Intel® XeSS Frame Generation (XeSS-FG) 开发者指南 1.1

英特尔® XeSS Frame Generation(XeSS-FG) 是一项基于 AI 的帧插值技术，也叫帧生成技术，支持英特尔® Arc™图形显卡。XeSS-FG 使用深度学习生成高质量、高分辨率的游戏帧，在不降低图像质量的情况下提高了游戏性能和流畅度。

目录

介绍 4
配置要求 4
部署 5
命名规范和品牌指南 5
快速入门指南 6
编程指南 8
XeSS-FG 集成步骤 8
创建上下文 8
日志回调 9
XeLL 关联 9
初始化 9
初始化标志 11
UI 合成模式 12
UI 合成模式比较表 13
性能影响 13
错误处理 14
HDR 显示支持 14
启用帧生成 14
资源标记 15
输入和输出 16
运动矢量 17
HUD-less 颜色纹理 18

UI 纹理 18
帧常量 18
描述符堆（Descriptor Heap） 19
信箱效果（Letterboxing） 19
异步着色器管线构建 21
执行 21
Present 状态 21
更改窗口状态 22
关闭 22
与第三方帧生成软件的兼容性 22
调试 23
版本控制 23
声明 25

介绍

英特尔® XeSS 帧生成（XeSS-FG）由一系列计算着色器批次实现，在 Present 事件之前运行以生成额外的插值帧。XeSS-FG 提供了一个代理交换链(Proxy Swapchain)接口，把每一游戏渲染帧进行两次 Present 的复杂性抽象出来，以简化集成，同时保持平滑的 Present 节奏。应用程序通过传递 Direct3D* 12（D3D12）设备来初始化库，该设备用于主渲染。英特尔® XeSS-FG 可以与多种技术结合使用，以在集成和独立显卡平台上提供最佳体验。为了获得最佳性能，我们推荐在使用 XeSS-FG 之前游戏能达到以下帧率：

- **最低 40 FPS（XeSS-SR 或原生输入）：** 这提供了流畅游戏所需的基础，非常适合入门级显卡。
- **推荐 60 FPS（XeSS-SR 或原生输入）：** 这确保了最佳的延迟体验和最流畅的游戏体验，针对高端显卡。

XeSS-FG 支持所有的垂直同步（V-Sync）和刷新率模式（固定刷新率和可变刷新率 - VRR）。为了获得最佳效果，应该关闭或减少运动模糊（Motion Blur）以适应 XeSS-FG 产生的额外帧率。

配置要求

- DirectX 12
- Windows 10/11 x64 - 10.0.19043/22000 或更高版本
- 搭载 Intel® Xe Matrix 扩展（XMX）的 Intel® Arc™ 显卡
- Intel® 图形驱动 32.0.101.6252 或更高版本
- Intel® XeLL 1.1.0 或更高版本

部署

要在项目中使用 XeSS-FG：

- 将"inc" 文件夹添加到包含路径中
- 包含头文件 xefg_swapchain.h, xefg_swapchain_d3d12.h 和 xefg_swapchain_debug.h
- 链接 lib/libxess_fg.lib

以下文件必须放置在可执行文件旁边或 DLL 搜索路径中：

- libxess_fg.dll
- Microsoft Visual C++ Redistributable version 14.40.33810 或更高版本。需要以下库：
  - msvcp140.dll
  - vcruntime140.dll
  - vcruntime140_1.dll

命名规范和品牌指南

请参阅“XeSS 2 Naming Structure and Examples.pdf”以获取批准的命名规范、品牌指南和设置菜单示例。

快速入门指南

将 XeSS-FG 集成到应用程序中是一个简单而直接的过程。首先，应用程序必须确保其拥有所有必需的输入。如果您的应用程序支持 XeSS-SR 或 TAA，那么几乎所有必需的资源都已经存在。唯一 XeSS-FG 需要但 XeSS-SR 可选的资源是深度缓冲区。以下是所需的最小输入集：

- 运动矢量（Motion Vectors）
  - 必须从当前帧指向前一帧
  - R16G16_FLOAT 或类似格式
  - 可以在 NDC 或像素空间中
  - 可以是抖动过的
  - 优先选低分辨率（与超采样或 TAAU 输入相同），但也可以是高分辨率（目标分辨率）
- 深度缓冲区（Depth buffer）
  - 可以是反转的
  - 缓冲区大小必须与运动矢量缓冲区相同
- 帧常量（Frame constants）
  - 需要视图/投影（View/Projection）矩阵
  - 如果运动矢量是抖动过的，则可选输入抖动值
  - 可选的运动矢量缩放因子以匹配运动矢量的要求

重要提示! XeLL 集成是 XeSS-FG 集成的前提条件。应用程序必须在初始化 XeSS-FG 之前提供 XeLL 上下文给 XeSS-FG – 有关详细信息，请参见英特尔® XeLL 相关技术资料。

以下是将 XeSS-FG 集成到应用程序中的正确步骤：

- 创建 XeSS-FG 上下文
  - 可以在应用程序启动时完成
  - XefgSwapChainD3D12CreateContext
- 注册日志回调
  - xefgSwapChainSetLoggingCallback
- 关联到 XeLL context
  - xefgSwapChainSetLatencyReduction

- 初始化上下文
    - 应在 swap chain 创建后完成
    - xefgSwapChainGetProperties
    - xefgSwapChainD3D12InitFromSwapChain or xefgSwapChainD3D12InitFromSwapChainDesc
- 获取 XeSS-FG 的代理交换链指针
    - 必须使用这个代理交换链而不是应用程序的交换链
    - xefgSwapChainD3D12GetSwapChainPtr
- 标记资源
    - 每一帧必须标记每个必需的资源。
    - xefgSwapChainD3D12TagFrameResource
        - 需要 XEFG_SWAPCHAIN_RES_MOTION_VECTOR
        - 需要 XEFG_SWAPCHAIN_RES_DEPTH
    - xefgSwapChainTagFrameConstants
    - 如果在 XeSS-FG 初始化期间使用了 XEFG_SWAPCHAIN_INIT_FLAG_EXTERNAL_DESCRIPTOR_HEAP，则在 Present 之前必须调用 xefgSwapChainD3D12SetDescriptorHeap
- 执行帧显示 Present
    - 在 Present 之前必须调用 xefgSwapChainSetPresentId
- 应用程序可以暂停 XeSS-FG
    - xefgSwapChainSetEnabled
- 应用程序在终止之前应销毁 XeSS-FG 上下文。
    - xefgSwapChainDestroy
- 当 XeSS-FG 打开时应关闭或减少运动模糊效果

编程指南

XeSS-FG 集成步骤

XeSS-FG API 提供了 IDXGISwapChain 接口的一种实现，并包含了资源的生命周期管理和与英特尔® 图形驱动程序进行显示节奏（Display Pacing）交互的功能。XeSS-FG API 管理着一个内部命令列表，并使用应用程序提供的命令队列（Command Queue）提交 GPU 工作。

重要提示! XeLL 集成是 XeSS-FG 集成的前提条件。应用程序必须在初始化 XeSS-FG 之前提供 XeLL 上下文给 XeSS-FG – 有关详细信息，请参见英特尔® XeLL 相关技术资料。XeSS-FG 使用 XeLL 进行帧节奏调整，这对使用帧生成时减少延迟至关重要。

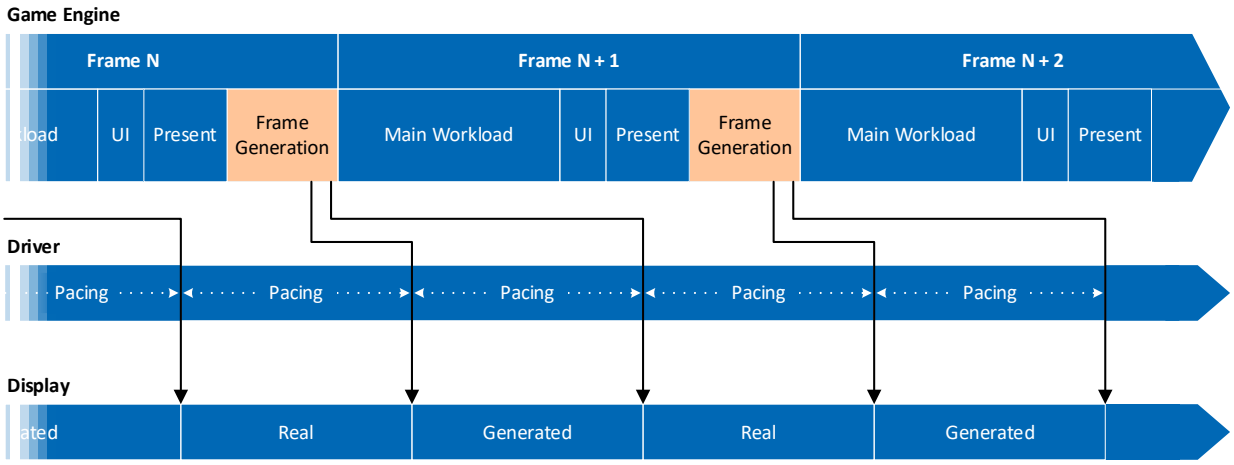


图 1. XeSS-FG 与应用程序管线的交互关系

创建上下文

要使用 XeSS-FG，应用程序必须调用 xefgSwapChainD3D12CreateContext 创建 XeSS-FG 上下文。

在创建上下文时，xefgSwapChainD3D12CreateContext 将检查所在GPU设备的必要功能，如果功能不足则会返回失败。应用程序可以通过调用 xefgSwapChainGetProperties 查询资源处理和插值选项的额外信息。

日志回调

应用程序可以使用 xefgSwapChainSetLoggingCallback 注册日志回调，获取额外的诊断信息：

```
static void LogCallback(const char* message,
      xefg_swapchain_logging_level_t level, void* userData);
xefg_swapchain_logging_level_t log_level =
      XEFG_SWAPCHAIN_LOGGING_LEVEL_WARNING;
xefgSwapChainSetLoggingCallback(xefgSwapChain, log_level, LogCallback,
userData);
```

日志记录有四个级别：

- XEFG_SWAPCHAIN_LOGGING_LEVEL_DEBUG
- XEFG_SWAPCHAIN_LOGGING_LEVEL_INFO
- XEFG_SWAPCHAIN_LOGGING_LEVEL_WARNING
- XEFG_SWAPCHAIN_LOGGING_LEVEL_ERROR

我们建议在生产中使用错误级别，在开发中使用警告或调试级别。

XeLL 关联

XeSS-FG 需要一个工作正常的 XeLL 集成。请参阅 XeLL 开发者指南。

重要提示! 如果没有提供 XeLL 上下文，XeSS-FG 将无法初始化，如果 XeLL 未初始化且启用，XeSS-FG 将被禁用。

可以调用 xefgSwapChainSetLatencyReduction 建立 XeSS-FG 和 XeLL 两个 API 之间的关联。应用程序必须在两个 API 中使用相同的帧计数器：

```
xefgSwapChainSetLatencyReduction(xefgSwapChain, xellContext);
```

请确保在整个 XeSS-FG 上下文生命周期中启用 XeLL。在帧生成禁用时允许禁用 XeLL，但在启用帧生成之前必须重新启用 XeLL。

初始化

有两种方法可以初始化 XeSS-FG：

- xefgSwapChainD3D12InitFromSwapChain

    - 从提供的交换链查询参数
    - 应用程序必须提供引用计数为 1 的交换链指针
    - 应用程序提供的交换链将被释放！如果无法释放交换链，函数将失败
    - 成功调用后，提供的交换链指针将无效
    - 为窗口创建新的交换链
    - 全屏独占模式下返回错误
- xefgSwapChainD3D12InitFromSwapChainDesc
    - 为窗口创建新的交换链
    - 应用程序不应有任何与窗口相关的交换链
    - 不会在全屏独占模式下失败，但帧生成无法启用

一旦 XeSS-FG 初始化完成，应用程序必须使用 xefgSwapChainD3D12GetSwapChainPtr 获取代理交换链的指针。

重要提示! 从此时起，所有与交换链的交互都必须只使用这个代理交换链！

XeSS-FG 将存储初始化期间提供的命令队列的引用，并使用此队列执行插值工作。应用程序必须确保在使用 XeSS-FG 交换链时此命令队列仍然存在。

```
ComPtr<IDXGISwapChain4> swapChain = CreateApplicationSwapChain();
xefg_swapchain_d3d12_init_params_t params = {};
params.pApplicationSwapChain = swapChain.Get();
params.maxInterpolatedFrames = 1;
xefgSwapChainD3D12InitFromSwapChain(
        xefgSwapChain, commandQueue, &params);
xefgSwapChainD3D12GetSwapChainPtr(
        xefgSwapChain, IID_PPV_ARGS(&swapChain));
```

请注意，XeSS-FG 默认是关闭的，可以通过调用 xefgSwapChainSetEnabled 启用。

XeSS-FG 分配的 GPU 资源分为两类：

- 持久性分配，例如网络权重和其他常量数据。
- 临时分配，例如网络激活。

应用程序可以选择提供外部资源堆。通过调用 xefgSwapChainGetProperties 可以获得这些资源堆所需的大小。分配的堆必须使用默认堆类型（D3D12_HEAP_TYPE_DEFAULT）。并且为了

确保最佳性能，此堆应该具有高内存驻留优先级。在相应的初始化参数中使用资源堆指针和偏移量。

应用程序可以选择提供一个外部描述符堆，XeSS-FG 在其中创建所有内部资源描述符。您需要通过调用 xefgSwapChainGetProperties 获取所需的描述符数量。您必须在初始化时指定 XEFG_SWAPCHAIN_INIT_FLAG_EXTERNAL_DESCRIPTOR_HEAP 标志，并将描述符堆和偏移量传递给 xefgSwapChainD3D12SetDescriptorHeap 。描述符堆必须是 D3D12_SRV_UAV_CBV_DESCRIPTOR_HEAP 类型并且着色器可见。

应用程序可以提供初始化标志和/或 UI 处理模式的配置来修改库行为。

```
xefg_swapchain_d3d12_init_params_t params = {};
params.initFlags = XEFG_SWAPCHAIN_INIT_FLAG_USE_NDC_VELOCITY;
params.uiMode = XEFG_SWAPCHAIN_UI_MODE_BACKBUFFER_HUDLESS;
```

更多初始化参数及其要求见下文。

初始化标志

应用程序可以使用不同的初始化标志来配置 XeSS-FG：

- XEFG_SWAPCHAIN_INIT_FLAG_NONE – 默认
- XEFG_SWAPCHAIN_INIT_FLAG_INVERTED_DEPTH – 深度纹理使用反转值
- XEFG_SWAPCHAIN_INIT_FLAG_EXTERNAL_DESCRIPTOR_HEAP – 使用应用程序提供的外部描述符堆
- XEFG_SWAPCHAIN_INIT_FLAG_HIGH_RES_MV – 高分辨率运动矢量纹理（交换链分辨率）
- XEFG_SWAPCHAIN_INIT_FLAG_USE_NDC_VELOCITY – 运动矢量使用归一化设备坐标（NDC）
- XEFG_SWAPCHAIN_INIT_FLAG_JITTERED_MV – 运动矢量没有取消抖动
- XEFG_SWAPCHAIN_INIT_FLAG_UITEXTURE_NOT_PREMUL_ALPHA – 纹理中的 RGB 值未被相应的 alpha 通道值预乘

UI 合成模式

XeSS-FG 提供了几种处理插值帧的用户界面（UI）元素的选项。应用程序可以提供额外的资源：

- **HUD-less Color** - 一个应用了所有后处理效果的完整场景颜色缓冲区，与后台缓冲区的纹理格式和颜色空间相同，但没有任何 UI 或 HUD 元素。
- **UI-Only Texture** - 仅包含 UI/HUD 元素的纹理，具有适当的 alpha 值表示透明度，与后台缓冲区的纹理格式和颜色空间相同。UI-Only 纹理的内容必须满足以下公式：

```
Final_Color.RGB = UI.RGB + (1 - UI.Alpha) * Hudless.RGB
```

重要提示! 为了获得更好的 UI 合成质量并最小化 UI 失真，应用程序必须提供这两种额外资源：HUD -less 颜色和 UI-Only 纹理。

默认情况下，XeSS-FG 使用以下 UI 合成模式：

- XEFG_SWAPCHAIN_UI_MODE_AUTO - 根据额外资源的可用性（HUD-less 颜色和 UI-only 纹理）自动确定最佳 UI 处理模式。

应用程序可以手动选择 UI 合成模式。在提供了 HUD-less 颜色和 UI-Only 纹理资源的情况下，使用以下 UI 合成模式之一：

- XEFG_SWAPCHAIN_UI_MODE_HUDLESS_UITEXTURE - 在 HUD-less 颜色缓冲区上进行插值，并使用上面的公式融合 UI 纹理。
  如果游戏可以提供公式中所需要的完全正确的 UI 纹理，使用此模式。
- XEFG_SWAPCHAIN_UI_MODE_BACKBUFFER_HUDLESS_UITEXTURE - 在 HUD-less 颜色缓冲区上插值，并使用从后台缓冲区提取的 UI 纹理进一步优化 UI。
  如果游戏可以提供了 UI 纹理但不能完全满足混合公式，请使用此模式。例如，这是基于 Unreal Engine* 游戏的首选选项。

如果应用程序无法提供其中一种或两种额外资源：

- XEFG_SWAPCHAIN_UI_MODE_BACKBUFFER_HUDLESS - 在 HUD-less 颜色缓冲区上插值，并从后台缓冲区提取 UI。这种 UI 合成的画面质量可能不如提供 UI-Only 纹理的情况，特别是对于半透明的 UI 元素。
  如果游戏不能提供 UI-Only 纹理但可以提供 HUD-less 颜色纹理，请使用此模式。
- XEFG_SWAPCHAIN_UI_MODE_BACKBUFFER_UITEXTURE - 在后台缓冲区上插值，并使用带 alpha 的 UI 纹理进一步优化 UI。
  如果游戏不能提供 HUD-less 颜色纹理但可以提供 UI-Only 纹理，请使用此模式。
- XEFG_SWAPCHAIN_UI_MODE_NONE - 在后台缓冲区上插值，没有任何特殊的 UI 处理。此模式不推荐，因为它可能会在 UI 元素的插值中产生不可接受的视觉质量瑕疵。

UI 合成模式比较表

以下表格显示了不同 UI 处理模式需要提供的资源：

ID	**模式名称**	HUD-less Color	UI-Only Texture
0	AUTO	Optional	Optional
1	NONE	Not Used	Not Used
2	BACKBUFFER_UITEXTURE	Not Used	Required
3	HUDLESS_UITEXTURE	Required	Required
4	BACKBUFFER_HUDLESS	Required	Not Used
5	BACKBUFFER_HUDLESS_UITEXTURE	Required	Required

性能影响

每种 UI 合成模式都有不同的性能特性。例如，需要额外输入（如 UI-Only 纹理）的模式可能由于额外的混合处理而产生轻微的性能开销。以下是每种模式的性能影响概述：

- XEFG_SWAPCHAIN_UI_MODE_NONE - 此模式没有性能影响，因为它不对 UI 元素进行任何额外处理。

- XEFG_SWAPCHAIN_UI_MODE_BACKBUFFER_HUDLESS_UITEXTURE - 虽然提供了最健壮的 UI 元素合成，但与其他模式相比，此模式可能会略微增加帧生成的 GPU 开销。
- **所有其他模式** - 这些模式会因为额外的 UI 处理（如纹理混合和从后台缓冲区提取 UI）轻微的增加性能开销。

错误处理

在将 XeSS-FG 集成到您的应用程序中时，在 UI 处理模式选择和使用过程中可能出现的潜在错误需要进行处理。以下是常见的错误场景：

- **无效模式(Invalid Mode)**- 如果提供的值不是 xefg_swapchain_ui_mode_t 的常量。SDK 将在初始化时返回 XEFG_SWAPCHAIN_RESULT_ERROR_INVALID_ARGUMENT 错误代码，表示选择了无效的模式。
- **缺少资源 Missing Resources** - 如果在所选模式所需时缺少 HUD-less 颜色缓冲区或 UI-Only 纹理，XeSS-FG 将使用 Present Status 报告 XEFG_SWAPCHAIN_RESULT_ERROR_MISMATCH_INPUT_RESOURCES（有关更多信息，请参见下文）。

HDR 显示支持

XeSS-FG 提供对使用 R10G10B10A2_UNORM 像素格式的高动态范围 (HDR) 显示的支持。使用 HDR 时，后台缓冲区、HUD-less 纹理和 UI-Only 纹理必须具有相同的像素格式并处于 HDR10/BT.2100 颜色空间中。由于 2 位 alpha 可能不足以表示半透明 UI 元素，为避免闪烁，可能需要优先选择 XEFG_SWAPCHAIN_UI_MODE_BACKBUFFER_HUDLESS_UITEXTURE。请注意，XeSS-FG 不支持 FP16 HDR 格式和 scRGB 颜色空间。请参阅 https://learn.microsoft.com/en-us/windows/win32/direct3darticles/high-dynamic-range#option-2-use-uint10rgb10-pixel-format-and-hdr10bt2100-color-space 获取详细信息。

启用帧生成

在创建上下文和代理交换链后，应用程序可以通过调用 xefgSwapChainSetEnabled 启用帧生成。强烈建议应用程序尽早启用 XeSS-FG。

```
xefgSwapChainSetEnabled(xefgSwapChain, true);
```

当应用程序处于全屏独占模式时，无法启用帧生成。如果在 XeSS-FG 启用时应用程序进入全屏独占模式，帧生成将被禁用，交换链将以“直通模式”（pass through）工作，即仅 Present 游戏渲染帧而不使用 XeSS-FG。

在离开全屏独占模式后，必须再次调用 xefgSwapChainSetEnabled 以启用帧生成。

应用程序可以随时打开或关闭 XeSS-FG，但在已经标记了具有 XEFG_SWAPCHAIN_RESOURCE_VALIDITY_ONLY_NOW 有效性的资源且命令队列尚未执行时除外。每次关闭调用将标记历史记录，以便在下次启用帧生成时进行清理。当启用帧生成时，应用程序必须确保启用了 XeLL 低延迟技术。

建议在游戏处于菜单或任何暂停状态或未提交渲染命令时禁用 XeSS-FG。

当 XeSS-FG 被禁用时，代理交换链仍然会有一些小的开销。如果应用程序希望实现零开销，则必须销毁 XeSS-FG 上下文并重新创建交换链。

资源标记

应用程序必须通过调用 xefgSwapChainD3D12TagFrameResource 并提供相应的帧计数值和填充的 xefg_swapchain_d3d12_resource_data_t 结构，来标记每帧所需的帧生成资源。

除了运动矢量和深度之外，所有资源在帧之间必须具有相同的尺寸。如果应用程序需要更改资源尺寸，则必须重新初始化 XeSS-FG。

每次调用标记资源时，必须指定一种以下资源有效性模式：

- XEFG_SWAPCHAIN_RV_UNTIL_NEXT_PRESENT – 此标志表示应用程序保证资源在下次 Present 调用之前是有效的。XeSS-FG 将使用此标志优化执行。
- XEFG_SWAPCHAIN_RV_ONLY_NOW – 此标志表示资源仅在当前有效，XeSS-FG 将复制操作记录到提供的命令列表中。

应用程序必须为每个标记的资源提供有效的内部状态。建议根据资源有效性将资源设置为以下状态：

- 为 XEFG_SWAPCHAIN_RV_UNTIL_NEXT_PRESENT 使用 D3D12_RESOURCE_STATE_NON_PIXEL_SHADER_RESOURCE
- 为 XEFG_SWAPCHAIN_RV_ONLY_NOW 使用 D3D12_RESOURCE_STATE_COPY_SOURCE

以下是输入颜色帧资源标记的示例：

```
xefg_swapchain_d3d12_resource_data_t hudlessColor = {};
hudlessColor.type = XEFG_SWAPCHAIN_RES_HUDLESS_COLOR;
hudlessColor.validity = XEFG_SWAPCHAIN_RV_UNTIL_NEXT_PRESENT;
hudlessColor.resourceSize = { width, height };
hudlessColor.pResource = renderTargets[m_frameIndex]
hudlessColor.incomingState =
  D3D12_RESOURCE_STATE_NON_PIXEL_SHADER_RESOURCE;

xefgSwapChainD3D12TagFrameResource(xefgSwapChain, commandList,
  frameCounter, &hudlessColor));
```

不同类型的资源将在下面更详细地描述。

输入和输出

XeSS-FG 将自动从代理交换链中获取所需的后台缓冲区和颜色资源。

此外，XeSS-FG 每帧需要以下最小输入集：

- 速度纹理(运动矢量)
- 深度纹理
- 一组帧常量

根据所选的 UI 处理模式，应用程序可能需要提供额外的资源：

- HUD-less 颜色纹理
- UI 纹理

应用程序通过标记将这些输入提供给 XeSS-FG：资源使用 xefgSwapChainD3D12TagFrameResource 标记，帧常量则使用 xefgSwapChainTagFrameConstants 标记。

如果应用程序使用 XEFG_SWAPCHAIN_INIT_FLAG_EXTERNAL_DESCRIPTOR_HEAP 初始化了 XeSS-FG，则在调用 Present 之前必须提供相应的描述符堆。

运动矢量

运动矢量指定从当前帧到前一帧的屏幕空间运动（以像素为单位）。如果应用程序使用归一化设备坐标（NDC） 进行运动矢量计算，则应在 XeSS-FG 上下文初始化期间传递一个附加标志（ XEFG_SWAPCHAIN_INIT_FLAG_USE_NDC_VELOCITY ） 。 XeSS-FG 接受 R16G16_FLOAT 或类似格式的运动矢量，其中 R 通道编码 X 方向的运动，G 通道编码 Y 方向的运动。运动矢量不包括由相机抖动引起的运动。运动矢量可以是低分辨率（默认且推荐）或高分辨率。低分辨率运动矢量由在超采样通道或 TAAU 输入分辨率下的 2D 纹理表示，高分辨率运动矢量由目标分辨率下的 2D 纹理表示。

重要提示! 使用高分辨率运动矢量可能会大大降低帧生成的性能，特别是输出分辨率很高的情况下，因此通常推荐使用低分辨率运动矢量。

对于高分辨率运动矢量，由相机动画产生的速度分量在延迟渲染通道中使用相机变换和深度值在目标分辨率下计算。然而，与粒子和物体动画相关的速度分量通常在输入分辨率下计算并存储在 G-Buffer 中。这个速度分量被上采样并与相机速度组合以生成高分辨率运动矢量的纹理。XeSS-FG 也需要扩展后的高分辨率运动矢量。例如，运动矢量表示输入像素的小邻域（如 3×3）的最前面表面的运动。用户可以在单独的渲染通道中实现高分辨率运动矢量的计算。

低分辨率运动矢量未被扩展，直接表示在每个抖动像素位置采样的速度。XeSS-FG 在内部将运动矢量上采样到目标网格，并使用深度纹理扩展它们。图 2 显示了同一运动矢量在低分辨率和高分辨率下的表示。

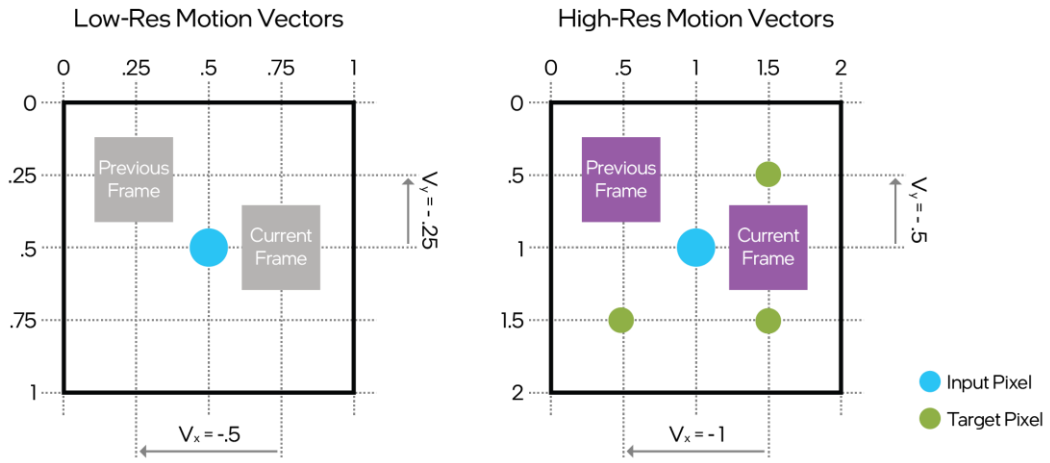


图 2. XeSS-FG 指定低分辨率和高分辨率运动矢量的约定

一些游戏引擎仅将对象渲染到 G-Buffer 缓冲区，并在 TAA 着色器中快速计算相机速度。在这种情况下，在执行 XeSS-FG 之前需要额外的一次处理，以合并对象和摄像机速度并生成一个展平的速度缓冲区。在这种情况下，高分辨率运动向量可能是更好的选择，因为展平处理可以在目标分辨率下执行。深度

支持任何深度格式，如 D32_FLOAT 或 D24_UNORM。默认情况下，XeSS-FG 假设较小的深度值更接近相机。然而，有些游戏引擎使用反转深度，这可以通过设置 XEFG_SWAPCHAIN_INIT_FLAG_INVERTED_DEPTH 来启用。

缓冲区尺寸须与运动矢量缓冲区相同。

HUD-less 颜色纹理

HUD-less 颜色纹理是应用了所有后处理效果的完整场景颜色纹理，与后台缓冲区的颜色空间和大小相同，但没有任何 UI 或 HUD 元素。

需要使用与主后台缓冲区相同的纹理格式。

UI 纹理

UI 纹理是仅包含 UI 元素（包括 HUD）的纹理，具有适当的 alpha 值表示透明度。UI-Only 纹理的内容必须满足以下公式：

```
Final_Color.RGB = UI.RGB + (1 - UI.Alpha) * Hudless.RGB
```

此纹理应具有与后台缓冲区相同的纹理格式、颜色空间和尺寸。

帧常量

应用程序必须在每帧提供一个 xefg_swapchain_frame_constant_data_t 结构，以通过调用 xefgSwapChainTagFrameConstants 正确更新配置并设置场景数据。该结构包括以下数据：

- viewMatrix - 视图矩阵，行优先，没有应用抖动
- projectionMatrix - 相机投影矩阵，行优先，没有应用抖动
- jitterOffsetX/Y - 抖动偏移，范围 [-0.5, 0.5]
- motionVectorScaleX/Y - 运动矢量缩放因子

- resetHistory – 如果设置此标志，XeSS-FG 将忽略之前的帧数据，并且不会对当前帧执行插值。当输出发生大变化时应设置此标志
- frameRenderTime – GPU 上的帧渲染时间，以毫秒为单位

示例：

```
xefg_swapchain_frame_constant_data_t constData = {};
XMFLOAT4X4 float4x4;
float fx = m_constantBufferData.offset.x;
float fy = m_constantBufferData.offset.y;
XMStoreFloat4x4(&float4x4, DirectX::XMMatrixTranslation(fx, fy, 0));
memcpy(constData.viewMatrix, float4x4.m, sizeof(float) * 16);
XMStoreFloat4x4(&float4x4, DirectX::XMMatrixIdentity());
memcpy(constData.projectionMatrix, float4x4.m, sizeof(float) * 16);

constData.jitterOffsetX = constData.jitterOffsetY = 0.0f;
constData.motionVectorScaleX = constData.motionVectorScaleY = 1.0f;

xefgSwapChainTagFrameConstants(xefgSwapChain, frameCounter, &constData );
```

描述符堆（Descriptor Heap）

如果 XeSS-FG 使用 XEFG_SWAPCHAIN_INIT_FLAG_EXTERNAL_DESCRIPTOR_HEAP 初始化，则在调用 Present 之前必须通过 xefgSwapChainD3D12SetDescriptorHeap 提供一个 ID3D12DescriptorHeap。

提供的描述符堆必须至少有 xefg_swapchain_properties_t::requiredDescriptorCount 项，以容纳插值执行期间的所有描述符。

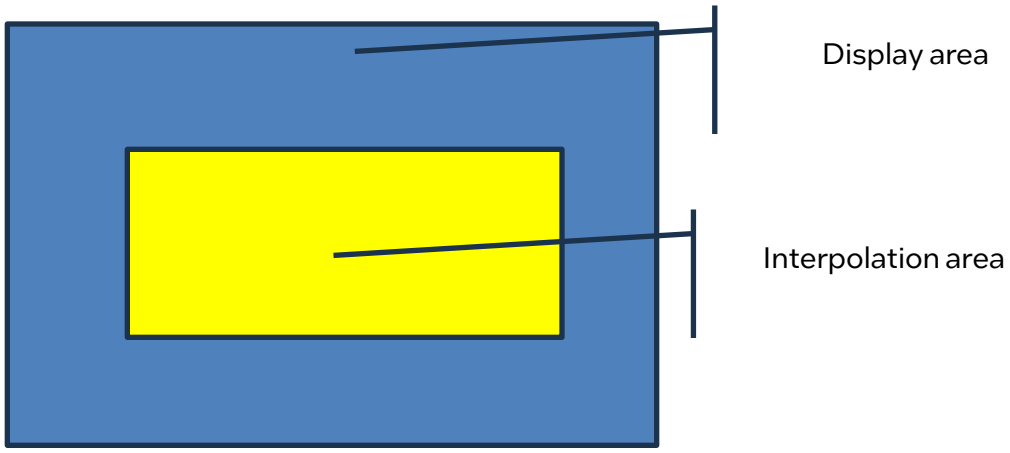
必须注意为每帧提供不同的偏移或不同的堆，因为帧生成执行可能仍在进行中，而应用程序正在编程下一帧。一旦描述符堆和偏移不再使用，它们可以再次用于新帧插值。

信箱效果（Letterboxing）

显示区域（Display Area）： 后台缓冲区的大小。由交换链初始化大小定义。

插值区域（Interpolation Area）： 在颜色资源（HUD、后台缓冲区、UI）内进行插值的感兴趣区域。由资源大小和偏移量定义。

为了实现信箱效果，XeSS-FG 支持部分帧插值，即在定义的插值区域内进行插值。下图显示了后台缓冲区资源内部插值区域和显示区域的关系：



为了启用此类插值模式，应用程序必须遵循以下步骤：

1. 使用所需显示大小初始化或重新初始化 XeSS-FG。
2. 对于每一帧：
    a. 调用 xefgSwapChainD3D12TagFrameResource 函数，以所需的插值大小（xefg_swapchain_d3d12_resource_data_t 结构体的 resourceBase 和 resourceSize 成员）标记 HUD-less 和/或 UI 资源。
    b. 调用 xefgSwapChainD3D12TagFrameResource 函数，以所需的插值大小（xefg_swapchain_d3d12_resource_data_t 结构体的 resourceBase 和 resourceSize 成员）标记后台缓冲区资源。注意：XeSS-FG 将忽略 xefg_swapchain_d3d12_resource_data_t 结构的其他成员。

插值仅在满足以下条件时有效：

1. 在 HUD-less、UI 和后台缓冲区资源中，插值区域（宽度和高度）在给定帧中相同。
2. 插值区域（宽度和高度）在前一帧和下一帧中相同。

否则，插值将被跳过，将呈现先前渲染的帧。

在插值执行期间，后台缓冲区中的显示区域将完全由 XeSS-FG 填充：

1. 插值区域内的像素进行插值。
2. 插值区域外的像素从先前生成的后台缓冲区资源中复制。

异步着色器管线构建

为了提高整体启动时间，XeSS-FG 支持异步方式编译着色器：

1. 在游戏初始化期间尽早创建 XeSS-FG 上下文。
2. 之后立即以非阻塞模式调用 xefgSwapChainD3D12BuildPipelines。XeSS-FG 将立即开始在后台编译内核，同时游戏继续加载。
3. 在稍后初始化 XeSS-FG 时（例如，当用户在菜单中选择它时）。内核着色器编译应该已经完成。
4. 可以通过调用 xefgSwapChainGetPipelineBuildStatus 查询着色器编译状态。

执行

在代理交换链的 Present 调用后，XeSS-FG 会自动执行。实际 Present 通过使用单独的线程异步进行，以正确调整帧节奏。

所有请求的插值帧将按顺序生成、呈现并正确调整节奏。XeSS-FG 高效地管理资源，必要时执行复制操作。

应用程序必须在每次交换链的 Present 之前调用 xefgSwapChainSetPresentId。此调用是为了正确识别为此帧标记的资源，未执行此调用将导致画面损坏或执行失败。在标记所有资源后调用 xefgSwapChainSetPresentId 很重要。

```
xefgSwapChainSetPresentId(xefgSwapChain, frameCounter);
++frameCounter;
swapChain->Present(…);
```

Present 状态

为了改进应用程序状态跟踪，XeSS-FG 提供了 xefgSwapChainGetLastPresentStatus 接口，该接口返回 xefg_swapchain_present_status_t 结构体，此结构体包含最后一次插值的结果、Present 帧队列的大小以及 XeSS-FG 是否启用的标志。

```
xefg_swapchain_present_status_t lastPresentStatus = {};
xefgSwapChainGetLastPresentStatus(xefgSwapChain, &lastPresentStatus);
```

重要提示! 它仅在 Present 之后和下次调用 xefgSwapChainSetPresentId 之前有效，因为数据与 present ID 相关联。

建议尽快在 Present 调用之后调用它。

更改窗口状态

当应用程序调整窗口大小或更改全屏状态时，必须调用 ResizeBuffers。应用程序必须确保释放所有未完成的资源以便重新创建它们。这对于标记的资源也很重要，因为此时历史记录将重置，任何与标记资源相关的即时工作必须完成。

XeSS-FG 更新与交换链关联的交换链队列，并忽略作为参数提供给 ResizeBuffers1 的队列，而用内部队列代替它们。

关闭

在调用 xefgSwapChainDestroy 之前，应用程序必须确保：

- 所有与 XeSS-FG 相关的操作已完成，不会再标记资源/常量或启用 XeSS-FG。
- 已释放 XeSS-FG 交换链的所有引用。

调用 xefgSwapChainDestroy 将清除/释放所有与帧生成相关的对象，包括 XeSS-FG 交换链 API 上下文句柄。如果 XeSS-FG 交换链有任何额外引用，函数将返回错误。这意味着交换链未被销毁，应用程序必须首先释放所有交换链引用。

XeSS-FG 使用 COM 指针来保存从应用程序收到的一些资源的引用。XeSS-FG 会释放这些资源。

在调用 xefgSwapChainDestroy 之后，如果应用程序打算继续渲染，即使没有 XeSS-FG 或其他帧生成技术，也必须重新创建交换链。

与第三方帧生成软件的兼容性

XeSS-FG 不兼容任何第三方帧生成软件。

如果应用程序希望从 XeSS-FG 切换到任何其他帧生成技术，则必须完全关闭 XeSS-FG。如果应用程序希望切换到 XeSS-FG，则应关闭任何其他运行的帧生成技术并从交换链创建开始执行完整初始化。XeSS-FG 不支持与任何其他库共享交换链。

调试

调试 API 的最常见方法是附加日志回调并检查日志消息，但如果这还不够并且需要更复杂的选项，则有两个附加工具 – XeSS Inspector 工具和调试选项 API：

- XEFG_SWAPCHAIN_DEBUG_FEATURE_SHOW_ONLY_INTERPOLATION – 只会在屏幕上看到生成的帧。如果帧生成失败 – 将呈现当前后台缓冲区，此行为可以通过附加标志 XEFG_SWAPCHAIN_DEBUG_FEATURE_PRESENT_FAILED_INTERPOLATION 更改。
- XEFG_SWAPCHAIN_DEBUG_FEATURE_TAG_INTERPOLATED_FRAMES – 在插值帧上显示紫色方框和垂直线，以区分它与渲染帧，并直观检查撕裂问题。
- XEFG_SWAPCHAIN_DEBUG_FEATURE_PRESENT_FAILED_INTERPOLATION – 如果插值失败，启用黑帧 Present，否则将呈现当前后台缓冲区。

可以使用 xefgSwapChainEnableDebugFeature 启用调试选项。此函数可以在上下文创建和销毁之间的任何时间调用。

示例：

```
xefgSwapChainEnableDebugFeature(xefgSwapChain,
XEFG_SWAPCHAIN_DEBUG_FEATURE_SHOW_ONLY_INTERPOLATION, true, nullptr);
```

版本控制

XeSS-SR 库使用了 <major>.<minor>.<patch> 版本格式和 Numeric 90+ 方案用于开发阶段版本。 版本号由 64 位大小的结构（xefg_swapchain_version_t）指定，其中：

- major 增量表示新 API，并可能导致功能中断。
- minor 增量表示 API 改进，例如可选输入或标志。
- patch 增量可能包括性能或质量调整或已知问题的修复。

版本嵌入到 XeSS-FG SDK 发布中，可以使用函数 xefgSwapChainGetVersion 访问。

声明

您不得使用或协助使用本文档连接到任何侵犯或其他法律分析有关的 Intel 产品。您同意授予 Intel 非排他性、免版税的许可，以使用本文所披露的主题撰写的任何专利声明。

性能因使用、配置和其他因素而异。了解更多信息，请访问 www.Intel.com/PerformanceIndex。

没有产品或组件是绝对安全的。

所有产品计划和路线图如有更改，恕不另行通知。

您的成本和结果可能会有所不同。

Intel 技术可能需要启用硬件、软件或服务激活。

Intel 技术的功能和好处取决于系统配置，可能需要启用硬件、软件或服务激活。性能因系统配置而异。请与您的系统制造商或零售商联系，了解更多信息，请访问 intel.com。

本文档未授予任何知识产权权利（明示或暗示，通过禁止反言或其他方式）。

Intel 不承担任何明示或暗示的保证，包括但不限于适销性、适合特定用途和不侵权的暗示保证，以及任何因履行、交易过程或贸易使用而产生的保证。

本文档包含有关开发中产品、服务和/或流程的信息。本文提供的所有信息如有更改，恕不另行通知。请联系您的 Intel 代表，获取最新的预测、时间表、规格和路线图。

描述的产品和服务可能包含被称为“已知问题”的缺陷或错误，可能导致与发布的规格有所偏离。当前已知问题的详细信息可供请求。

可通过致电 1-800-548-4725 或访问 www.intel.com/design/literature.htm 获取本文档中引用的具有订单编号的文件副本。

Microsoft、Windows 和 Windows 徽标是 Microsoft Corporation 在美国及/或其他国家的商标或注册商标。