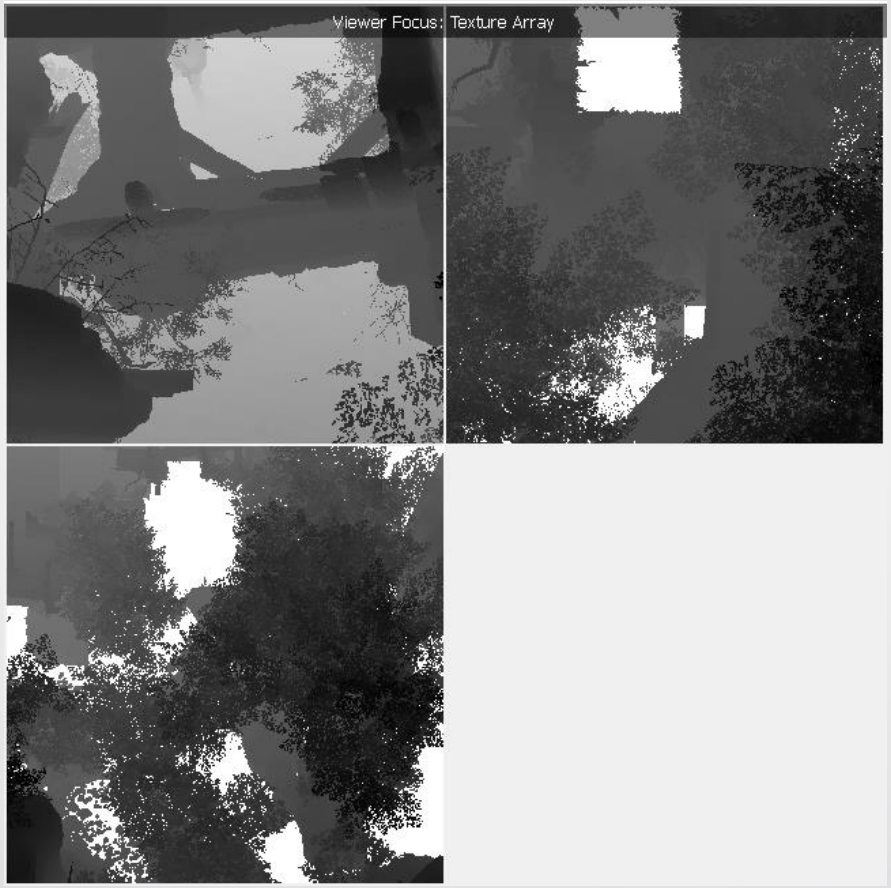
NVIDIA

使用Fast GS实现CSM

- 在GS中做视图剪切
  - 通过控制Viewport Mask上的标志位设置，我们可以在GS中视图剪切，把几何体投射到需要渲染的视图上去
  - 如果Viewport Mask为0，则该几何体被剔除

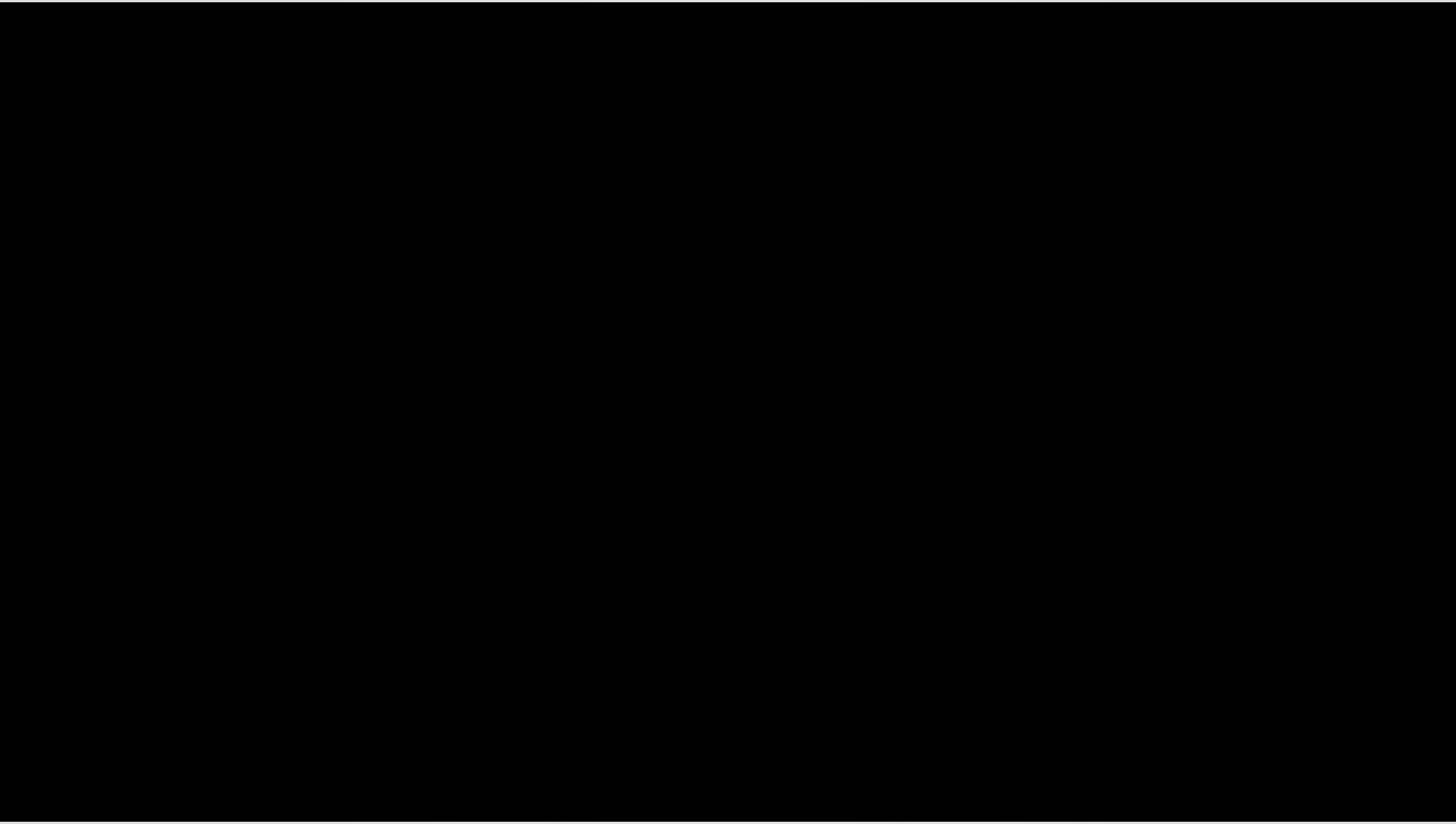


使用Fast GS实现CSM



NVIDIA

HairWorks & Clothing



NVIDIA

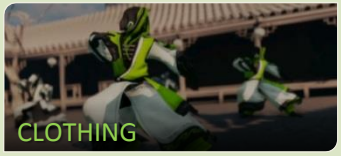
总结

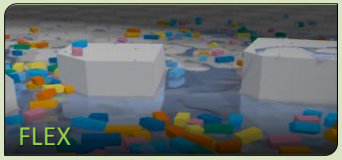
NVIDIA

总结

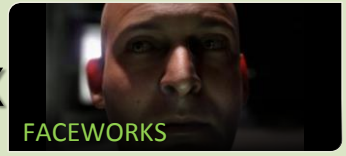
NVIDIA GAMEWORKS™

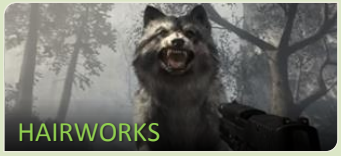
PhysX

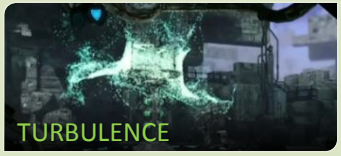


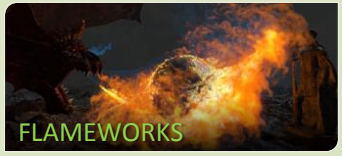


VisualFX

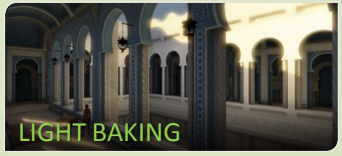








OptiX



Samples

NVIDIA

总结

- NVIDIA GameWorks
  - 提供了诸多易于集成的高质高效的图形及物理效果库
  - 提供各种例程，文档教程，利于游戏开发者学习掌握新技术
  - 提供了多种开发工具
  - 为游戏开发提供了极大的便利

  https://developer.nvidia.com/gameworks

- GPU的新特性为高质高效的图形开发提供了硬件上的支持
  - 可以有更多的优化手段
  - 支持更多的新算法

谢谢！

提问？

youngy@nvidia.com


CGDC 2015 中国游戏开发者大会
CHINA GAME DEVELOPERS CONFERENCE


分享智慧
LET US SHARE