NAME

OpenGL Interoperability −

Modules

OpenGL Interoperability [DEPRECATED]

Enumerations

enum **CUGLDeviceList** { **CU_GL_DEVICE_LIST_ALL** = 0x01,
CU_GL_DEVICE_LIST_CURRENT_FRAME = 0x02,
CU_GL_DEVICE_LIST_NEXT_FRAME = 0x03 }

Functions

CUresult cuGLGetDevices (unsigned int *pCudaDeviceCount, **CUdevice** *pCudaDevices, unsigned int cudaDeviceCount, **CUGLDeviceList** deviceList)
Gets the CUDA devices associated with the current OpenGL context.

CUresult cuGraphicsGLRegisterBuffer (**CUgraphicsResource** *pCudaResource, GLuint buffer, unsigned int Flags)
Registers an OpenGL buffer object.

CUresult cuGraphicsGLRegisterImage (**CUgraphicsResource** *pCudaResource, GLuint image, GLenum target, unsigned int Flags)
Register an OpenGL texture or renderbuffer object.

CUresult cuWGLGetDevice (**CUdevice** *pDevice, HGPUNV hGpu)
Gets the CUDA device associated with hGpu.

Detailed Description

\brief OpenGL interoperability functions of the low-level CUDA driver API (**cudaGL.h**)

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

Enumeration Type Documentation

enum CUGLDeviceList

CUDA devices corresponding to an OpenGL device

Enumerator:

CU_GL_DEVICE_LIST_ALL
: The CUDA devices for all GPUs used by the current OpenGL context

CU_GL_DEVICE_LIST_CURRENT_FRAME
: The CUDA devices for the GPUs used by the current OpenGL context in its currently rendering frame

CU_GL_DEVICE_LIST_NEXT_FRAME
: The CUDA devices for the GPUs to be used by the current OpenGL context in the next frame

Function Documentation

CUresult cuGLGetDevices (unsigned int * pCudaDeviceCount, CUdevice * pCudaDevices, unsigned int cudaDeviceCount, CUGLDeviceList deviceList)

Returns in `*pCudaDeviceCount` the number of CUDA-compatible devices corresponding to the current OpenGL context. Also returns in `*pCudaDevices` at most cudaDeviceCount of the CUDA-compatible devices corresponding to the current OpenGL context. If any of the GPUs being used by the current OpenGL context are not CUDA capable then the call will return CUDA_ERROR_NO_DEVICE.

The `deviceList` argument may be any of the following:

- **CU_GL_DEVICE_LIST_ALL**: Query all devices used by the current OpenGL context.
- **CU_GL_DEVICE_LIST_CURRENT_FRAME**: Query the devices used by the current OpenGL context to render the current frame (in SLI).
- **CU_GL_DEVICE_LIST_NEXT_FRAME**: Query the devices used by the current OpenGL context to render the next frame (in SLI). Note that this is a prediction, it can’t be guaranteed that this is correct in all cases.

Parameters:

pCudaDeviceCount - Returned number of CUDA devices.
pCudaDevices - Returned CUDA devices.
cudaDeviceCount - The size of the output device array pCudaDevices.
deviceList - The set of devices to return.

Returns:
CUDA_SUCCESS, **CUDA_ERROR_NO_DEVICE**, **CUDA_ERROR_INVALID_VALUE**, **CUDA_ERROR_INVALID_CONTEXT**, **CUDA_ERROR_INVALID_GRAPHICS_CONTEXT**

Note:
This function is not supported on Mac OS X.

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

See also:
cuWGLGetDevice

CUresult cuGraphicsGLRegisterBuffer (CUgraphicsResource * pCudaResource, GLuint buffer, unsigned int Flags)
Registers the buffer object specified by `buffer` for access by CUDA. A handle to the registered object is returned as `pCudaResource`. The register flags `Flags` specify the intended usage, as follows:

- CU_GRAPHICS_REGISTER_FLAGS_NONE: Specifies no hints about how this resource will be used. It is therefore assumed that this resource will be read from and written to by CUDA. This is the default value.
- CU_GRAPHICS_REGISTER_FLAGS_READ_ONLY: Specifies that CUDA will not write to this resource.
- CU_GRAPHICS_REGISTER_FLAGS_WRITE_DISCARD: Specifies that CUDA will not read from this resource and will write over the entire contents of the resource, so none of the data previously stored in the resource will be preserved.

Parameters:
pCudaResource - Pointer to the returned object handle
buffer - name of buffer object to be registered
Flags - Register flags

Returns:
CUDA_SUCCESS, **CUDA_ERROR_INVALID_HANDLE**, **CUDA_ERROR_ALREADY_MAPPED**, **CUDA_ERROR_INVALID_CONTEXT**,

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

See also:
cuGraphicsUnregisterResource, **cuGraphicsMapResources**, **cuGraphicsResourceGetMappedPointer**

CUresult cuGraphicsGLRegisterImage (CUgraphicsResource * pCudaResource, GLuint image, GLenum target, unsigned int Flags)
Registers the texture or renderbuffer object specified by `image` for access by CUDA. A handle to the registered object is returned as `pCudaResource`.

`target` must match the type of the object, and must be one of GL_TEXTURE_2D, GL_TEXTURE_RECTANGLE, GL_TEXTURE_CUBE_MAP, GL_TEXTURE_3D, GL_TEXTURE_2D_ARRAY, or GL_RENDERBUFFER.

The register flags `Flags` specify the intended usage, as follows:

- CU_GRAPHICS_REGISTER_FLAGS_NONE: Specifies no hints about how this resource will be used. It is therefore assumed that this resource will be read from and written to by CUDA. This is the default value.
- CU_GRAPHICS_REGISTER_FLAGS_READ_ONLY: Specifies that CUDA will not write to this resource.

- CU_GRAPHICS_REGISTER_FLAGS_WRITE_DISCARD: Specifies that CUDA will not read from this resource and will write over the entire contents of the resource, so none of the data previously stored in the resource will be preserved.
- 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.

The following image formats are supported. For brevity's sake, the list is abbreviated. For ex., {GL_R, GL_RG} X {8, 16} would expand to the following 4 formats {GL_R8, GL_R16, GL_RG8, GL_RG16} :

- GL_RED, GL_RG, GL_RGBA, GL_LUMINANCE, GL_ALPHA, GL_LUMINANCE_ALPHA, GL_INTENSITY
- {GL_R, GL_RG, GL_RGBA} X {8, 16, 16F, 32F, 8UI, 16UI, 32UI, 8I, 16I, 32I}
- {GL_LUMINANCE, GL_ALPHA, GL_LUMINANCE_ALPHA, GL_INTENSITY} X {8, 16, 16F_ARB, 32F_ARB, 8UI_EXT, 16UI_EXT, 32UI_EXT, 8I_EXT, 16I_EXT, 32I_EXT}

The following image classes are currently disallowed:

- Textures with borders
- Multisampled renderbuffers

Parameters:

pCudaResource - Pointer to the returned object handle
image - name of texture or renderbuffer object to be registered
target - Identifies the type of object specified by `image`
Flags - Register flags

Returns:

CUDA_SUCCESS, **CUDA_ERROR_INVALID_HANDLE**, **CUDA_ERROR_ALREADY_MAPPED**, **CUDA_ERROR_INVALID_CONTEXT**,

Note:

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

See also:

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

CUresult cuWGLGetDevice (CUdevice * pDevice, HGPUNV hGpu)

Returns in `*pDevice` the CUDA device associated with a `hGpu`, if applicable.

Parameters:

pDevice - Device associated with hGpu
hGpu - Handle to a GPU, as queried via WGL_NV_gpu_affinity()

Returns:

CUDA_SUCCESS, **CUDA_ERROR_DEINITIALIZED**, **CUDA_ERROR_NOT_INITIALIZED**, **CUDA_ERROR_INVALID_CONTEXT**, **CUDA_ERROR_INVALID_VALUE**

Note:

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

See also:

cuGLMapBufferObject, **cuGLRegisterBufferObject**, **cuGLUnmapBufferObject**, **cuGLUnregisterBufferObject**, **cuGLUnmapBufferObjectAsync**, **cuGLSetBufferObjectMapFlags**

Author

Generated automatically by Doxygen from the source code.