

高级 ESRAM 12 样本

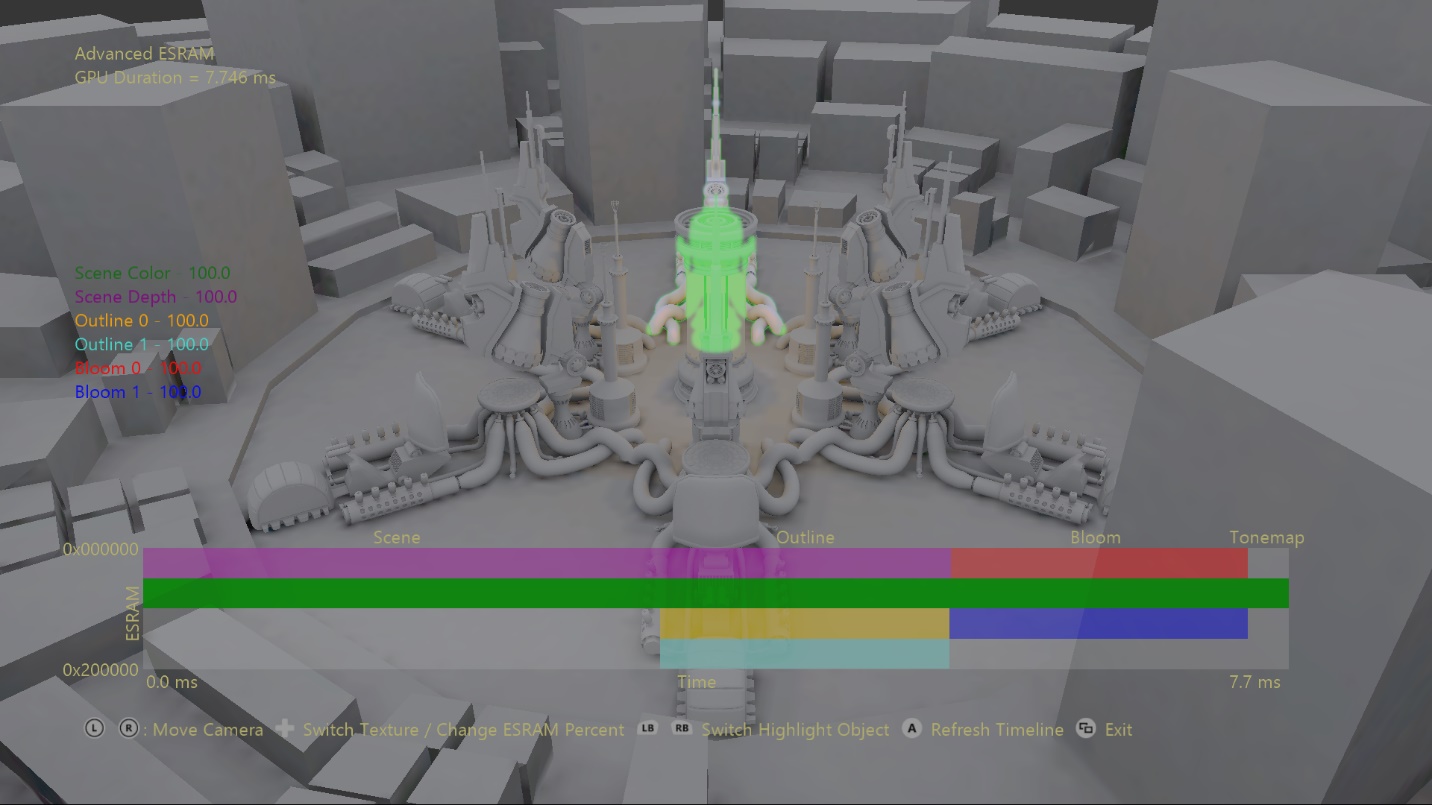
*\*此样本与 2016 年 10 月 Xbox One XDK 兼容*

# 描述

此样本演示了如何使用高级 DirectX 12.x 内存功能，以有效地为 D3D 资源添加别名内存。此样本核心的 API 为 ID3D12CommandQueue 的‘CopyPageMappingsX’和‘CopyPageMappingsBatchX’。这些功能提供在 GPU 时间线上将 CPU 页表条目复制到 GPU 的能力，让全虚拟化 D3D 资源能够映射至高速旋转的内存页。

使用此功能，样本实现了以 64 KB 页粒度分级 ESRAM 和 DRAM 的瞬态资源分配器。这使得帧的 GPU 内存使用保持最佳紧凑，用于充分利用 ESRAM 的全部功能。其接口镜像了 XG 内存库中 XGMemoryLayout 页面映射函数的重要方面。

注意：Xbox One X 没有 ESRAM，而是使用了带宽更高的 DRAM。在此主机上，样本将简单地渲染场景，并禁用所有 ESRAM 选项和可视化。



# 使用样本

该样本的主要功能使得在分配瞬态纹理资源的地方能够进行操作。框架中使用的资源是场景颜色、场景深度的纹理，两处用于轮廓，两处用于光晕。资源的 ESRAM 组成显示在最左侧，占资源内存的百分比。展示 ESRAM 布局的可视化提供了每个纹理的 ESRAM 和 DRAM 组成变化的即时可视反馈。每个资源的 ESRAM 覆盖区沿 Y 轴显示，其使用期则沿 X 轴可见。沿着时间轴使用的 GPU 计时可通过按下按钮进行刷新。

## 控制

|  |  |
| --- | --- |
| 操作 | 游戏手柄 |
| 将相机移向/远离原点 | 左控制棒上/下 |
| 环游相机 | 右控制棒 |
| 重设摄像机 | 右控制棒（点击） |
| 循环瞬态纹理 | D-Pad 左/右 |
| 更改 ESRAM 百分比 | D-Pad上/下 |
| 循环突显对象 | 左/右缓冲键 |
| 刷新时间轴 | A 键 |
| 退出 | 查看按键 |

# 实施说明

创建大型虚拟地址空间 (VAS) 以对从 ESRAM 和 DRAM 分配的 64 KB 内存页（页面池）的块进行映射。页面池使用 ID3D12Device 的‘RegisterPagePoolX’函数在 DirectX12 中进行注册 - 页面池不再使用时，必须使用‘UnregisterPagePoolX’进行注销。此映射用于暂存 CPU 页表条目以直接复制到 GPU 页表。

ID3D12CommandQueue 的‘CopyPageMappingsX’或‘CopyPageMappingsBatchX’功能使这些页面池中的页面范围能够映射到 GPU 时间轴上的指定 GPU 虚拟地址。这让全虚拟化 D3D 资源能够灵活映射到高速旋转的 64 KB 物理页面上。这种能力使资源之间的内存别名变得微不足道，整体都变得很有趣！

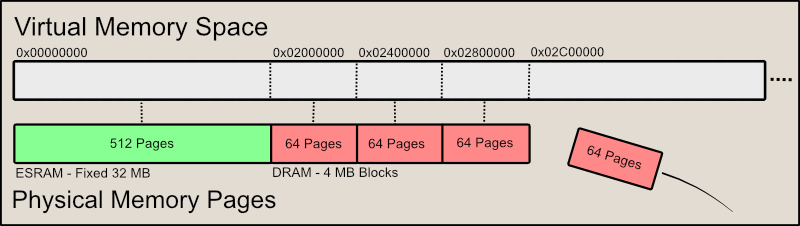


图 1：样本中使用的内存映射范型。ESRAM（如适用）映射到 VAS 的前 32 Mb，同时根据需要附加 4 MB DRAM 页面池。

页面块的创建和管理由 PageAllocator 类执行。提供虚拟地址范围，分配器根据需要按顺序将页面池映射到此范围。然后使用‘RegisterPagePoolX’在 DirectX12 注册页面池。完全追踪分配器页面的使用 - 从头到尾分配页面，在释放回页面后替换页面。

TransientCache 负责管理全虚拟化 D3D 资源。这些均按需创建，但经过缓存以避免重新创建公共资源的不必要开销。缓存这些资源的内存开销实际上为零，因为这些资源仅分配虚拟地址空间。每个资源每帧只能分配一次。

此 TransientAllocator 类使用页面分配器和瞬态缓存来满足用户的资源请求。请求资源时，将从 TransientCache 中获取实例。然后，从 PageAllocators 分配所需数量的页面，解析标记以确定是否以页面级粒度使用 ESRAM 或 DRAM。然后生成适当的结构，以便稍后供应给‘CopyPageMappingsBatchX’，也就是 D3D12XBOX\_PAGE\_MAPPING\_BATCH 和 D3D12XBOX\_PAGE\_MAPPING\_RANGE 结构的向量。

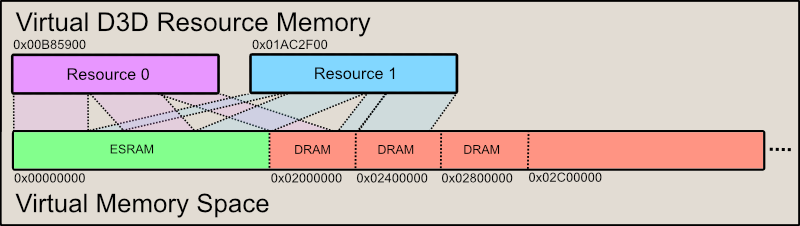


图 2：虚拟 D3D 资源将映射到页面池中的页面范围，以满足其内存需求。这些映射为 CopyPageMappingsX 和 CopyPageMappingsBatchX 调用的结果。为了视觉上的简化，在此可视化中仅使用两个资源，并且没有内存具有别名。但是，内存别名是这种技术的预期好处。

由于内存别名，TransientAllocator 还负责在必要时执行着色器和缓存刷新。在 DirectX12 中，插入刷新作为资源障碍的一部分。由于我们已绕过这个系统来执行内存别名，必须手动插入自己的刷新。TransientAllocator 通过检查资源的关联视图来确定应刷新哪些着色器阶段和缓存。

最后，在使用分配瞬态资源的命令列表提交到其命令队列之前，必须在 TransientAllocator 上调用“完成”以完成资源映射。此时，‘CopyPageMappingsBatchX’调用将置于命令队列中，设置要在后续命令列表中使用的资源内存映射。

# 更新历史记录

8/6/2018 - 样本创建。

# 隐私声明

编译和运行示例时，示例可执行文件的文件名将发送给Microsoft以帮助跟踪示例使用情况。要选择退出此数据收集，您可以删除Main.cpp中标记为“Sample Usage Telemetry”的代码块。

有关 Microsoft 隐私政策的更多信息，请参阅 [Microsoft 隐私声明](https://privacy.microsoft.com/zh-cn/privacystatement/)。