Programmable Blend with Pixel Shader Ordering

Introduction

Pixel Shader Ordering is a graphics extension that Intel has implemented for 4th generation Intel® Core™ processors with Iris™ and Iris™ Pro graphics. Pixel Shader Ordering guarantees ordered access to unordered access view resources from a pixel shader.

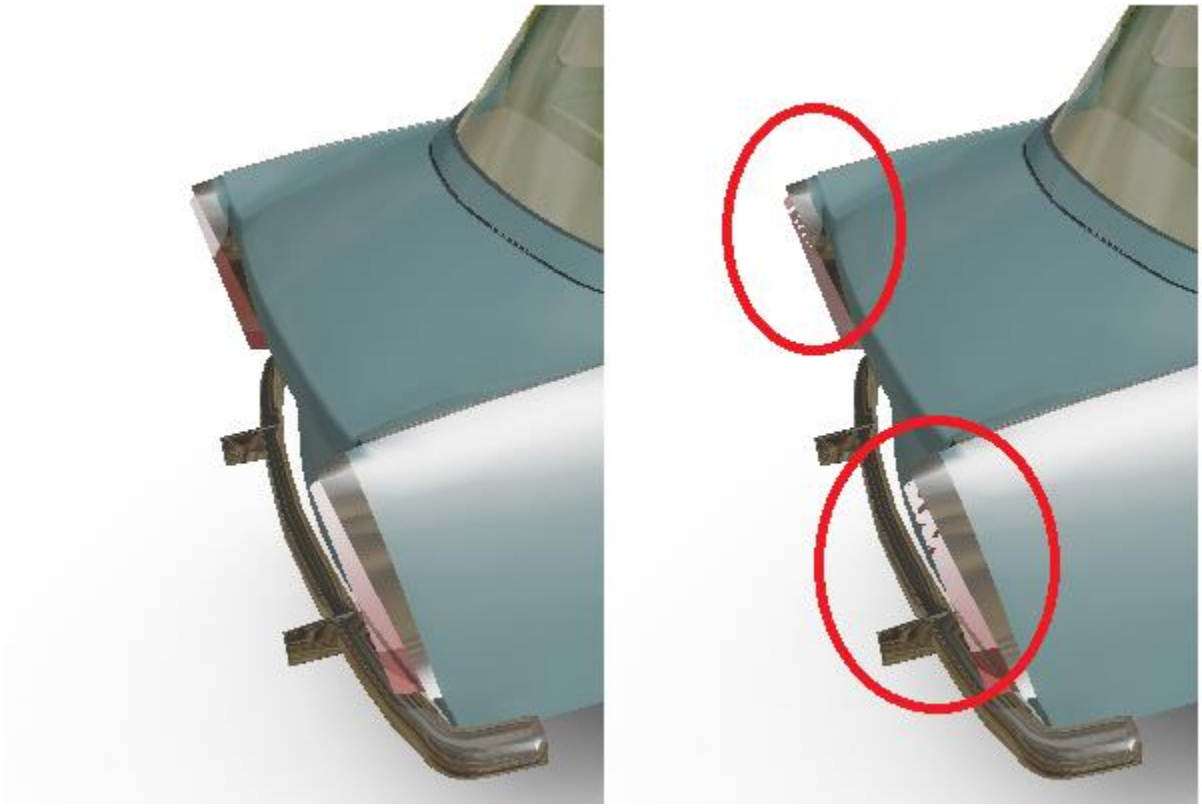
This sample demonstrates how to use Pixel Shader Ordering to perform blending in a pixel shader without using fixed function blending. For DirectX* 11.3 and DirectX 12, Rasterizer Order Views provide the similar functionality. This sample has been updated to include a Direct 11.3 path that utilizes Rasterizer Ordered Views.

Pixel Shader Ordering

Pixel Shader Ordering provides a mechanism for controlling access to memory in the pixel shader. When invoked, all reads and writes to the specified resources from a given pixel location are performed in submit order. Note that this order is not guaranteed across pixels. For example, if pixel shader invocations operating in two different pixel locations attempt to modify the same memory location, no ordering can be assumed.

To utilize the extension, some host-side initialization code is required. Sample code is provided in the IGFXExtensionsHelper.h/cpp files. Before using any extension, the Init function must be called.

Additionally, prior to utilizing the extension in a pixel shader, the shader must execute the IntelExt_Init() provided in IntelExtensions.hlsl. IntelExt_BeginPixelShaderOrderingOnUAV(RENDER_TARGET_UAV_SLOT) or IntelExt_BeginPixelShaderOrdering( ) to serialize access to all bound resources. The extension mechanism uses the final Render Target View / Unordered Access View slot, so that slot will not be available for use in the shader.



Rasterizer Order Views

To implement the same functionality using DirectX 11.3, the buffers in the shader are declared as “RasterizerOrdered*” instead of “RW*” for example, “RasterizerOrderedBuffer” instead of “RWBuffer.” For DirectX11.3, all writes to the Rasterizer Order Views are in triangle submit order; there is no function equivalent to “Intel_Ext_BeginPixelSHaderOrdering”. For additional information, on Rasterizer Order Views and Pixel Shader Ordering, refer to https://software.intel.com/en-us/gamedev/articles/rasterizer-order-views-101-a-primer.

Sample Implementation

The sample uses a shared exponent floating point format for the render target where 8 bits of precision are used to store the mantissa for the red, green, and blue values, and they all share a

single exponent value. This format is not supported by DirectX* and is fully defined in the pixel shaders and could be modified to change the distribution of values. Because the format is not supported, fixed function blending cannot be configured to correctly blend values in the render target.

The sample renders all of the opaque geometry, then binds the render target as an unordered access view and uses the Pixel Shader Ordering extension to perform blending of transparent geometry. If the extension is not available, the shader will still perform the blending in the pixel shader, but artifacts will appear where transparent geometry overlaps.

The sample also provides a path that uses the DXGI_R11G11B10_FLOAT format and fixed function blending for comparison.

Notices