英特尔® Xe Super Sampling (XeSS) API 开发人员指南 v1.2

英特尔® Xe Super Sampling (XeSS) 是一项提升帧速率的创新技术，支持英特尔锐炫™ 显卡和其他厂商的显卡。XeSS 可以利用 AI 深度学习进行放大，在不降低图像质量的情况下提高帧速率。
对于想要优化游戏中图像质量和性能的游戏开发人员来说，了解 XeSS API 非常重要。

本开发人员指南旨在补充《XeSS API 参考指南》的内容。

目录

简介

英特尔® XeSS 作为 Microsoft DirectX 3D* 12 (DX3D) 计算着色器通道的一个序列实施，在渲染引擎的后期处理阶段之前执行（如标题为“TAA 与 XeSS”的章节所述）。渲染引擎通过将用于主渲染的 Direct3D* 12 (D3D12) 设备以及一个指针传递到一个描述符堆（XeSS 会在其中创建其所有内部资源描述符），来对 XeSS 进行初始化。XeSS 为以下两个类别之一分配 GPU 资源：

- 持久分配，例如网络权重和其他常量数据。
- 临时分配，例如网络激活。

游戏引擎可以通过将 XESS_INIT_FLAG_EXTERNAL_DESCRIPTOR_HEAP 初始化标志传递到 xessD3D12Init 调用，并传递一个在 xessD3D12Execute 调用中指向一个外部资源堆的指针，来控制 XeSS 进行临时分配的位置。当游戏引擎提供外部资源堆时，为了确保游戏最佳游戏性能，该堆应拥有高内存驻留优先级 。另一方面，持久分配始终归 XeSS 库所有。

XeSS 组件 简介

XeSS 可通过 XeSS SDK 获取，该 SDK 提供了基于 D3D12 的 API（用于集成到游戏引擎中），并包含以下 D3D12 组件：

- 一个可在任何支持 SM 6.4 的 GPU 上运行的基于 HLSL 的跨供应商实施。
  建议使用 DP4a 或同等的硬件加速
- 一个经过优化在英特尔锐炫™ 显卡和英特尔锐炬® Xe 显卡上运行的英特尔实施
- 一个实施调度程序，可加载游戏随附的 XeSS 运行时、与英特尔® 显卡驱动程序一起提供的版本或跨供应商实施

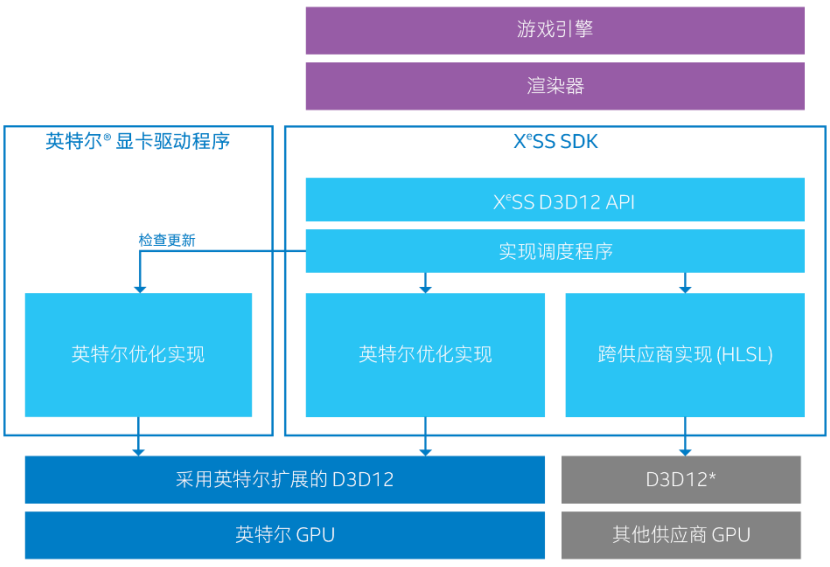

图 1. 英特尔专属解决方案和跨供应商解决方案的 XeSS SDK 组件

版本管理

对于开发阶段内部版本，XeSS 会使用 major.minor.patch 版本格式和 Numeric 90+ 方案。
XeSS 版本由 64 位函数 [xess_version_t] 结构指定，其中：

- 主要版本递增表示新的 API，并且可能会出现功能中断。
- 次要版本递增表示递增更改，例如可选输入或标志。这不会改变现有功能。
- 补丁版本递增可能包括针对已知问题的性能或质量调整或是修复。界面没有变化。
  90 以上的版本用于开发阶段内部版本，为下一个发行版改变界面。

XeSS 版本合并到 XeSS SDK 发行版中，可通过函数 xessGetVersion 获取。该版本包含在 zip 文件和随附的自述文件中，以及代码示例的标题中。

兼容性

未来所有英特尔® 发布的显卡驱动程序都会支持 XeSS 最新版本和历史版本。

具体来说，在英特尔® 平台上，加载器将根据以下规则运行：

- 加载器会检查与游戏一起安装的 XeSS 版本与系统上安装的驱动程序的兼容性。
- 如果兼容，则加载器会使用通过游戏安装的 XeSS 版本。
- 如果不兼容，且驱动程序较新，则加载器会忽略游戏版 XeSS，并使用通过驱动程序分发的版本。
- 如果不兼容且驱动程序较旧，则加载器会返回失败代码，并且 XeSS 不会初始化。

命名规范

XeSS API 使用以下命名规范：

- 所有函数必须以 xess 为前缀
- 所有函数必须使用骆驼式命名法 xessObjectAction 规范
- 所有宏必须使用全部大写 XESS_NAME 规范
- 所有结构、枚举和其他类型必须遵循 xess_name_t 蛇形命名规范
- 所有结构成员和函数参数必须使用骆驼式命名规范
- 所有枚举值必须使用全大写 XESS_ENUM_ETOR_NAME 命名规范
- 所有句柄类型必须以 handle_t 结尾
- 所有参数结构必须以 params_t 结尾
- 所有属性结构必须以 properties_t 结尾
- 所有标志枚举必须以 flags_t 结尾

TAA 和 XeSS

XeSS 是一种时域分摊超级采样/上采样技术，可取代游戏渲染器中的时间性抗锯齿 (TAA) 阶段，实现显著优于游戏领域当前先进技术的图像质量。

下图显示了一个包含 TAA 的渲染器。该渲染器会在每一帧中抖动相机，对屏幕空间中的不同坐标进行采样。TAA 阶段会在时间上累积这些样本，生成超采样图像。在累积之前，会使用渲染器生成的运动矢量对先前累积的帧（历史）进行扭曲，以将其与当前帧对齐。遗憾的是，由于可见性的帧到帧变化以及运动矢量的着色或错误，扭曲后的样本历史可能与当前像素不匹配。这通常会导致重影伪影。TAA 实施采用启发式算法（例如邻域钳制）来检测不匹配问题并拒绝相应的历史。然而，这些启发式算法经常失败，并产生大量重影、过度模糊或闪烁。

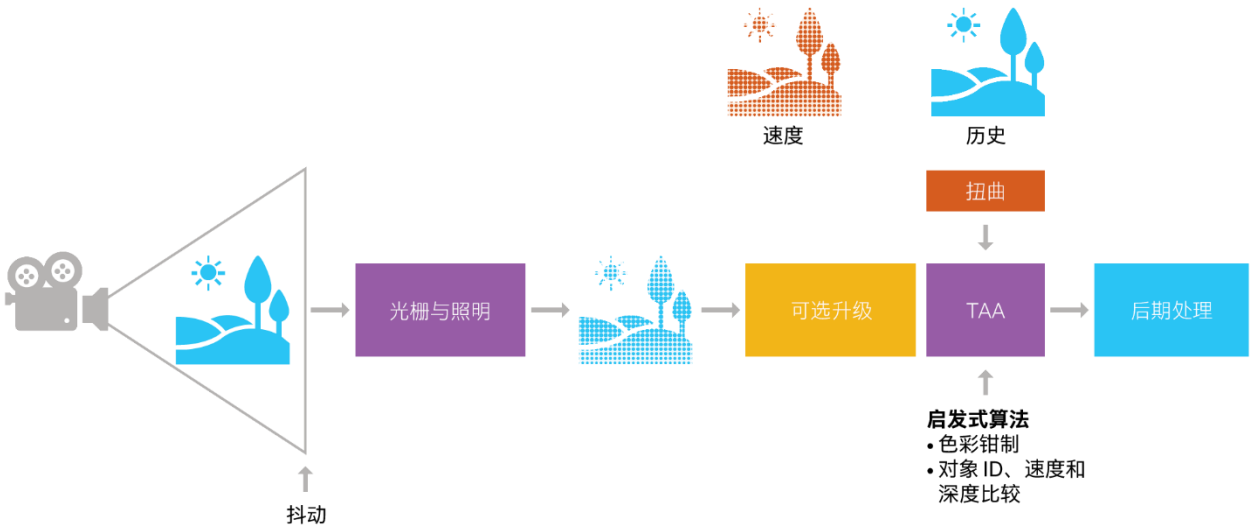

图 2. 显示包含 TAA 的典型渲染管道的流程图

如下所示，XeSS 用基于神经网络的方法替换了 TAA 阶段，输入和输出与 TAA 相同。有关 TAA 技术的概述，请参阅此报告。

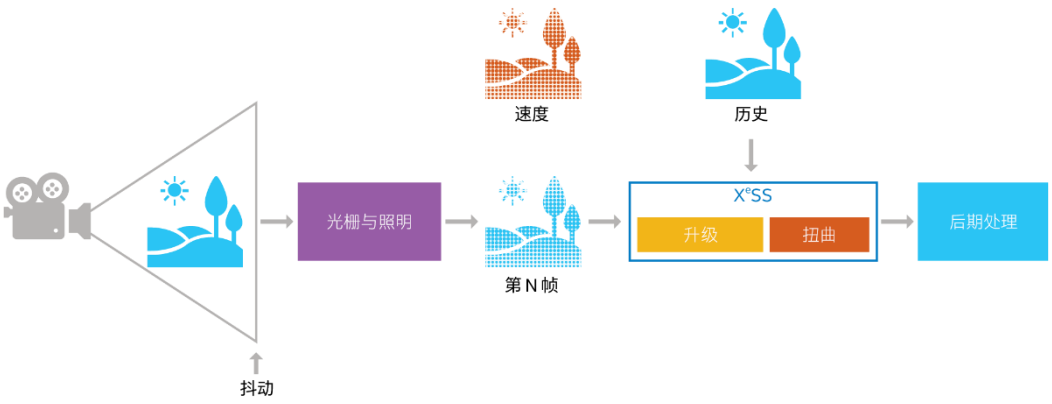

图 3. 在渲染管道中引入 XeSS

XeSS 游戏设置建议

将 XeSS 集成到您的游戏中时，请务必让游戏遵循这些指南，以便在修改 XeSS 选项时，用户能够获得一致的体验。

下面还有关于 XeSS 功能字体、官方命名和描述的指南。

英特尔 Xe Super Sampling 品牌命名规范

游戏开发人员需要在其设置菜单和描述中遵循经批准的 XeSS 命名规范。对于 XeSS 中较小的 *e*，可以缩小该字符的字体大小以保持比例。

XeSS 相关通信的官方字体是 IntelOneText-Regular。请在 XeSS 中使用官方版本中以上标字体显示的 e，除非字体系统不支持上标，在此情况下可以接受 XeSS。

标签	英特尔 XeSS
简短说明	英特尔 Xe Super Sampling (XeSS) 技术利用机器学习来提升性能并确保出色的图像质量。硬件加速 XeSS 针对基于 Xe-HPG 微架构的 GPU 进行了优化。
最短说明	英特尔 Xe Super Sampling (XeSS) 技术利用机器学习来提升性能并确保出色的图像质量。

游戏显卡设置菜单/游戏安装程序/启动器设置

游戏显卡设置应清楚显示 XeSS 选项名称，并允许用户选择质量/性能级别选项设置，如下所示。

预设	描述	建议分辨率
超高质量	侧重于提供最高质量视觉效果放大	1080p 及以上
质量	侧重于提供高质量视觉效果放大	1080p 及以上
平衡	侧重于实现最优的性能和图像质量	1080p 及以上
性能	侧重于提升整体游戏性能	1440p 及以上
关闭	关闭英特尔 XeSS	无

注意：启用 XeSS 时，您的游戏需要禁用其他画质放大技术，例如 DLSS 和 FSR，以及时间性抗锯齿（TAA）技术，以降低出现任何不兼容问题的可能性。

显卡预设默认建议

游戏菜单中默认选择的 XeSS 预设应基于用户设置的目标分辨率。以下条目是推荐的默认设置。

默认 XeSS 建议	描述	建议设置
特定于分辨率	您的游戏会根据输出分辨率调整 XeSS 默认预设	若分辨率为 1080p 及以下，则设置为“平衡”；若为 1440p 及以上，则设置为“性能”
一般	您的游戏会选择一个默认 XeSS 预设。	英特尔 XeSS ON 设置为“性能”

注意：所有英特尔 Xe Super Sampling 设置都应通过选择菜单（如果支持）向用户公开，以鼓励自定义。

英特尔 Xe Super Sampling 示例 UI

图 4. 具有 XeSS 设置的游戏 UI 示例

注：

- 屏幕分辨率：XeSS 支持 1080p 及以上分辨率
- 显示模式：XeSS 支持全屏、无边框窗口模式和窗口模式
- XeSS：关闭、性能、平衡、质量、超高质量
- 抗锯齿：如果英特尔 XeSS 被禁用，则抗锯齿模式应该会恢复之前的设置

开发

使用 XeSS 库进行构建时，您需要完成以下步骤：

- Add "inc" folder to include path
- Include xess.h and xess_d3d12.h
- Link with lib/libxess.lib

以下文件必须放置在可执行文件的位置或 dll 库搜索路径中

- libxess.dll

编程指导

输入和输出

XeSS 在每一帧都需要一个最小输入集：

- 抖动
- 输入颜色
- 扩张的高分辨率运动矢量

渲染器可以取代高分辨率运动矢量，提供输入分辨率下的运动矢量以及深度值：

- 非扩张的低分辨率运动矢量
- 深度

在后一种情况下，运动矢量的扩张和上采样将在 XeSS 内部进行。

抖动

XeSS 是一种时间性超采样技术，每一帧的投影矩阵均需要应用亚像素抖动偏移 *(J_x,J_y)*。此过程本质上会在每一帧产生一个新的亚像素样本位置，并且即使在静态场景中也能保证时间收敛。抖动偏移值应在 *[-0.5,0.5]* 范围内。可以通过向相机投影矩阵添加剪切变换来应用此抖动：

```
ProjectionMatrix.M[2][0] += Jx * 2.0f / InputWidth
ProjectionMatrix.M[2][1] -= Jy * 2.0f / InputHeight
```

如下所示，应用于相机的抖动会导致帧中的采样点发生位移，其中目标图像的宽度和高度放大 2 倍。
请注意，有效抖动对 *(J_x, J_y)* 无效，因为投影矩阵会被应用于几何形状并与相机负抖动对应。

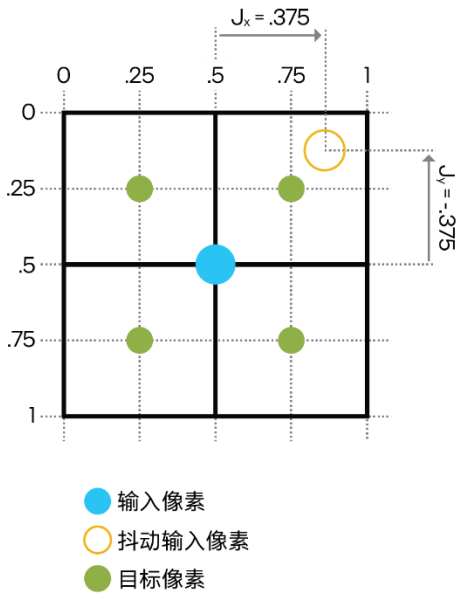

图 5. 采样点的抖动位移

抖动序列

要想让 XeSS 算法实现最佳质量，就需要采用具有良好空间特征分布的准随机采样序列。Halton 序列是不错的选择。使用这种序列修改重复模式的长度时，应该考虑到缩放因子。例如：如果游戏在原生渲染中使用长度为 8 的 Halton 序列，那么在与 XeSS 放大一起使用的情况下，它必须变为 *8 * scale^2*，以确保样本在单个低分辨率像素覆盖的区域中的良好分布。有时，进一步增加长度可以让图像质量得到进一步提升。我们鼓励用户在序列长度上进行尝试。必须避免对于输入像素的抖动样本分布造成偏差的重要性采样技巧。

颜色

XeSS 接受采用任何线性颜色格式的 SDR 和 HDR 输入颜色，例如：DXGI_FORMAT_R16G16B16A16_FLOAT、DXGI_FORMAT_R11G11B10_FLOAT、DXGI_FORMAT_R8G8B8A8_UNORM 等。输入颜色应在 scRGB 颜色空间中，该颜色空间以场景为参考，即颜色值表示亮度级别。(1.0,1.0,1.0) 的值将 D65 白色编码为 80 尼特，表示 SDR 显示器的最大亮度。HDR 内容的颜色值可以超出 (1.0,1.0,1.0)。

如果输入颜色尚未对曝光度进行调整，或者输入颜色的范围与 sRGB 空间不同，则可以通过以下方式提供单独的调整：

- 在没有可用的输入曝光值的情况下，只要初始化时使用 XESS_INIT_FLAG_ENABLE_AUTOEXPOSURE 标志即可自动计算。但请注意，使用此标志将会对性能产生一定的影响。
- 在未提供 XESS_INIT_FLAG_ENABLE_AUTOEXPOSURE 标志时，可以使用一个在 CPU 上记录命令列表期间所提供的输入曝光值，或者是一张由 GPU 更新的记录了曝光值的纹理。

这些曝光值以如下方式应用：

```
if (autoexposure)
{
  scale = XeSSCalculatedExposure(…)
}
else if (useExposureScaleTexture)
{
  scale = exposureScaleTexture.Load(int3(0, 0, 0)).x
}
else
{
   scale = inputScale
 }
inputColor *= scale
```

输出与输入在相同的颜色空间中。它可以是任何与输入类似的三通道或四通道线性颜色格式。
如上所示，如果将某个比例值应用于输入，则该比例的反值会应用于输出颜色。XeSS 会维护一个内部历史状态来执行传入样本的时间累积。这意味着如果场景或视图突然发生变化，则应该丢弃历史记录。这需要通过在 xess_xxx_execute_params_t 中传递 historyReset 标志的设置来实现。

运动矢量

运动矢量以像素为单位指定从前一帧到当前帧的屏幕空间运动。XeSS 接受格式为 DXGI_FORMAT_R16G16_FLOAT 的运动矢量，其中 R 通道在 x 中编码运动，而 G 通道在 y 中编码运动。运动矢量不包括由相机抖动引起的运动。运动矢量可以采用低分辨率（默认），也可采用高分辨率 (XESS_INIT_FLAG_HIGH_RES_MV)。低分辨率运动矢量由输入分辨率下的 2D 纹理表示，而高分辨率运动矢量由目标分辨率下的 2D 纹理表示。

对于高分辨率运动矢量，由相机动画产生的速度分量是使用相机变换和深度值在延迟通道中的目标分辨率下计算的。但是，与粒子和对象动画相关的速度分量通常以输入分辨率计算，并存储在 G-Buffer 中。对此速度分量进行上采样，并将其与相机速度相结合，可以生成高分辨率运动矢量的纹理。XeSS 还需要高分辨率运动矢

量发生扩张，即运动矢量表示输入像素的小邻域（例如 3 * 3）中最重要表面的运动。
用户可以在单独的通道中计算高分辨率运动矢量。

低分辨率运动矢量不会扩张，而是直接表示每个抖动像素位置的采样速度。XeSS 在内部将运动矢量上采样到目标网格，并使用深度纹理来扩张运动矢量。下图显示了使用低分辨率和高分辨率运动矢量指定的同一运动。

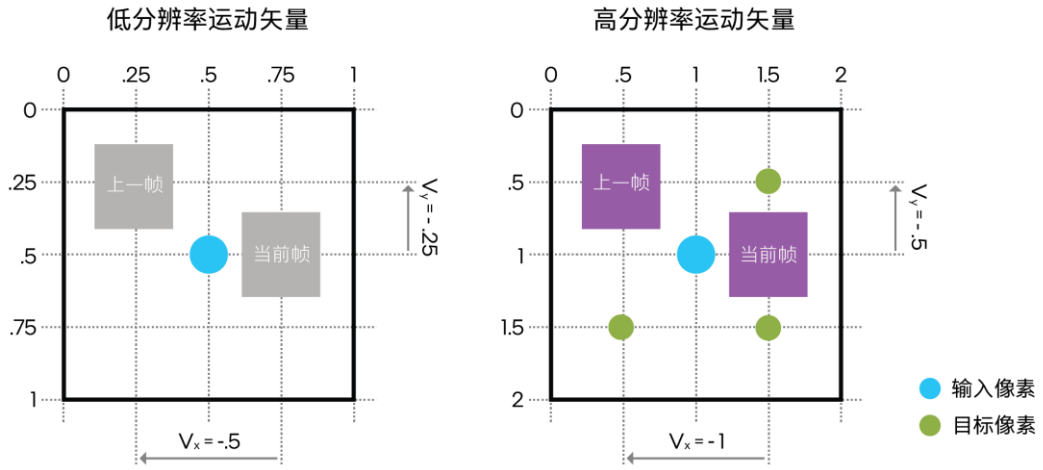

图 6. 为 XeSS 指定低分辨率和高分辨率运动矢量的规范

一些游戏引擎仅将对象运动渲染到 G-Buffer 中，并在 TAA 着色器中即时计算相机速度。在这种情况下，在 XeSS 执行之前需要一个额外的通道来合并对象速度和相机速度，并生成一个扁平化速度缓冲区。在此类场景中，高分辨率运动矢量可能是更好的选择，因为可以在目标分辨率下执行平展通道。

深度

如果与低分辨率运动矢量一起使用，XeSS 还需要深度纹理来进行速度扩张。支持任何深度格式，
例如 D32_FLOAT 或 D24_UNORM。默认情况下，XeSS 假定较小的深度值与相机距离较近。
但是，一些游戏引擎采用倒置深度，这可以通过设置 XESS_INIT_FLAG_INVERTED_DEPTH 来启用。

响应式像素掩模

用户可以提供一个掩模值为 1 的响应像素掩模，以强制 XeSS 忽略来自先前帧的信息。尽管 XeSS 是一项通用技术，应该可以处理多种渲染场景，但在极少数情况下，没有有效运动矢量的对象可能会产生伪影，例如粒子。在此类情况下，可以为这些对象设置响应像素掩模。只要掩模值在 R 通道中，任何纹理格式都可用于掩模。

资源状态

XeSS 需要所有输入纹理都处于状态 D3D12_RESOURCE_STATE_NON_PIXEL_SHADER_RESOURCE，输出纹理处于状态 D3D12_RESOURCE_STATE_UNORDERED_ACCESS。

资源格式

XeSS 需要为所有输入纹理指定类型。对于无类型格式，XeSS 会根据下表进行转换：

输入格式	输出格式
DXGI_FORMAT_R32G32B32A32_TYPELESS	DXGI_FORMAT_R32G32B32A32_FLOAT
DXGI_FORMAT_R32G32B32_TYPELESS	DXGI_FORMAT_R32G32B32_FLOAT
DXGI_FORMAT_R16G16B16A16_TYPELESS	DXGI_FORMAT_R16G16B16A16_FLOAT
DXGI_FORMAT_R32G32_TYPELESS	DXGI_FORMAT_R32G32_FLOAT
DXGI_FORMAT_R32G8X24_TYPELESS	DXGI_FORMAT_R32_FLOAT_X8X24_TYPELESS
DXGI_FORMAT_R10G10B10A2_TYPELESS	DXGI_FORMAT_R10G10B10A2_UNORM
DXGI_FORMAT_R8G8B8A8_TYPELESS	DXGI_FORMAT_R8G8B8A8_UNORM
DXGI_FORMAT_R16G16_TYPELESS	DXGI_FORMAT_R16G16_FLOAT
DXGI_FORMAT_R32_TYPELESS	DXGI_FORMAT_R32_FLOAT
DXGI_FORMAT_R24G8_TYPELESS	DXGI_FORMAT_R24_UNORM_X8_TYPELESS
DXGI_FORMAT_R8G8_TYPELESS	DXGI_FORMAT_R8G8_UNORM
DXGI_FORMAT_R16_TYPELESS	DXGI_FORMAT_R16_FLOAT
DXGI_FORMAT_R8_TYPELESS	DXGI_FORMAT_R8_UNORM
DXGI_FORMAT_B8G8R8A8_TYPELESS	DXGI_FORMAT_B8G8R8A8_UNORM
DXGI_FORMAT_B8G8R8X8_TYPELESS	DXGI_FORMAT_B8G8R8X8_UNORM
DXGI_FORMAT_D16_UNORM	DXGI_FORMAT_R16_UNORM
DXGI_FORMAT_D32_FLOAT	DXGI_FORMAT_R32_FLOAT
DXGI_FORMAT_D24_UNORM_S8_UINT	DXGI_FORMAT_R24_UNORM_X8_TYPELESS
DXGI_FORMAT_D32_FLOAT_S8X24_UINT	DXGI_FORMAT_R32_FLOAT_X8X24_TYPELESS

Mip Bias

在某些情况下，进一步增加 mip bias 可以进一步提升视觉质量，但由于内存带宽要求的增加，性能开销可能会随之增加，并可能降低时间稳定性，导致出现闪烁和波纹。用户可以自由尝试更激进或不太激进的纹理 LOD 偏差以找到适当的平衡。

在动态分辨率的场景中，建议在每次 XeSS 输入分辨率变化时更新 mip bias。如果无法在每次 XeSS 输入分辨率变化时更新 mip bias， 请考虑保留一组预估的 mip bias。

初始化

用户首先创建一个 XeSS 上下文，如下所示。在英特尔 GPU 上，此步骤会加载最新的英特尔优化 XeSS 实施。然后可以将返回的上下文句柄用于初始化和执行。

```
xess_context_handle_t context;
xessD3D12CreateContext(pD3D12Device, &context)
```

在初始化 XeSS 之前，用户可以请求管道预构建流程，以避免在初始化期间以高昂成本进行内核编译和管道创建。

```
xessD3D12BuildPipelines(context, NULL, false, initFlags);
```

然后调用 xessD3D12Init 函数来初始化 XeSS。在初始化期间，XeSS 可以创建暂存缓冲区并复制队列以上传权重。初始化结束时会销毁这些暂存缓冲区。XeSS 存储和层的限定由目标分辨率决定。因此，在初始化期间必须设置目标宽度和高度。

```
xess_d3d12_init_params_t initParams;
initParams.outputWidth = 3840;
initParams.outputHeight = 2160;
initParams.initFlags = XESS_INIT_FLAG_HIGH_RES_MV;
initParams.pTempStorageHeap = NULL;

xessD3D12Init(&context, &initParams);
```

XeSS 包括三种类型的存储：

- **独立于输出的持久存储**：持久存储，例如在初始化期间由 XeSS 在内部分配和上传权重。
- **依赖于输出的持久存储**：持久存储，例如内部历史纹理。
- **临时存储**：临时存储仅在 XeSS 执行期间具有有效数据。

临时存储可以在通过库管理的堆（默认）中内部分配，也可在 xess_d3d12_init_params_t 结构的 pTempStorageHeap 字段中由用户提供的堆中分配。如果用户分配了临时存储，则在 XeSS 执行之外可以重复使用该临时存储。

```
ComPtr<ID3D12Heap> pHeap;
CD3DX12_HEAP_DESC
heapDesc(xessProp.tempHeapSize,D3D12_HEAP_TYPE_DEFAULT);

d3dDevice->CreateHeap(&heapDesc, IID_PPV_ARGS(&pHeap));

initParams.tempStorageOffset = 0;
initParams.pTempStorageHeap = pHeap.Get();

xessD3D12Init(&context, &initParams);
```

用户可以指定 XESS_INIT_FLAG_EXTERNAL_DESCRIPTOR_HEAP 初始化标志，以便稍后在执行阶段使用外部描述符堆。

如果目标分辨率或任何其他初始化参数发生变化，用户可以重新初始化 XeSS。但是，待处理的 XeSS 命令列表必须在重新初始化之前完成。如果用户分配了临时 XeSS 存储，用户应取消分配或重新分配堆。可以自由更改质量预设，但更改任何其他参数可能会导致 xessD3D12Init 执行时间变长。

执行

XeSS 执行函数不涉及任何 GPU 工作负载，而是将 XeSS 命令记录到指定的命令列表中。然后由用户将命令列表加入队列。这意味着用户有责任确保所有输入/输出资源在 GPU 实际执行时都处于活动状态。

在默认情况下，XeSS 会创建一个内部描述符堆，但如果用户在初始化阶段指定了 XESS_INIT_FLAG_EXTERNAL_DESCRIPTOR_HEAP，则可以在执行参数中传递指向外部描述符堆的指针及其偏移。

如果在 xessD3D12Init 参数中指定了 EXTERNAL_DESCRIPTOR_HEAP 标志，则用户必须在与内部描述符相同的描述符堆中的相邻位置为输入和输出缓冲区创建描述符。外部描述符堆通过 xess_d3d12_execute_params_t 结构的 pDescriptorHeap 字段传递。DescriptorHeapOffset 应该指向 XeSS 描述符表。

固定输入分辨率

在默认情况下，XeSS 是根据所需的图像质量设置和目标分辨率使用固定输入分辨率。请调用 xessGetOptimalInputResolution 并使用 pInputResolutionOptimal 值来确定基于图像质量设置和目标分辨率的输入分辨率。在每次调用 xessD3D12Execute 时，必须提供实际的输入分辨率值作为 xess_d3d12_execute_params_t 架构的一部分。请留意，xessGetInputResolution 函数在 XeSS SDK 1.2 中已弃用，但为了兼容性已被保留。

动态输入分辨率

XeSS 支持同时使用动态输入分辨率和固定目标分辨率。在这种情况下，建议忽略 pInputResolutionOptimal 的值，并在支持范围内 [pInputResolutionMin; pInputResolutionMax] 自由更改输入分辨率，以确保应用程序在保持固定帧率的同时达到最佳的画质水平。在每次调用 xessD3D12Execute 时，您必须提供实际的输入分辨率值作为 xess_d3d12_execute_params_t 架构的一部分。当改变输入分辨率时，建议尽量将渲染分辨率的长宽比例保留相近或一致，以确保 XeSS 最佳和稳定的画质水平。

```
xess_d3d12_execute_params_t params;


params.jitterOffsetX    = 0.4375f;
params.jitterOffsetY    = 0.3579f;

params.inputWidth = 1920;
params.inputHeight = 1080;

// xess records commands into the command list
xessD3D12Execute(&context, pd3dCommandList, &params);

// Application may record more commands as needed
pD3D12GraphicsCommandList->Close();

// Application submits the command list for GPU execution
pCommandQueue->ExecuteCommandLists(1, &pCommandLists);
```

抖动比例

函数 xessSetJitterScale 将比例因子应用于抖动偏移。如果应用程序以像素以外的单位存储抖动，这可能会很有用。例如：NDC 抖动可以通过设置适当的比例转换为像素抖动。

速度比例

函数 xessSetVelocityScale 将比例因子应用于速度。如果应用程序以像素以外的单位存储速度，这可能会很有用。例如，可以通过设置适当的比例将标准化的视口速度转换为像素速度。

调试和日志记录功能

日志记录回调

XeSS SDK 提供了一个 API 来设置日志记录回调。请使用函数 xessSetLoggingCallback 来定义在以下情况下需要调用的函数：

- 可以从不同的线程调用回调。
- 可以同时从多个线程调用回调。
- 消息指针仅在函数内部有效，并且在返回调用后可能会立即变为无效。
- 消息是以 null 结尾的 UTF-8 字符串。

输入转储功能

XeSS SDK 提供了一个 API 来转储 SDK 输入、输出和历史状态。为了转储输入，应用程序应该调用函数 xessStartDump。由于内部实现，SDK 可以转储的帧数少于 xess_dump_parameters_t 结构的 frame_count 字段中提供的帧数。

推荐实践

视觉效果质量

强烈建议在色调映射之前在后期处理链的开头运行 XeSS。在某些场景下可以在色调映射之后执行；但此模式是实验性的，不能保证良好的质量。

为了最大限度提升图像质量，应注意以下事项：

- 对屏幕空间环境光遮蔽 (SSAO) 和阴影使用高质量或超高质量设置。
- 关闭所有用于降低着色率和渲染分辨率缩放的技术，例如可变速率着色 (VRS)、自适应着色、棋盘渲染、抖色等。
- 避免在 XeSS 放大之前使用四分之一分辨率效果。
- 不要依赖 XeSS 进行任何类型的去噪；充满噪声的信号会严重损害重建质量。
- 优先使用 FP16 精度作为场景线性 HDR 空间中的颜色缓冲区。
- 优先使用 FP16 精度作为速度缓冲区。
- 调整 mip bias 以最大限度提高图像质量并控制开销。
- 确保提供适当的场景曝光值。正确的曝光对于减少移动物体的重影、模糊度和精确的亮度重建至关重要。

驱动验证

为确保最佳画质和性能，建议安装最新的驱动程序。为了帮助实现这一点， 在使用 xessD3D12CreateContext 进行初始化后，建议调用 xessIsOptimalDriver 函数来验证系统里所安装的驱动程序是可以提供最佳体验的。如果该函数返回值是 XESS_RESULT_WARNING_OLD_DRIVER， 则建议通知用户更新驱动程序。XESS_RESULT_WARNING_OLD_DRIVER 不是致命错误，用户应被允许继续操作。

调试技巧

运动矢量调试

如果 XeSS 产生带有锯齿或晃动的图像，则有必要对静态场景进行调试：

- 在引擎中模拟帧之间的零时差，以保持完全静态的场景。
- 将运动矢量比例设置为 0，以排除运动矢量的潜在问题。
- 显著增加重复抖动模式的长度。

XeSS 应该生成高质量的超采样图像。如果没有这样，则说明抖动序列或输入纹理的内容可能存在问题；否则，很可能是运动矢量的解码出了问题。确保运动矢量缓冲区内容与当前设置的单位（NDC 或像素）相对应，且坐标轴方向正确。尝试将运动矢量比例因子加 1 或减 1，以恰当地对齐坐标轴。

抖动偏移调试

如果静态场景看起来不理想，请尝试将抖动偏移比例加 1 或减 1，以与坐标轴适当对齐。
确保抖动不会超出 [-0.5, 0.5] 界限范围。

其他资源

http://behindthepixels.io/assets/files/TemporalAA.pdf

英特尔锐炫™ 登陆页

DirectX 下载页面

声明

您使用本文档时不得涉及针对本文档所述英特尔® 产品的任何侵权分析或其他法律分析，也不得为他人以这种方式使用本文档提供便利。对于后续起草的包含本文所披露标的物的任何专利权利要求，您同意授予英特尔非独占、免版税许可。

性能因用途、配置和其他因素而异。请访问 www.intel.cn/PerformanceIndex 了解详情。

没有任何产品或组件能够做到绝对安全。

所有产品计划和路线图可能随时更改，恕不另行通知。

成本和结果可能会有所不同。

英特尔技术可能需要支持的硬件、软件或服务激活。

英特尔技术特性和优势取决于系统配置，并可能需要支持的硬件、软件或服务得以激活。实际性能可能因系统配置的不同而有所差异。请咨询您的系统制造商或零售商，也可登录 intel.cn 获取更多信息。

本文档未（明示、暗示、以禁止反言或以其他方式）授予任何知识产权许可。

英特尔不承诺任何明示或暗示的担保，包括但不限于对适销性、特定用途适用性和不侵权的暗示担保，以及由履约习惯、交易习惯和贸易惯例引起的任何担保。

本文包含尚处于开发阶段的产品、服务和/或流程的信息。此处提供的所有信息如有更改，恕不另行通知。如需获取最新的预告、时间表、规格和路线图，请联系您的英特尔代表。

所描述的产品和服务可能包含可能导致产品和服务与公布的技术规格有所偏离的瑕疵或误差，一经发现将被收入勘误说明。英特尔提供最新确定的勘误表备索。

如欲获取本文档提及的按顺序编号的文件副本，可致电 1-800-548-4725，或访问 www.intel.com/design/literature.htm。

Microsoft、Windows 和 Windows 标志是微软公司在美国和/或其他国家（地区）的商标或注册商标。