NAME

Direct3D 9 Interoperability −

Modules

Direct3D 9 Interoperability [DEPRECATED]

Enumerations

enum **CUd3d9DeviceList** { **CU_D3D9_DEVICE_LIST_ALL** = 0x01,
CU_D3D9_DEVICE_LIST_CURRENT_FRAME = 0x02,
CU_D3D9_DEVICE_LIST_NEXT_FRAME = 0x03 }

Functions

CUresult cuD3D9CtxCreate (**CUcontext** *pCtx, **CUdevice** *pCudaDevice, unsigned int Flags, IDirect3DDevice9 *pD3DDevice)
Create a CUDA context for interoperability with Direct3D 9.

CUresult cuD3D9CtxCreateOnDevice (**CUcontext** *pCtx, unsigned int flags, IDirect3DDevice9 *pD3DDevice, **CUdevice** cudaDevice)
Create a CUDA context for interoperability with Direct3D 9.

CUresult cuD3D9GetDevice (**CUdevice** *pCudaDevice, const char *pszAdapterName)
Gets the CUDA device corresponding to a display adapter.

CUresult cuD3D9GetDevices (unsigned int *pCudaDeviceCount, **CUdevice** *pCudaDevices, unsigned int cudaDeviceCount, IDirect3DDevice9 *pD3D9Device, **CUd3d9DeviceList** deviceList)
Gets the CUDA devices corresponding to a Direct3D 9 device.

CUresult cuD3D9GetDirect3DDevice (IDirect3DDevice9 **ppD3DDevice)
Get the Direct3D 9 device against which the current CUDA context was created.

CUresult cuGraphicsD3D9RegisterResource (**CUgraphicsResource** *pCudaResource, IDirect3DResource9 *pD3DResource, unsigned int Flags)
Register a Direct3D 9 resource for access by CUDA.

Detailed Description

\brief Direct3D 9 interoperability functions of the low-level CUDA driver API (**cudaD3D9.h**)

This section describes the Direct3D 9 interoperability functions of the low-level CUDA driver application programming interface. Note that mapping of Direct3D 9 resources is performed with the graphics API agnostic, resource mapping interface described in **Graphics Interoperability**.

Enumeration Type Documentation

enum CUd3d9DeviceList

CUDA devices corresponding to a D3D9 device

Enumerator:

CU_D3D9_DEVICE_LIST_ALL
The CUDA devices for all GPUs used by a D3D9 device

CU_D3D9_DEVICE_LIST_CURRENT_FRAME
The CUDA devices for the GPUs used by a D3D9 device in its currently rendering frame

CU_D3D9_DEVICE_LIST_NEXT_FRAME
The CUDA devices for the GPUs to be used by a D3D9 device in the next frame

Function Documentation

CUresult cuD3D9CtxCreate (CUcontext * pCtx, CUdevice * pCudaDevice, unsigned int Flags, IDirect3DDevice9 * pD3DDevice)

Creates a new CUDA context, enables interoperability for that context with the Direct3D device `pD3DDevice`, and associates the created CUDA context with the calling thread. The created **CUcontext** will be returned in `*pCtx`. Direct3D resources from this device may be registered and mapped through the lifetime of this CUDA context. If `pCudaDevice` is non-NULL then the **CUdevice** on which this CUDA context was created will be returned in `*pCudaDevice`.

On success, this call will increase the internal reference count on `pD3DDevice`. This reference count will be decremented upon destruction of this context through **cuCtxDestroy()**. This context will cease to function if `pD3DDevice` is destroyed or encounters an error.

Note that this function is never required for correct functionality. Use of this function will result in

accelerated interoperability only when the operating system is Windows Vista or Windows 7, and the device `pD3DDdevice` is not an IDirect3DDevice9Ex. In all other circumstances, this function is not necessary.

Parameters:

pCtx - Returned newly created CUDA context
pCudaDevice - Returned pointer to the device on which the context was created
Flags - Context creation flags (see **cuCtxCreate()** for details)
pD3DDevice - Direct3D device to create interoperability context with

Returns:

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

Note:

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

See also:

cuD3D9GetDevice, **cuGraphicsD3D9RegisterResource**

CUresult cuD3D9CtxCreateOnDevice (CUcontext * pCtx, unsigned int flags, IDirect3DDevice9 * pD3DDevice, CUdevice cudaDevice)

Creates a new CUDA context, enables interoperability for that context with the Direct3D device `pD3DDevice`, and associates the created CUDA context with the calling thread. The created **CUcontext** will be returned in `*pCtx`. Direct3D resources from this device may be registered and mapped through the lifetime of this CUDA context.

On success, this call will increase the internal reference count on `pD3DDevice`. This reference count will be decremented upon destruction of this context through **cuCtxDestroy()**. This context will cease to function if `pD3DDevice` is destroyed or encounters an error.

Note that this function is never required for correct functionality. Use of this function will result in accelerated interoperability only when the operating system is Windows Vista or Windows 7, and the device `pD3DDdevice` is not an IDirect3DDevice9Ex. In all other circumstances, this function is not necessary.

Parameters:

pCtx - Returned newly created CUDA context
flags - Context creation flags (see **cuCtxCreate()** for details)
pD3DDevice - Direct3D device to create interoperability context with
cudaDevice - The CUDA device on which to create the context. This device must be among the devices returned when querying CU_D3D9_DEVICES_ALL from **cuD3D9GetDevices**.

Returns:

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

Note:

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

See also:

cuD3D9GetDevices, **cuGraphicsD3D9RegisterResource**

CUresult cuD3D9GetDevice (CUdevice * pCudaDevice, const char * pszAdapterName)

Returns in `*pCudaDevice` the CUDA-compatible device corresponding to the adapter name `pszAdapterName` obtained from EnumDisplayDevices() or IDirect3D9::GetAdapterIdentifier().

If no device on the adapter with name `pszAdapterName` is CUDA-compatible, then the call will fail.

Parameters:

pCudaDevice - Returned CUDA device corresponding to pszAdapterName
pszAdapterName - Adapter name to query for 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:
cuD3D9CtxCreate

CUresult cuD3D9GetDevices (unsigned int * pCudaDeviceCount, CUdevice * pCudaDevices, unsigned int cudaDeviceCount, IDirect3DDevice9 * pD3D9Device, CUd3d9DeviceList deviceList)
Returns in `*pCudaDeviceCount` the number of CUDA-compatible device corresponding to the Direct3D 9 device `pD3D9Device`. Also returns in `*pCudaDevices` at most `cudaDeviceCount` of the CUDA-compatible devices corresponding to the Direct3D 9 device `pD3D9Device`.

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 `pD3D9Device`
pCudaDevices - Returned CUDA devices corresponding to `pD3D9Device`
cudaDeviceCount - The size of the output device array `pCudaDevices`
pD3D9Device - Direct3D 9 device to query for CUDA devices
deviceList - The set of devices to return. This set may be **CU_D3D9_DEVICE_LIST_ALL** for all devices, **CU_D3D9_DEVICE_LIST_CURRENT_FRAME** for the devices used to render the current frame (in SLI), or **CU_D3D9_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:
cuD3D9CtxCreate

CUresult cuD3D9GetDirect3DDevice (IDirect3DDevice9 ** ppD3DDevice)
Returns in `*ppD3DDevice` the Direct3D device against which this CUDA context was created in **cuD3D9CtxCreate()**.

Parameters:
ppD3DDevice - Returned Direct3D device corresponding to CUDA context

Returns:
CUDA_SUCCESS, **CUDA_ERROR_DEINITIALIZED**, **CUDA_ERROR_NOT_INITIALIZED**, **CUDA_ERROR_INVALID_CONTEXT**

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

See also:
cuD3D9GetDevice

CUresult cuGraphicsD3D9RegisterResource (CUgraphicsResource * pCudaResource, IDirect3DResource9 * pD3DResource, unsigned int Flags)
Registers the Direct3D 9 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.

- IDirect3DVertexBuffer9: may be accessed through a device pointer
- IDirect3DIndexBuffer9: may be accessed through a device pointer
- IDirect3DSurface9: may be accessed through an array. Only stand-alone objects of type IDirect3DSurface9 may be explicitly shared. In particular, individual mipmap levels and faces of cube maps may not be registered directly. To access individual surfaces associated with a texture, one must register the base texture object.
- IDirect3DBaseTexture9: individual surfaces on this texture may be accessed through an array.

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 formats is as follows:

- D3DFMT_L8
- D3DFMT_L16
- D3DFMT_A8R8G8B8
- D3DFMT_X8R8G8B8
- D3DFMT_G16R16
- D3DFMT_A8B8G8R8
- D3DFMT_A8
- D3DFMT_A8L8
- D3DFMT_Q8W8V8U8
- D3DFMT_V16U16
- D3DFMT_A16B16G16R16F
- D3DFMT_A16B16G16R16
- D3DFMT_R32F
- D3DFMT_G16R16F
- D3DFMT_A32B32G32R32F
- D3DFMT_G32R32F
- D3DFMT_R16F

If Direct3D interoperability is not initialized for this context using **cuD3D9CtxCreate** then **CUDA_ERROR_INVALID_CONTEXT** is returned. 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:

cuD3D9CtxCreate, **cuGraphicsUnregisterResource**, **cuGraphicsMapResources**, **cuGraphicsSubResourceGetMappedArray**, **cuGraphicsResourceGetMappedPointer**

Author

Generated automatically by Doxygen from the source code.