NAME

Direct3D 10 Interoperability −

Modules

Direct3D 10 Interoperability [DEPRECATED]

Enumerations

enum **CUd3d10DeviceList** { **CU_D3D10_DEVICE_LIST_ALL** = 0x01,
CU_D3D10_DEVICE_LIST_CURRENT_FRAME = 0x02,
CU_D3D10_DEVICE_LIST_NEXT_FRAME = 0x03 }

Functions

CUresult cuD3D10GetDevice (**CUdevice** *pCudaDevice, IDXGIAdapter *pAdapter)
Gets the CUDA device corresponding to a display adapter.

CUresult cuD3D10GetDevices (unsigned int *pCudaDeviceCount, **CUdevice** *pCudaDevices, unsigned int cudaDeviceCount, ID3D10Device *pD3D10Device, **CUd3d10DeviceList** deviceList)
Gets the CUDA devices corresponding to a Direct3D 10 device.

CUresult cuGraphicsD3D10RegisterResource (**CUgraphicsResource** *pCudaResource, ID3D10Resource *pD3DResource, unsigned int Flags)
Register a Direct3D 10 resource for access by CUDA.

Detailed Description

\brief Direct3D 10 interoperability functions of the low-level CUDA driver API (cudaD3D10.h)

This section describes the Direct3D 10 interoperability functions of the low-level CUDA driver application programming interface. Note that mapping of Direct3D 10 resources is performed with the graphics API agnostic, resource mapping interface described in **Graphics Interoperability**.

Enumeration Type Documentation

enum CUd3d10DeviceList

CUDA devices corresponding to a D3D10 device

Enumerator:

CU_D3D10_DEVICE_LIST_ALL
The CUDA devices for all GPUs used by a D3D10 device

CU_D3D10_DEVICE_LIST_CURRENT_FRAME
The CUDA devices for the GPUs used by a D3D10 device in its currently rendering frame

CU_D3D10_DEVICE_LIST_NEXT_FRAME
The CUDA devices for the GPUs to be used by a D3D10 device in the next frame

Function Documentation

CUresult cuD3D10GetDevice (CUdevice * pCudaDevice, IDXGIAdapter * pAdapter)

Returns in `*pCudaDevice` the CUDA-compatible device corresponding to the adapter `pAdapter` obtained from IDXGIFactory::EnumAdapters.

If no device on `pAdapter` is CUDA-compatible then the call will fail.

Parameters:
pCudaDevice - Returned CUDA device corresponding to `pAdapter`
pAdapter - Adapter to query for CUDA device

Returns:
CUDA_SUCCESS, **CUDA_ERROR_DEINITIALIZED**, **CUDA_ERROR_NOT_INITIALIZED**, **CUDA_ERROR_INVALID_VALUE**, **CUDA_ERROR_NOT_FOUND**, **CUDA_ERROR_UNKNOWN**

Note:
Note that this function may also return error codes from previous, asynchronous launches.

See also:
cuD3D10GetDevices

CUresult cuD3D10GetDevices (unsigned int * pCudaDeviceCount, CUdevice * pCudaDevices, unsigned int cudaDeviceCount, ID3D10Device * pD3D10Device, CUd3d10DeviceList deviceList)

Returns in `*pCudaDeviceCount` the number of CUDA-compatible device corresponding to the

Direct3D 10 device `pD3D10Device`. Also returns in `*pCudaDevices` at most `cudaDeviceCount` of the CUDA-compatible devices corresponding to the Direct3D 10 device `pD3D10Device`.

If any of the GPUs being used to render `pDevice` are not CUDA capable then the call will return **CUDA_ERROR_NO_DEVICE**.

Parameters:

pCudaDeviceCount - Returned number of CUDA devices corresponding to `pD3D10Device`
pCudaDevices - Returned CUDA devices corresponding to `pD3D10Device`
cudaDeviceCount - The size of the output device array `pCudaDevices`
pD3D10Device - Direct3D 10 device to query for CUDA devices
deviceList - The set of devices to return. This set may be **CU_D3D10_DEVICE_LIST_ALL** for all devices, **CU_D3D10_DEVICE_LIST_CURRENT_FRAME** for the devices used to render the current frame (in SLI), or **CU_D3D10_DEVICE_LIST_NEXT_FRAME** for the devices used to render the next frame (in SLI).

Returns:

CUDA_SUCCESS, **CUDA_ERROR_DEINITIALIZED**, **CUDA_ERROR_NOT_INITIALIZED**, **CUDA_ERROR_NO_DEVICE**, **CUDA_ERROR_INVALID_VALUE**, **CUDA_ERROR_NOT_FOUND**, **CUDA_ERROR_UNKNOWN**

Note:

Note that this function may also return error codes from previous, asynchronous launches.

See also:

cuD3D10GetDevice

CUresult cuGraphicsD3D10RegisterResource (CUgraphicsResource * pCudaResource, ID3D10Resource * pD3DResource, unsigned int Flags)

Registers the Direct3D 10 resource `pD3DResource` for access by CUDA and returns a CUDA handle to `pD3Dresource` in `pCudaResource`. The handle returned in `pCudaResource` may be used to map and unmap this resource until it is unregistered. On success this call will increase the internal reference count on `pD3DResource`. This reference count will be decremented when this resource is unregistered through **cuGraphicsUnregisterResource()**.

This call is potentially high-overhead and should not be called every frame in interactive applications.

The type of `pD3DResource` must be one of the following.

- ID3D10Buffer: may be accessed through a device pointer.
- ID3D10Texture1D: individual subresources of the texture may be accessed via arrays
- ID3D10Texture2D: individual subresources of the texture may be accessed via arrays
- ID3D10Texture3D: individual subresources of the texture may be accessed via arrays

The `Flags` argument may be used to specify additional parameters at register time. The valid values for this parameter are

- CU_GRAPHICS_REGISTER_FLAGS_NONE: Specifies no hints about how this resource will be used.
- CU_GRAPHICS_REGISTER_FLAGS_SURFACE_LDST: Specifies that CUDA will bind this resource to a surface reference.
- CU_GRAPHICS_REGISTER_FLAGS_TEXTURE_GATHER: Specifies that CUDA will perform texture gather operations on this resource.

Not all Direct3D resources of the above types may be used for interoperability with CUDA. The following are some limitations.

- The primary rendertarget may not be registered with CUDA.
- Resources allocated as shared may not be registered with CUDA.
- Textures which are not of a format which is 1, 2, or 4 channels of 8, 16, or 32-bit integer or floating-point data cannot be shared.

- Surfaces of depth or stencil formats cannot be shared.

A complete list of supported DXGI formats is as follows. For compactness the notation A_{B,C,D} represents A_B, A_C, and A_D.

- DXGI_FORMAT_A8_UNORM
- DXGI_FORMAT_B8G8R8A8_UNORM
- DXGI_FORMAT_B8G8R8X8_UNORM
- DXGI_FORMAT_R16_FLOAT
- DXGI_FORMAT_R16G16B16A16_{FLOAT,SINT,SNORM,UINT,UNORM}
- DXGI_FORMAT_R16G16_{FLOAT,SINT,SNORM,UINT,UNORM}
- DXGI_FORMAT_R16_{SINT,SNORM,UINT,UNORM}
- DXGI_FORMAT_R32_FLOAT
- DXGI_FORMAT_R32G32B32A32_{FLOAT,SINT,UINT}
- DXGI_FORMAT_R32G32_{FLOAT,SINT,UINT}
- DXGI_FORMAT_R32_{SINT,UINT}
- DXGI_FORMAT_R8G8B8A8_{SINT,SNORM,UINT,UNORM,UNORM_SRGB}
- DXGI_FORMAT_R8G8_{SINT,SNORM,UINT,UNORM}
- DXGI_FORMAT_R8_{SINT,SNORM,UINT,UNORM}

If `pD3DResource` is of incorrect type or is already registered then **CUDA_ERROR_INVALID_HANDLE** is returned. If `pD3DResource` cannot be registered then **CUDA_ERROR_UNKNOWN** is returned. If `Flags` is not one of the above specified value then **CUDA_ERROR_INVALID_VALUE** is returned.

Parameters:

pCudaResource - Returned graphics resource handle
pD3DResource - Direct3D resource to register
Flags - Parameters for resource registration

Returns:

CUDA_SUCCESS, **CUDA_ERROR_DEINITIALIZED**, **CUDA_ERROR_NOT_INITIALIZED**, **CUDA_ERROR_INVALID_CONTEXT**, **CUDA_ERROR_INVALID_VALUE**, **CUDA_ERROR_INVALID_HANDLE**, **CUDA_ERROR_OUT_OF_MEMORY**, **CUDA_ERROR_UNKNOWN**

Note:

Note that this function may also return error codes from previous, asynchronous launches.

See also:

cuGraphicsUnregisterResource, **cuGraphicsMapResources**, **cuGraphicsSubResourceGetMappedArray**, **cuGraphicsResourceGetMappedPointer**

Author

Generated automatically by Doxygen from the source code.