GPU Framework

14.0.0.0

Generated by Doxygen 1.8.12

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Namespace Documentation

4.1 OpenGLRenderingEngine Namespace Reference

Namespace OpenGLRenderingEngine for the OpenGL rendering.

Namespaces

- OpenGLUtilityFunctions

  Namespace OpenGLUtilityFunctions for the OpenGL utility functions.

Classes

- struct OpenGLAssetManager

  This class encapsulates usage of an OpenGL Asset Manager.
- class OpenGLCameraAbstractBase

  This abstract class encapsulates usage of an OpenGL camera.
- class OpenGLDriverInfo

  Gets GL vendor, version, supported extensions and other states using glGet∗ functions and store them in OpenG↩ LDriverInfo class variables.
- class OpenGLEulerCamera

  This class encapsulates usage of an OpenGL Euler camera.
- class OpenGLFrameBufferObject

  This class provides Frame Buffer Object support using the GL_EXT_framebuffer_object OpenGL extension.
- class OpenGLQueryTimer

  This class contains an AccurateTimers encapsulation of OpenGL query timers.
- class OpenGLShaderCompileAndLink

  This class encapsulates loading, compilation & linking of a GLSL program.
- class OpenGLShaderGLSLPreProcessorCommands

  This class is responsible for the GLSL shader preprocessor process.
- class OpenGLShaderObjects

  This class is holding all shader objects GL handles and type information.
- class OpenGLShaderProgram

  This abstract class encapsulates usage of a GLSL program.
- class ShaderFilesGenerator

  This class includes shader files header/implementation generator related functionality.

4.1.1 Detailed Description

Namespace OpenGLRenderingEngine for the OpenGL rendering.

PLEASE DO NOT EDIT.

Author

Thanos Theo, 2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled CubeCapping_CubeCapping header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.2 OpenGLRenderingEngine::OpenGLUtilityFunctions Namespace Reference

Namespace OpenGLUtilityFunctions for the OpenGL utility functions.

Classes

- struct GLAuxiliaryFunctions
  This class contains only static CG & OpenGL related methods.

4.2.1 Detailed Description

Namespace OpenGLUtilityFunctions for the OpenGL utility functions.

Author

Thanos Theo, 2018

Version

14.0.0.0

4.3 OpenGLRenderingEngineTests Namespace Reference

Namespace OpenGLRenderingEngineTests for the OpenGL rendering engine tests.

Classes

- class ConfigFile

  This class encapsulates config file handling.
- class CubeCappingTest

  CubeCappingTest is the 1st set of OpenGL rendering tests.
- class TestAbstractBase

  TestAbstractBase is the abstract base class for all GLUT tests.
- struct TestGLUTInterface

  TestGLUTInterface is the interface (pure abstract class) for all GLUT tests (FreeGlut pure virtual void function to be implemented in sub-classes).

4.3.1 Detailed Description

Namespace OpenGLRenderingEngineTests for the OpenGL rendering engine tests.

Author

Thanos Theo, 2018

Version

14.0.0.0

4.4 Tests Namespace Reference

Namespace Tests for all relevant unit testing host & device (CPU & GPU) code.

Classes

- struct DeviceGoogleTest01__UTILS_CUDA_Class

  Device Google Test 01 for the UtilsCUDA::CUDADriverInfo class.
- struct DeviceGoogleTest02__UTILS_CUDA_Class

  Device Google Test 02 for the UtilsCUDA::CUDALinearAlgebraGPUComputing class.
- struct DeviceGoogleTest03__UTILS_CUDA_Classes

  Device Google Test 03 for the UtilsCUDA::CUDADriverInfo class CUDA Memory Registry functionality.
- struct DeviceGoogleTest04__UTILS_CUDA_Classes

  Device Google Test 04 for the UtilsCUDA::CUDAMemoryHandler set of classes functionality.
- struct HostGoogleTest01__UTILS_Class

  Host Google Test 01 for the Utils::AccurateTimers::AccurateCPUTimer class.
- struct HostGoogleTest02__UTILS_Class

  Host Google Test 02 for the Utils::Randomizers::RandomRNGWELL512 class.

- struct HostGoogleTest03__UTILS_Class

    Host Google Test 03 for the Utils::SIMDVectorizations classes.
- struct HostGoogleTest04__UTILS_Class

    Host Google Test 04 for the Utils::UtilityFunctions::BitManipulationFunctions class.
- struct HostGoogleTest05__UTILS_CPUParallelism_Class

    Host Google Test 05 for the Utils::CPUParallelism parallelFor() functionality.
- struct HostGoogleTest06__UTILS_CPUParallelism_Class

    Host Google Test 06 for the Utils::CPUParallelism::CPUParallelismUnitTests class for the parallelFor() functionality.
- struct HostGoogleTest07__Lodepng_Class

    Host Google Test 07 for the lodepng class for png encoding/decoding functionality.
- struct HostGoogleTest08__UTILS_Class

    Host Google Test 08 for the Utils::UtilityFunctions::MathFunctions class.

4.4.1 Detailed Description

Namespace Tests for all relevant unit testing host & device (CPU & GPU) code.

Author

Thanos Theo, 2018

Version

14.0.0.0

4.5 Utils Namespace Reference

Namespace Utils contains utility classes with mainly static CPU related methods.

Namespaces

- AccurateTimers

    Namespace AccurateTimers contains utility classes for accurate timer logging.
- CPUParallelism

    Namespace CPUParallelism encapsulates usage of the N-CP parallelism idea.
- Randomizers

    Namespace Randomizers contains random number generator classes.
- SIMDVectorizations

    Namespace SIMDVectorizations contains utility classes for SIMD vectorizations.
- UnitTests

    Namespace UnitTests contains classes used for unit testing.
- UtilityFunctions

    Namespace UtilityFunctions contains classes with only static CG GLSL-style & CPU related methods.
- VectorTypes

    Namespace VectorTypes provides CUDA-style float2-3-4 functionality.

Classes

- class FunctionView

  This class encapsulates usage of a function view (lightweight replacement of std::function).
- class FunctionView< Ret(Params...)>
- class NewHandlerSupport

  "Mixin-style" base class for class-specific std::set_new_handler support.
- struct ReverseIterationWrapper

  The ReverseIterationWrapper dummy struct provides additional generic functionality which std doesn't still provide.

Functions

- template<typename Container >
  auto **begin** (ReverseIterationWrapper< Container > wrapper)
- template<typename Container >
  auto **end** (ReverseIterationWrapper< Container > wrapper)
- template<typename Container >
  ReverseIterationWrapper< Container > **reverse** (Container &&iterable)

4.5.1 Detailed Description

Namespace Utils contains utility classes with mainly static CPU related methods.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.6 Utils::AccurateTimers Namespace Reference

Namespace AccurateTimers contains utility classes for accurate timer logging.

Classes

- class AccurateCPUTimer

  The AccurateCPUTimer class provides a concrete implementation of a high resolution CPU timer using the 'chrono' C++11 namespace.
- struct AccurateTimerInterface

  The AccurateTimerInterface struct encapsulates a basic interface for a generic high resolution timer.
- struct AccurateTimerLog

  The AccurateTimerLog struct is to be used for composition in timer related sub-classes through private inheritance.

4.6.1 Detailed Description

Namespace AccurateTimers contains utility classes for accurate timer logging.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.7 Utils::CPUParallelism Namespace Reference

Namespace CPUParallelism encapsulates usage of the N-CP parallelism idea.

Classes

- class ConcurrentBlockingQueue

  This class encapsulates usage of a concurrent blocking queue.
- class CPUParallelismUnitTests

  This class encapsulates unit testing of CPUParallelism libraries.
- class ThreadBarrier

  This class encapsulates usage of a thread barrier.
- class ThreadGuard

  This class encapsulates usage of a thread guard using std::move() & the RAII C++ idiom.
- class ThreadJoiner

  This class encapsulates usage of a vector< thread > joiner using the RAII C++ idiom.
- class ThreadPool

  This class encapsulates usage of a thread pool.

Functions

- UTILS_MODULE_API std::size_t numberOfHardwareThreads ()

  auxiliary parallelism functions
- UTILS_MODULE_API void **threadSleep** (std::size_t millisecs)
- UTILS_MODULE_API void parallelFor (std::size_t indexEnd, const FunctionView< void(std::size_t)> &kernelFunction, std::size_t numberOfThreads=numberOfHardwareThreads())

  parallelFor() versions with only the index provided
- UTILS_MODULE_API void **parallelFor** (std::size_t indexStart, std::size_t indexEnd, const FunctionView< void(std::size_t)> &kernelFunction, std::size_t numberOfThreads=numberOfHardwareThreads())
- UTILS_MODULE_API void parallelForThreadLocal (std::size_t indexEnd, const FunctionView< void(std↩ ::size_t, std::size_t)> &kernelFunction, std::size_t numberOfThreads=numberOfHardwareThreads())

  parallelFor() versions with both the index & threadId provided
- UTILS_MODULE_API void **parallelForThreadLocal** (std::size_t indexStart, std::size_t indexEnd, const FunctionView< void(std::size_t, std::size_t)> &kernelFunction, std::size_t numberOfThreads=numberOf↩ HardwareThreads())
- template<typename F , typename... Ts>
  auto reallyAsync (F &&f, Ts &&... params)

According to Scott Meyers, enforce task parallelism execution with the std::launch::async parameter in std::async().

- template< typename T >
  T atomicAdd (std::atomic< T > &value, T newValue, typename std::enable_if< std::is_floating_point< T >::value >::type ∗=nullptr)

  Perform an atomic addition to the T (decimal type only allowed for T, as C++ has specialized versions for integral types in its atomic library) via spin-locking on compare_exchange_weak(), the Compare-and-Swap (CAS) algorithm.
- template< typename T >
  T atomicMultiply (std::atomic< T > &value, T newValue, typename std::enable_if< std::is_floating_point< T >::value >::type ∗=nullptr)

  Perform an atomic multiply to the T (decimal type only allowed for T, as C++ has specialized versions for integral types in its atomic library) via spin-locking on compare_exchange_weak(), the Compare-and-Swap (CAS) algorithm.
- template< typename T >
  T atomicMin (std::atomic< T > &value, T newValue, typename std::enable_if< std::is_arithmetic< T >::value >::type ∗=nullptr)

  Perform an atomic min to the T (arithmetic type only allowed for T) via spin-locking on compare_exchange_weak(), the Compare-and-Swap (CAS) algorithm.
- template< typename T >
  T atomicMax (std::atomic< T > &value, T newValue, typename std::enable_if< std::is_arithmetic< T >↩ ::value >::type ∗=nullptr)

  Perform an atomic max to the T (arithmetic type only allowed for T) via spin-locking on compare_exchange_weak(), the Compare-and-Swap (CAS) algorithm.

4.7.1 Detailed Description

Namespace CPUParallelism encapsulates usage of the N-CP parallelism idea.

This namespace encapsulates usage of the N-CP parallelism idea.
CPUParallelism libraries originally based on with further extensions: `http://www.manning.com/williams/`.
The N-CP idea was based on: `http://www.biolayout.org/wp-content/uploads/2013/01/↩ Manuscript.pdf`.
Further inspiration was found here: `http://jcip.net.s3-website-us-east-1.amazonaws.com/`.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.8 Utils::Randomizers Namespace Reference

Namespace Randomizers contains random number generator classes.

Classes

- class ExponentialRandom

  The ExponentialRandom class provides a exponential random number generator.
- class NormalRandom

  The NormalRandom class provides a normal random number generator.
- class RandomRNGWELL512

  The RandomRNGWELL512 class provides the very fast RNG WELL512 algorithm random number generator initialized with a random integer.
- class UniformRandom

  The UniformRandom class provides a uniform random number generator.

4.8.1 Detailed Description

Namespace Randomizers contains random number generator classes.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.9 Utils::SIMDVectorizations Namespace Reference

Namespace SIMDVectorizations contains utility classes for SIMD vectorizations.

Classes

- class not_vec4

  The not_vec4 class is an internal class: not be used directly.
- class not_vec8

  The not_vec8 class is an internal class: not be used directly.
- class vec4

  The vec4 class is the main SIMD float4 class using the GLSL nomenclature.
- class vec4_unaligned

  The vec4_unaligned class is the main unaligned SIMD float4 class using the GLSL nomenclature.
- class vec8

  The vec8 class is the main SIMD float8 class using the GLSL nomenclature.
- class vec8_unaligned

  The vec8_unaligned class is the main unaligned SIMD float8 class using the GLSL nomenclature.

Functions

- vec4 **sqrt** (const vec4 &v)
- vec4 **rsqrt** (const vec4 &v)
- vec4 dot (const vec4 &a, const vec4 &b)

  Return value = dot product of a & b, replicated 4 times.
- vec8 **sqrt** (const vec8 &v)
- vec8 **rsqrt** (const vec8 &v)
- vec8 dot (const vec8 &a, const vec8 &b)

  Return value = dot product of a & b, replicated 8 times.
- bool isSupportedSSE3 ()

  Function to test for SSE3 support (x86 architecture).
- bool isSupportedAVX ()

  Function to test for AVX support (x86 architecture).
- bool isSupportedAVX2 ()

  Function to test for AVX2 support (x86 architecture).
- bool isSupportedNEON ()

  Function to test for NEON support (ARM NEON SIMD architecture).
- void **memcpy_GL_matrices_SSE** (float ∗__restrict destination, const float ∗__restrict source)
- void **memcpy_unaligned_GL_matrices_SSE** (float ∗__restrict destination, const float ∗__restrict source)
- void **memcpy_GL_matrices_AVX** (float ∗__restrict destination, const float ∗__restrict source)
- void **memcpy_unaligned_GL_matrices_AVX** (float ∗__restrict destination, const float ∗__restrict source)
- std::array< float, 16 > **convert_to_float_GL_matrix_SSE** (const double ∗__restrict source)

4.9.1 Detailed Description

Namespace SIMDVectorizations contains utility classes for SIMD vectorizations.

SIMDVectorizations.h:

These classes encapsulate the SSE/AVX SIMD instructions on Intel Hardware in an syntactical GLSL-friendly way. Originally based on with further extensions: `https://www.cs.uaf.edu/2011/fall/cs441/lecture/09↩_29_SSE.html`.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.9.2 Function Documentation

4.9.2.1 dot() [1/2]

```
vec4 Utils::SIMDVectorizations::dot (
            const vec4 & a,
            const vec4 & b )  [inline]
```

Return value = dot product of a & b, replicated 4 times.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.9.2.2 dot() [2/2]

```
vec8 Utils::SIMDVectorizations::dot (
            const vec8 & a,
            const vec8 & b )  [inline]
```

Return value = dot product of a & b, replicated 8 times.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.9.2.3 isSupportedAVX()

```
bool Utils::SIMDVectorizations::isSupportedAVX ( )  [inline]
```

Function to test for AVX support (x86 architecture).

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.9.2.4 isSupportedAVX2()

```
bool Utils::SIMDVectorizations::isSupportedAVX2 ( )  [inline]
```

Function to test for AVX2 support (x86 architecture).

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.9.2.5 isSupportedNEON()

```
bool Utils::SIMDVectorizations::isSupportedNEON ( )  [inline]
```

Function to test for NEON support (ARM NEON SIMD architecture).

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.9.2.6 isSupportedSSE3()

```
bool Utils::SIMDVectorizations::isSupportedSSE3 ( )  [inline]
```

Function to test for SSE3 support (x86 architecture).

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.10 Utils::UnitTests Namespace Reference

Namespace UnitTests contains classes used for unit testing.

Classes

- struct UnitTestInterface

  The UnitTestInterface struct encapsulate a basic unit test interface.
- class UnitTestUtilityFunctions

  The UnitTestUtilityFunctions class adds unit testing utility function support through private inheritance.

Typedefs

- using **UnitTestUtilityFunctions_flt** = UnitTestUtilityFunctions< float >
- using **UnitTestUtilityFunctions_dbl** = UnitTestUtilityFunctions< double >

4.10.1 Detailed Description

Namespace UnitTests contains classes used for unit testing.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.11 Utils::UtilityFunctions Namespace Reference

Namespace UtilityFunctions contains classes with only static CG GLSL-style & CPU related methods.

Classes

- struct ArrayIndicingFunctions

  The ArrayIndicingFunctions class provides array indexing functionality.
- struct Base64CompressorScrambler

  The Base64CompressorScrambler class provides encoding/decoding functionality to strings.
- struct BitManipulationFunctions

  The BitManipulationFunctions class provides bit manipulation functionality.
- class DebugConsole

  The DebugConsole class provides debugging & logging functionality.
- struct MathFunctions

  The MathFunctions class provides some needed mathematical functions functionality (note that some functions emulate GLSL-style CPU functionality).
- struct StdAuxiliaryFunctions

  The StdAuxiliaryFunctions class provides additional generic functionality which std doesn't (currently) still provide.
- class StdReadWriteFileFunctions

  The StdReadWriteFileFunctions class provides additional i/o functionality.
- class StringAuxiliaryFunctions

  The StringAuxiliaryFunctions class provides additional string functionality which std doesn't (currently) still provide.

4.11.1 Detailed Description

Namespace UtilityFunctions contains classes with only static CG GLSL-style & CPU related methods.

UtilityFunctions.h:

Namespace UtilityFunctions contains classes with only static CG GLSL-style & CPU related methods.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.12 Utils::VectorTypes Namespace Reference

Namespace VectorTypes provides CUDA-style float2-3-4 functionality.

Classes

- struct double2

  The double2 class provides double2 functionality.
- struct double3

  The double3 class provides double3 functionality.
- struct double4

  The double4 class provides double4 functionality.
- struct float2

  The float2 class provides float2 functionality.
- struct float3

  The float3 class provides float3 functionality.
- struct float4

  The float4 class provides float4 functionality.

4.12.1 Detailed Description

Namespace VectorTypes provides CUDA-style float2-3-4 functionality.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.13 UtilsCUDA Namespace Reference

namespace UtilsCUDA for encapsulating all the CUDA related code compiled by the NVCC compiler.

Classes

- struct CUDADeleter
- class CUDADriverInfo

  This class encapsulates CUDA driver info for detection & reporting.
- class CUDAEventTimer

  This class contains an AccurateTimers encapsulation of CUDA event timers.
- class CUDAGPUComputingAbstraction

  This class encapsulates a basic abstraction layer for CUDA GPU Computing.
- class CUDALinearAlgebraGPUComputing

  This class contains a basic Linear Algebra GPU Computing test case in CUDA.
- class CUDAMemoryRegistry

  This class encapsulates CUDA memory registry functionality for both host & device with reporting.
- class CUDASpinLock

  This class is based on the book'The CUDA Handbook - A comprehensive Guide to GPU Programming'.
- class CUDAStreamsHandler

  This class encapsulates usage of a collection of CUDA streams & the RAII C++ idiom.

- struct CUDAUtilityFunctions

  This class encapsulates all the CUDA related utility functions.
- class DeviceMemory

  This class encapsulates usage of a collection of CUDA memory handling techniques (device only) & the RAII C++ idiom.
- class HostDeviceMemory

  This class encapsulates usage of a collection of host & CUDA memory handling techniques (host & device) & the RAII C++ idiom.
- struct OutputTypes

  Usage of a C-style enum (not typesafe C++11 enum class) to be able to use a viz-style bitwise flag OR API on enum values.
- struct PinnedDeleter

Typedefs

- template< typename T >
  using **DeviceUniquePtr** = std::unique_ptr< T, CUDADeleter< T >>
- template< typename T >
  using **PinnedUniquePtr** = std::unique_ptr< T, PinnedDeleter< T >>

Functions

- template< typename T >
  DeviceUniquePtr< T > **make_unique_device** (std::size_t numberOfElements, int device=0, bool use←UnifiedMemory=false) noexcept
- template< typename T >
  PinnedUniquePtr< T > **make_unique_pinned** (std::size_t numberOfElements) noexcept
- template< typename T >
  std::future< DeviceUniquePtr< T > > **make_unique_device_async** (std::size_t numberOfElements, int device=0, bool useUnifiedMemory=false) noexcept
- template< typename T >
  std::future< PinnedUniquePtr< T > > **make_unique_pinned_async** (std::size_t numberOfElements) noexcept

4.13.1 Detailed Description

namespace UtilsCUDA for encapsulating all the CUDA related code compiled by the NVCC compiler.

Author

Thanos Theo, 2018

Version

14.0.0.0

4.14 UtilsCUDAKernels Namespace Reference

namespace UtilsCUDAKernels for encapsulating all the CUDA kernels of the GPU Framework.

Functions

- __device__ __forceinline__ void kernelAdd1DArray (const int32_t ∗__restrict a, const int32_t ∗__restrict b, int32_t ∗__restrict c, uint32_t arraySizeXY)

    kernelAdd1DArray() function to perform c = a + b with array indices using arraySizeXY as a check.
- __device__ __forceinline__ void kernelAdd2DArray (const int32_t ∗__restrict a, const int32_t ∗__restrict b, int32_t ∗__restrict c, uint32_t powerOfTwoDimension)

    kernelAdd2DArray() function to perform c = a + b with array indices using arraySizeX & arraySizeY as a check.

4.14.1 Detailed Description

namespace UtilsCUDAKernels for encapsulating all the CUDA kernels of the GPU Framework.

Author

Thanos Theo, 2018

Version

14.0.0.0

4.14.2 Function Documentation

4.14.2.1 kernelAdd1DArray()

```
__device__ __forceinline__ void UtilsCUDAKernels::kernelAdd1DArray (
             const int32_t ∗__restrict a,
             const int32_t ∗__restrict b,
             int32_t ∗__restrict c,
             uint32_t arraySizeXY )
```

kernelAdd1DArray() function to perform c = a + b with array indices using arraySizeXY as a check.

kernelAdd1DArray():

- Currently, we get the index by using blockDim.x ∗ gridDim.x as the 'scanline width', which may be larger than the array size in X (for a 2D array), but will give a unique index, and we check the combined index against the one dimensional array size. This way, any unused threads are on the last blocks in y, so visualizing a 2D array of threads, all lines except the last few are completely filled, then there is one half empty line, and the last few lines are empty.

kernelAdd2DArray():

- We can also check that both the xIndex and yIndex are within the powerOfTwoDimension (X dimension) and powerOfTwoDimension (Y dimension), but in this case we should also use the powerOfTwoDimension as the scanline width. This way, threads that do nothing are on the ends of the array both in x and y, so visualizing the threads as a 2D grid, the last few threads of each line are empty, and the last few lines are empty.
- The first method will be a tiny bit faster for problems without 2D coherency, as more blocks will be completely filled, so fewer warps are needed.
- The second method will be significantly faster for problems with 2D coherency, as threads within the same block will be much more likely to follow the same code path.
- Also note that for problems with powerOfTwoDimension a multiple of the blockDim.x the two methods are identical, as powerOfTwoDimension == blockDim.x ∗ gridDim.x. That is valid for our CUDALinearAlgebra←GPUComputing test example here, thus no runtime difference will be noticed with either of the two ways described above.

Author

Thanos Theo, 2018

Version

14.0.0.0

4.14.2.2 kernelAdd2DArray()

```
__device__ __forceinline__ void UtilsCUDAKernels::kernelAdd2DArray (
            const int32_t *__restrict a,
            const int32_t *__restrict b,
            int32_t *__restrict c,
            uint32_t powerOfTwoDimension )
```

kernelAdd2DArray() function to perform c = a + b with array indices using arraySizeX & arraySizeY as a check.

kernelAdd1DArray():

- Currently, we get the index by using blockDim.x ∗ gridDim.x as the 'scanline width', which may be larger than the array size in X (for a 2D array), but will give a unique index, and we check the combined index against the one dimensional array size. This way, any unused threads are on the last blocks in y, so visualizing a 2D array of threads, all lines except the last few are completely filled, then there is one half empty line, and the last few lines are empty.

kernelAdd2DArray():

- We can also check that both the xIndex and yIndex are within the powerOfTwoDimension (X dimension) and powerOfTwoDimension (Y dimension), but in this case we should also use the powerOfTwoDimension as the scanline width. This way, threads that do nothing are on the ends of the array both in x and y, so visualizing the threads as a 2D grid, the last few threads of each line are empty, and the last few lines are empty.
- The first method will be a tiny bit faster for problems without 2D coherency, as more blocks will be completely filled, so fewer warps are needed.
- The second method will be significantly faster for problems with 2D coherency, as threads within the same block will be much more likely to follow the same code path.
- Also note that for problems with powerOfTwoDimension a multiple of the blockDim.x the two methods are identical, as powerOfTwoDimension == blockDim.x ∗ gridDim.x. That is valid for our CUDALinearAlgebra←GPUComputing test example here, thus no runtime difference will be noticed with either of the two ways described above.

Author

Thanos Theo, 2018

Version

14.0.0.0

Chapter 5

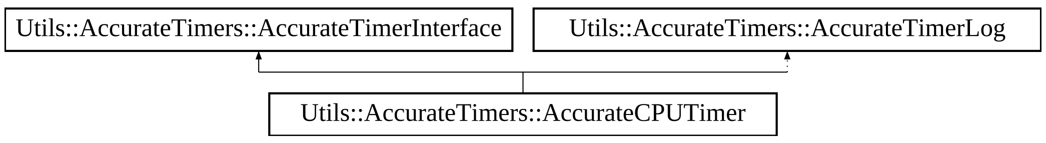
Class Documentation

5.1 Utils::AccurateTimers::AccurateCPUTimer Class Reference

The AccurateCPUTimer class provides a concrete implementation of a high resolution CPU timer using the 'chrono' C++11 namespace.

```
#include <AccurateTimers.h>
```

Inheritance diagram for Utils::AccurateTimers::AccurateCPUTimer:


Utils::AccurateTimers::AccurateTimerInterface
Utils::AccurateTimers::AccurateTimerLog
Utils::AccurateTimers::AccurateCPUTimer

Public Member Functions

- void **startTimer** () override
- void **stopTimer** () override
- double **getElapsedTimeInNanoSecs** () override
- double **getElapsedTimeInMicroSecs** () override
- double **getElapsedTimeInMilliSecs** () override
- double **getElapsedTimeInSecs** () override
- double **getMeanTimeInNanoSecs** () override
- double **getMeanTimeInMicroSecs** () override
- double **getMeanTimeInMilliSecs** () override
- double **getMeanTimeInSecs** () override
- double **getDecimalElapsedTimeInMicroSecs** () override
- double **getDecimalElapsedTimeInMilliSecs** () override
- double **getDecimalElapsedTimeInSecs** () override
- double **getDecimalMeanTimeInMicroSecs** () override
- double **getDecimalMeanTimeInMilliSecs** () override
- double **getDecimalMeanTimeInSecs** () override
- **AccurateCPUTimer** (const AccurateCPUTimer &)=delete
- **AccurateCPUTimer** (AccurateCPUTimer &&)=delete
- AccurateCPUTimer & **operator=** (const AccurateCPUTimer &)=delete
- AccurateCPUTimer & **operator=** (AccurateCPUTimer &&)=delete

Static Public Member Functions

- static std::uint64_t **getNanosecondsTimeSinceEpoch** ()
- static std::uint64_t **getMicrosecondsTimeSinceEpoch** ()
- static std::uint64_t **getMillisecondsTimeSinceEpoch** ()
- static std::uint64_t **getSecondsTimeSinceEpoch** ()

Private Member Functions

- template<typename ChronoType >
  double **getElapsedTime** ()

Private Attributes

- std::chrono::high_resolution_clock::time_point **_start** = std::chrono::high_resolution_clock::now()
- std::chrono::high_resolution_clock::time_point **_stop** = std::chrono::high_resolution_clock::now()

Additional Inherited Members

5.1.1 Detailed Description

The AccurateCPUTimer class provides a concrete implementation of a high resolution CPU timer using the 'chrono' C++11 namespace.

Note: no virtual destructor is needed for data-oriented design ie no up-casting should ever be used.

Author

Thanos Theo, 2009-2018

Version

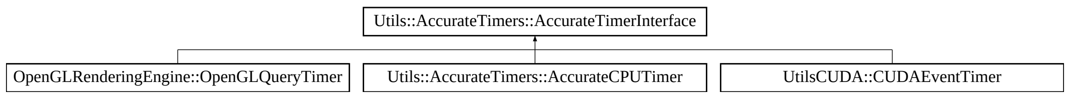
14.0.0.0

5.2 Utils::AccurateTimers::AccurateTimerInterface Struct Reference

The AccurateTimerInterface struct encapsulates a basic interface for a generic high resolution timer.

```
#include <AccurateTimers.h>
```

Inheritance diagram for Utils::AccurateTimers::AccurateTimerInterface:


Utils::AccurateTimers::AccurateTimerInterface
OpenGLRenderingEngine::OpenGLQueryTimer
Utils::AccurateTimers::AccurateCPUTimer
UtilsCUDA::CUDAEventTimer

Public Member Functions

- virtual void **startTimer** ()=0
- virtual void **stopTimer** ()=0
- virtual double **getElapsedTimeInNanoSecs** ()=0
- virtual double **getElapsedTimeInMicroSecs** ()=0
- virtual double **getElapsedTimeInMilliSecs** ()=0
- virtual double **getElapsedTimeInSecs** ()=0
- virtual double **getMeanTimeInNanoSecs** ()=0
- virtual double **getMeanTimeInMicroSecs** ()=0
- virtual double **getMeanTimeInMilliSecs** ()=0
- virtual double **getMeanTimeInSecs** ()=0
- virtual double **getDecimalElapsedTimeInMicroSecs** ()=0
- virtual double **getDecimalElapsedTimeInMilliSecs** ()=0
- virtual double **getDecimalElapsedTimeInSecs** ()=0
- virtual double **getDecimalMeanTimeInMicroSecs** ()=0
- virtual double **getDecimalMeanTimeInMilliSecs** ()=0
- virtual double **getDecimalMeanTimeInSecs** ()=0
- **AccurateTimerInterface** (const AccurateTimerInterface &)=delete
- **AccurateTimerInterface** (AccurateTimerInterface &&)=delete
- AccurateTimerInterface & **operator=** (const AccurateTimerInterface &)=delete
- AccurateTimerInterface & **operator=** (AccurateTimerInterface &&)=delete

5.2.1 Detailed Description

The AccurateTimerInterface struct encapsulates a basic interface for a generic high resolution timer.

Author

Thanos Theo, 2009-2018

Version

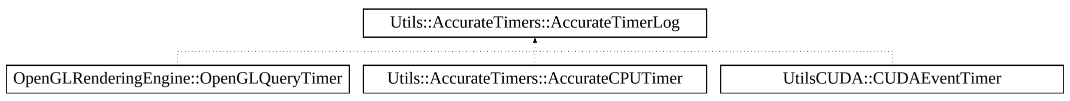
14.0.0.0

5.3 Utils::AccurateTimers::AccurateTimerLog Struct Reference

The AccurateTimerLog struct is to be used for composition in timer related sub-classes through private inheritance.

```
#include <AccurateTimers.h>
```

Inheritance diagram for Utils::AccurateTimers::AccurateTimerLog:


Utils::AccurateTimers::AccurateTimerLog
OpenGLRenderingEngine::OpenGLQueryTimer
Utils::AccurateTimers::AccurateCPUTimer
UtilsCUDA::CUDAEventTimer

Public Types

- enum **TimerTypes** : std::size_t { **NANOSECS** = 0, **MICROSECS** = 1, **MILLISECS** = 2, **SECS** = 3 }

Public Member Functions

- **AccurateTimerLog** (const AccurateTimerLog &)=delete
- **AccurateTimerLog** (AccurateTimerLog &&)=delete
- AccurateTimerLog & **operator=** (const AccurateTimerLog &)=delete
- AccurateTimerLog & **operator=** (AccurateTimerLog &&)=delete

Static Public Member Functions

- static double calculateMeanTime (double currentTime, double *__restrict timersBookKeeping, std::int64_t &timersBookKeepingIndex, bool &firstTimersBookKeepingIterationCompleted)

  The implementation below is based on BitSquid's Time Step Smoothing article.

Public Attributes

- double **_timersBookKeeping** [NUMBER_OF_TIMER_FORMATS][TIMERS_BOOK_KEEPING_SIZE] = { { 0.0 } }
- std::array< std::int64_t, NUMBER_OF_TIMER_FORMATS > **_timersBookKeepingIndex** { { 0 } }
- std::array< bool, NUMBER_OF_TIMER_FORMATS > **_firstTimersBookKeepingIterationCompleted** { { false } }
- bool **_stopped** = false

Static Public Attributes

- static constexpr double **NANO_TO_MICROSECS_CONVERSION** = 1000.0
- static constexpr double **NANO_TO_MILLISECS_CONVERSION** = 1000000.0
- static constexpr double **NANO_TO_SECS_CONVERSION** = 1000000000.0
- static constexpr std::size_t **NUMBER_OF_TIMER_FORMATS** = 4
- static constexpr std::size_t **TIMERS_BOOK_KEEPING_SIZE** = 11

5.3.1 Detailed Description

The AccurateTimerLog struct is to be used for composition in timer related sub-classes through private inheritance.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.3.2 Member Function Documentation

5.3.2.1 calculateMeanTime()

```
double AccurateTimerLog::calculateMeanTime (
             double currentTime,
             double *__restrict timersBookKeeping,
             std::int64_t & timersBookKeepingIndex,
             bool & firstTimersBookKeepingIterationCompleted )  [static]
```

The implementation below is based on BitSquid's Time Step Smoothing article.

The implementation below is based on BitSquid's Time Step Smoothing article: http://bitsquid.↩ blogspot.se/2010/10/time-step-smoothing.html It does it in 4 main steps: 1) Keep a history of the time step for the last 11 frames. 2) Throw away the outliers, the two highest and the two lowest values. 3) Calculate the mean of the remaining 7 values. 4) Lerp from the time step for the last frame to the calculated mean (adding more smoothness)

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.4 OpenGLRenderingEngine::GLSLShaderFiles::AllGLSLShaderFiles Class Reference

Public Member Functions

- std::tuple< const char ∗const ∗, std::size_t > **getShader** (const std::string &name)
- **AllGLSLShaderFiles** (const AllGLSLShaderFiles &)=delete
- **AllGLSLShaderFiles** (AllGLSLShaderFiles &&)=delete
- AllGLSLShaderFiles & **operator=** (const AllGLSLShaderFiles &)=delete
- AllGLSLShaderFiles & **operator=** (AllGLSLShaderFiles &&)=delete

Static Public Member Functions

- static AllGLSLShaderFiles & **getSingleton** ()

Private Attributes

- std::unordered_map< std::string, std::tuple< const char ∗const ∗, std::size_t > > **_allGLSLShaderFiles**

5.5 Utils::UtilityFunctions::ArrayIndicingFunctions Struct Reference

The ArrayIndicingFunctions class provides array indexing functionality.

```
#include <UtilityFunctions.h>
```

Public Member Functions

- **ArrayIndicingFunctions** (const ArrayIndicingFunctions &)=delete
- **ArrayIndicingFunctions** (ArrayIndicingFunctions &&)=delete
- ArrayIndicingFunctions & **operator=** (const ArrayIndicingFunctions &)=delete
- ArrayIndicingFunctions & **operator=** (ArrayIndicingFunctions &&)=delete

Static Public Member Functions

- static std::size_t flattenArray2DIndex (std::size_t x, std::size_t y, std::size_t dimensionY)

  Flattens the 2D array coordinates to an 1D index.
- static std::tuple< size_t, size_t > unflattenArray2DIndex (std::size_t array2DIndex, std::size_t dimensionY)

  Unflattens the 1D array index to 2D array coordinates.
- template<typename T >
  static T getArray2D (const T *__restrict array2D, std::size_t x, std::size_t y, std::size_t dimensionY)

  Getter from a 2D array laid out linearly in memory.
- template<typename T >
  static void setArray2D (T *__restrict array2D, std::size_t x, std::size_t y, std::size_t dimensionY, const T &value)

  Setter for a 2D array laid out linearly in memory.
- static std::size_t flattenArray3DIndex (std::size_t x, std::size_t y, std::size_t z, std::size_t dimensionY, std↩ ::size_t dimensionZ)

  Flattens the 3D array coordinates to an 1D index.
- static std::tuple< size_t, size_t, size_t > unflattenArray3DIndex (std::size_t array3DIndex, std::size_t dimensionY, std::size_t dimensionZ)

  Unflattens the 1D array index to 3D array coordinates.
- template<typename T >
  static T getArray3D (const T *__restrict array3D, std::size_t x, std::size_t y, std::size_t z, std::size_t dimensionY, std::size_t dimensionZ)

  Getter from a 3D array laid out linearly in memory.
- template<typename T >
  static void setArray3D (T *__restrict array3D, std::size_t x, std::size_t y, std::size_t z, std::size_t dimensionY, std::size_t dimensionZ, const T &value)

  Setter for a 3D array laid out linearly in memory.

5.5.1 Detailed Description

The ArrayIndicingFunctions class provides array indexing functionality.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.6 Utils::UtilityFunctions::Base64CompressorScrambler Struct Reference

The Base64CompressorScrambler class provides encoding/decoding functionality to strings.

```
#include <UtilityFunctions.h>
```

Public Member Functions

- **Base64CompressorScrambler** (const Base64CompressorScrambler &)=delete
- **Base64CompressorScrambler** (Base64CompressorScrambler &&)=delete
- Base64CompressorScrambler & **operator=** (const Base64CompressorScrambler &)=delete
- Base64CompressorScrambler & **operator=** (Base64CompressorScrambler &&)=delete

Static Public Member Functions

- static std::string **encodeBase64String** (const std::string &str)
- static std::string **decodeBase64String** (const std::string &str)
- static std::string **compressString** (const std::string &str)
- static std::string **decompressString** (const std::string &str)
- static std::string **flipString** (const std::string &line)
- static std::string **xorSwapString** (const std::string &line)

5.6.1 Detailed Description

The Base64CompressorScrambler class provides encoding/decoding functionality to strings.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.7 Utils::UtilityFunctions::BitManipulationFunctions Struct Reference

The BitManipulationFunctions class provides bit manipulation functionality.

```
#include <UtilityFunctions.h>
```

Public Member Functions

- **BitManipulationFunctions** (const BitManipulationFunctions &)=delete
- **BitManipulationFunctions** (BitManipulationFunctions &&)=delete
- BitManipulationFunctions & **operator=** (const BitManipulationFunctions &)=delete
- BitManipulationFunctions & **operator=** (BitManipulationFunctions &&)=delete

Static Public Member Functions

- static bool isPowerOfTwo (int value)

  Find if the given number is a power-of-two number.
- static int getLowestBitPositionOfPowerOfTwoNumber (int value)

  Find the lowest bit position of a given power-of-two integer number.
- static int countTurnedOnBitsOfNumber (int value)

  Count turned on bits of a given integer number.
- static unsigned int getNextPowerOfTwo (unsigned int value)

  Gets the next power-of-two of a given number.
- template<typename T , typename I >
  static bool hasCStyleEnumType (T enumType, I enumSelection)

  Checks if the enumType has the enumSelection (for C-style enums).
- template<typename T , typename I >
  static bool hasClassEnumType (T enumType, I enumSelection)

  Checks if the enumType has the enumSelection (for C++11 class enums).

5.7.1 Detailed Description

The BitManipulationFunctions class provides bit manipulation functionality.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.7.2 Member Function Documentation

5.7.2.1 countTurnedOnBitsOfNumber()

```
int BitManipulationFunctions::countTurnedOnBitsOfNumber (
            int value )  [static]
```

Count turned on bits of a given integer number.

Extremely efficient implementation taken from `http://graphics.stanford.edu/~seander/bithacks.↩ html`

5.7.2.2 getLowestBitPositionOfPowerOfTwoNumber()

```
int BitManipulationFunctions::getLowestBitPositionOfPowerOfTwoNumber (
            int value )  [static]
```

Find the lowest bit position of a given power-of-two integer number.

Extremely efficient implementation taken from `http://graphics.stanford.edu/~seander/bithacks.↩ html`

5.7.2.3 getNextPowerOfTwo()

```
unsigned int BitManipulationFunctions::getNextPowerOfTwo (
             unsigned int value )  [static]
```

Gets the next power-of-two of a given number.

Extremely efficient implementation taken from http://graphics.stanford.edu/~seander/bithacks.↩ html

5.7.2.4 hasClassEnumType()

```
template<typename T , typename I >
static bool Utils::UtilityFunctions::BitManipulationFunctions::hasClassEnumType (
             T enumType,
             I enumSelection )  [inline], [static]
```

Checks if the enumType has the enumSelection (for C++11 class enums).

Using the extremely efficient getLowestBitPositionOfPowerOfTwoNumber() implementation.

5.7.2.5 hasCStyleEnumType()

```
template<typename T , typename I >
static bool Utils::UtilityFunctions::BitManipulationFunctions::hasCStyleEnumType (
             T enumType,
             I enumSelection )  [inline], [static]
```

Checks if the enumType has the enumSelection (for C-style enums).

Using the extremely efficient getLowestBitPositionOfPowerOfTwoNumber() implementation.

5.7.2.6 isPowerOfTwo()

```
static bool Utils::UtilityFunctions::BitManipulationFunctions::isPowerOfTwo (
             int value )  [inline], [static]
```

Find if the given number is a power-of-two number.

Extremely efficient implementation taken from http://graphics.stanford.edu/~seander/bithacks.↩ html

5.8 Utils::CPUParallelism::ConcurrentBlockingQueue< T > Class Template Reference

This class encapsulates usage of a concurrent blocking queue.

```
#include <ConcurrentBlockingQueue.h>
```

Public Member Functions

- **ConcurrentBlockingQueue** (const ConcurrentBlockingQueue &other)=delete
- **ConcurrentBlockingQueue** (ConcurrentBlockingQueue &&other)=delete
- ConcurrentBlockingQueue & **operator=** (const ConcurrentBlockingQueue &other)=delete
- ConcurrentBlockingQueue & **operator=** (ConcurrentBlockingQueue &&other)=delete
- void **waitAndPop** (T &value)
- bool **tryPop** (T &value)
- std::shared_ptr< T > **waitAndPop** ()
- std::shared_ptr< T > **tryPop** ()
- void **emplace** (T newValue)
- bool **empty** () const

Private Attributes

- std::mutex **_dataMutex**
- std::queue< std::shared_ptr< T > > **_dataQueue**
- std::condition_variable **_dataCondition**

5.8.1 Detailed Description

template<typename T>
class Utils::CPUParallelism::ConcurrentBlockingQueue< T >

This class encapsulates usage of a concurrent blocking queue.

ConcurrentBlockingQueue.h:

This class encapsulates usage of a concurrent blocking queue.
CPUParallelism libraries originally based on with further extensions: `http://www.manning.com/williams/`.
The N-CP idea was based on: `http://www.biolayout.org/wp-content/uploads/2013/01/↩ Manuscript.pdf`.
Further inspiration was found here: `http://jcip.net.s3-website-us-east-1.amazonaws.com/`.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.9 OpenGLRenderingEngineTests::ConfigFile Class Reference

This class encapsulates config file handling.

```
#include <ConfigFile.h>
```

Public Member Functions

- bool **getFullScreen** () const
- bool **getMultiSample** () const
- std::size_t **getTest** () const
- **ConfigFile** (const ConfigFile &)=delete
- **ConfigFile** (ConfigFile &&)=delete
- ConfigFile & **operator=** (const ConfigFile &)=delete
- ConfigFile & **operator=** (ConfigFile &&)=delete

Static Public Member Functions

- static std::string **getConfigFileName** ()

Private Member Functions

- std::string **createDefaultConfigFileFromParameters** () const
- void **parseParametersFromConfigFile** (const std::list< std::string > &configFileLines)

Private Attributes

- bool **_fullScreen** = false
- bool **_multiSample** = true
- std::size_t **_test** = 1

5.9.1 Detailed Description

This class encapsulates config file handling.

Author

Thanos Theo, 2009-2018

Version

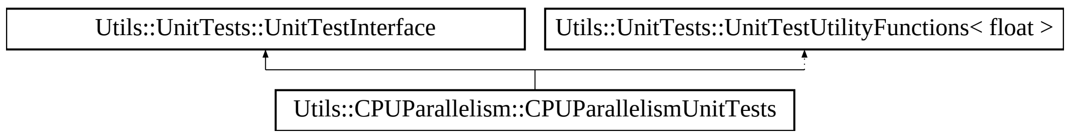
14.0.0.0

5.10 Utils::CPUParallelism::CPUParallelismUnitTests Class Reference

This class encapsulates unit testing of CPUParallelism libraries.

```
#include <CPUParallelismUnitTests.h>
```

Inheritance diagram for Utils::CPUParallelism::CPUParallelismUnitTests:


Utils::UnitTests::UnitTestInterface
Utils::UnitTests::UnitTestUtilityFunctions< float >
Utils::CPUParallelism::CPUParallelismUnitTests

Public Member Functions

- **CPUParallelismUnitTests** (std::size_t dimensions=512, std::size_t numberOfThreads=numberOf↩ HardwareThreads(), bool useRandomness=false) noexcept
- **CPUParallelismUnitTests** (std::tuple< std::size_t, std::size_t, std::size_t > dimensionsXYZ, std::size_↩ t numberOfThreads=numberOfHardwareThreads(), bool useRandomness=false) noexcept
- **CPUParallelismUnitTests** (const CPUParallelismUnitTests &)=delete
- **CPUParallelismUnitTests** (CPUParallelismUnitTests &&)=delete
- CPUParallelismUnitTests & **operator=** (const CPUParallelismUnitTests &)=delete
- CPUParallelismUnitTests & **operator=** (CPUParallelismUnitTests &&)=delete
- void **resetTests** () override
- bool **conductTests** () override
- void **reportTestResults** () const override

Private Attributes

- std::size_t **_dimensionX** = 512
- std::size_t **_dimensionY** = 512
- std::size_t **_dimensionZ** = 512
- std::size_t **_numberOfThreads** = numberOfHardwareThreads()
- bool **_useRandomness** = false
- int **_testIterations** = 0
- double **_meanTimeCounterRandomizer** = 0.0
- double **_meanTimeCounterSingleCore** = 0.0
- double **_meanTimeCounterNCP** = 0.0

Additional Inherited Members

5.10.1 Detailed Description

This class encapsulates unit testing of CPUParallelism libraries.

CPUParallelismUnitTests.h:

This class encapsulates unit testing of CPUParallelism libraries.
CPUParallelism libraries originally based on with further extensions: `http://www.manning.com/williams/`.
The N-CP idea was based on: `http://www.biolayout.org/wp-content/uploads/2013/01/↩ Manuscript.pdf`.
Further inspiration was found here: `http://jcip.net.s3-website-us-east-1.amazonaws.com/`.

Author

Thanos Theo, 2009-2018

Version

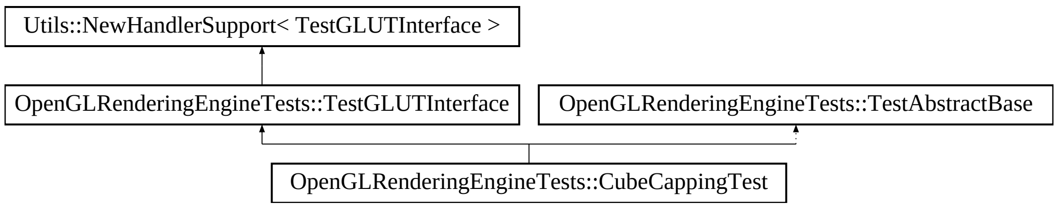
14.0.0.0

5.11 OpenGLRenderingEngineTests::CubeCappingTest Class Reference

CubeCappingTest is the 1st set of OpenGL rendering tests.

```
#include <CubeCappingTest.h>
```

Inheritance diagram for OpenGLRenderingEngineTests::CubeCappingTest:


Utils::NewHandlerSupport< TestGLUTInterface >
OpenGLRenderingEngineTests::TestGLUTInterface
OpenGLRenderingEngineTests::TestAbstractBase
OpenGLRenderingEngineTests::CubeCappingTest

Classes

- class OpenGLShaderCubeCapping

Public Member Functions

- void **renderScene** () override
- void **changeSize** (int w, int h) override
- void **keyboard** (unsigned char key, int x, int y) override
- void **specialKeysKeyboard** (int key, int x, int y) override
- void **mouse** (int button, int state, int x, int y) override
- void **mouseMotion** (int x, int y) override
- void **closeFunc** () override
- **CubeCappingTest** (int screenWidth, int screenHeight, bool multisample) noexcept
- **CubeCappingTest** (const CubeCappingTest &)=delete
- **CubeCappingTest** (CubeCappingTest &&)=delete
- CubeCappingTest & **operator=** (const CubeCappingTest &)=delete
- CubeCappingTest & **operator=** (CubeCappingTest &&)=delete

Private Types

- enum **AllCachedRenderingTests** : std::size_t { **DRAW_ARRAYS** = 0, **DRAW_ELEMENTS** = 1, **DRAW_↩ RANGE_ELEMENTS** = 2 }

Private Member Functions

- void **prepareCubeCappingShaders** ()
- void **prepareCubeCappingFBO** ()
- void **initCubeCappingFBO** () const
- void **prepareVBOs** ()
- void **deleteVBOs** ()
- void **clearScreen** () const
- void **renderCubeClippingPlane** () const
- void **renderCube** () const
- void **renderCubeScene** () const
- void **drawString** (const char ∗str, int x, int y, const GLfloat color[4], void ∗font) const
- void **drawString3D** (const char ∗str, float position[3], const GLfloat color[4], void ∗font) const
- void **showInfo** ()
- void **showFPS** ()

Private Attributes

- AllCachedRenderingTests **currentCachedRenderingTest** = DRAW_ARRAYS
- OpenGLRenderingEngineTests::CubeCappingTest::OpenGLShaderCubeCapping ∗ **openGLShaderCube←Capping** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject ∗ **openGLFrameBufferObjectForCubeCapping** = nullptr
- GLuint **VBOVerticesID** = 0
- GLuint **VBONormalsID** = 0
- GLuint **VBOTexCoordsID** = 0
- GLuint **VBOColorsID** = 0
- GLuint **VBOIndicesID** = 0
- bool **useCubeCapping** = true
- GLfloat **clipPlaneZ** = 0.0f

Additional Inherited Members

5.11.1 Detailed Description

CubeCappingTest is the 1st set of OpenGL rendering tests.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.12 UtilsCUDA::CUDADeleter< T > Struct Template Reference

Public Member Functions

- **CUDADeleter** (int device=0) noexcept
- void **operator()** (T ∗ptr) noexcept

Public Attributes

- int **device** = 0

5.13 UtilsCUDA::CUDADriverInfo Class Reference

This class encapsulates CUDA driver info for detection & reporting.

```
#include <CUDADriverInfo.h>
```

Public Member Functions

- int getDriverVersion () const

  CUDA driver version.
- int getRuntimeVersion () const

  CUDA runtime version.
- int getDeviceCount () const

  CUDA device count.
- bool getIsFermi (std::size_t device) const

  Device is a Fermi-based GPU.
- bool getIsKepler (std::size_t device) const

  Device is a Kepler-based GPU.
- bool getIsMaxwell (std::size_t device) const

  Device is a Maxwell-based GPU.
- bool getIsPascal (std::size_t device) const

  Device is a Pascal-based GPU.
- bool getIsVolta (std::size_t device) const

  Device is a Volta-based GPU.
- bool getIsTouring (std::size_t device) const

  Device is a Touring-based GPU.
- bool getIsAtLeastFermi (std::size_t device) const

  Device is at least a Fermi-based GPU.
- bool getIsAtLeastKepler (std::size_t device) const

  Device is at least a Kepler-based GPU.
- bool getIsAtLeastMaxwell (std::size_t device) const

  Device is at least a Maxwell-based GPU.
- bool getIsAtLeastPascal (std::size_t device) const

  Device is at least a Pascal-based GPU.
- bool getIsAtLeastVolta (std::size_t device) const

  Device is at least a Volta-based GPU.
- bool getIsAtLeastTouring (std::size_t device) const

  Device is at least a Touring-based GPU.
- bool getHasDynamicParallelism (std::size_t device) const

  Device support for Dynamic Parallelism.
- bool getHasUnifiedMemory (std::size_t device) const

  Device support for Unified Memory.
- std::string getName (std::size_t device) const

  ASCII string identifying device.
- std::size_t getTotalGlobalMemory (std::size_t device) const

  Global memory available on device in bytes.
- std::size_t getSharedMemoryPerBlock (std::size_t device) const

  Shared memory available per block in bytes.
- int getRegistersPerBlock (std::size_t device) const

  32-bit registers available per block
- int getWarpSize (std::size_t device) const

  Warp size in threads.
- std::size_t getMemoryPitch (std::size_t device) const

  Maximum pitch in bytes allowed by memory copies.
- int getMaxThreadsPerBlock (std::size_t device) const

  Maximum number of threads per block.
- const int ∗ getMaxThreadsDimension (std::size_t device) const

Maximum size of each dimension of a block.
- const int ∗ getMaxGridSize (std::size_t device) const

  Maximum size of each dimension of a grid.
- int getClockRate (std::size_t device) const

  Clock frequency in kilohertz.
- std::size_t getTotalConstMemory (std::size_t device) const

  Constant memory available on device in bytes.
- int getMajorVersion (std::size_t device) const

  Major compute capability.
- int getMinorVersion (std::size_t device) const

  Minor compute capability.
- std::size_t getTextureAlignment (std::size_t device) const

  Alignment requirement for textures.
- std::size_t getTexturePitchAlignment (std::size_t device) const

  Pitch alignment requirement for texture references bound to pitched memory.
- int getDeviceOverlap (std::size_t device) const

  Device can concurrently copy memory and execute a kernel. Deprecated. Use instead asyncEngineCount.
- int getMultiProcessorCount (std::size_t device) const

  Number of multiprocessors on device.
- int getKernelExecTimeoutEnabled (std::size_t device) const

  Specified whether there is a run time limit on kernels.
- int getIntegrated (std::size_t device) const

  Device is integrated as opposed to discrete.
- int getCanMapHostMemory (std::size_t device) const

  Device can map host memory with cudaHostAlloc/cudaHostGetDevicePointer.
- int getComputeMode (std::size_t device) const

  Compute mode (See ::cudaComputeMode)
- int getMaxTexture1D (std::size_t device) const

  Maximum 1D texture size.
- int getMaxTexture1DMipmap (std::size_t device) const

  Maximum 1D mipmapped texture size.
- int getMaxTexture1DLinear (std::size_t device) const

  Maximum size for 1D textures bound to linear memory.
- const int ∗ getMaxTexture2D (std::size_t device) const

  Maximum 2D texture dimensions.
- const int ∗ getMaxTexture2DMipmap (std::size_t device) const

  Maximum 2D mipmapped texture dimensions.
- const int ∗ getMaxTexture2DLinear (std::size_t device) const

  Maximum dimensions (width, height, pitch) for 2D textures bound to pitched memory.
- const int ∗ getMaxTexture2DGather (std::size_t device) const

  Maximum 2D texture dimensions if texture gather operations have to be performed.
- const int ∗ getMaxTexture3D (std::size_t device) const

  Maximum 3D texture dimensions.
- const int ∗ getMaxTexture3DAlt (std::size_t device) const

  Maximum alternate 3D texture dimensions.
- int getMaxTextureCubemap (std::size_t device) const

  Maximum Cubemap texture dimensions.
- const int ∗ getMaxTexture1DLayered (std::size_t device) const

  Maximum 1D layered texture dimensions.
- const int ∗ getMaxTexture2DLayered (std::size_t device) const

  Maximum 2D layered texture dimensions.

- const int ∗ getMaxTextureCubemapLayered (std::size_t device) const

    Maximum Cubemap layered texture dimensions.
- int getMaxSurface1D (std::size_t device) const

    Maximum 1D surface size.
- const int ∗ getMaxSurface2D (std::size_t device) const

    Maximum 2D surface dimensions.
- const int ∗ getMaxSurface3D (std::size_t device) const

    Maximum 3D surface dimensions.
- const int ∗ getMaxSurface1DLayered (std::size_t device) const

    Maximum 1D layered surface dimensions.
- const int ∗ getMaxSurface2DLayered (std::size_t device) const

    Maximum 2D layered surface dimensions.
- int getMaxSurfaceCubemap (std::size_t device) const

    Maximum Cubemap surface dimensions.
- const int ∗ getMaxSurfaceCubemapLayered (std::size_t device) const

    Maximum Cubemap layered surface dimensions.
- std::size_t getSurfaceAlignment (std::size_t device) const

    Alignment requirements for surfaces.
- int getConcurrentKernels (std::size_t device) const

    Device can possibly execute multiple kernels concurrently.
- int getECCEnabled (std::size_t device) const

    Device has ECC support enabled.
- int getPciBusID (std::size_t device) const

    PCI bus ID of the device.
- int getPciDeviceID (std::size_t device) const

    PCI device ID of the device.
- int getPciDomainID (std::size_t device) const

    PCI domain ID of the device.
- int getTccDriver (std::size_t device) const

    1 if device is a Tesla device using TCC driver, 0 otherwise
- int getAsyncEngineCount (std::size_t device) const

    Number of asynchronous engines.
- int getUnifiedAddressing (std::size_t device) const

    Device shares a unified address space with the host.
- int getMemoryClockRate (std::size_t device) const

    Peak memory clock frequency in kilohertz.
- int getMemoryBusWidth (std::size_t device) const

    Global memory bus width in bits.
- int getL2CacheSize (std::size_t device) const

    Size of L2 cache in bytes.
- int getMaxThreadsPerMultiProcessor (std::size_t device) const

    Maximum resident threads per multiprocessor.
- int getStreamPrioritiesSupported (std::size_t device) const

    Device supports stream priorities.
- int getGlobalL1CacheSupported (std::size_t device) const

    Device supports caching globals in L1.
- int getLocalL1CacheSupported (std::size_t device) const

    Device supports caching locals in L1.
- std::size_t getSharedMemoryPerMultiprocessor (std::size_t device) const

    Shared memory available per multiprocessor in bytes.
- int getRegistersPerMultiprocessor (std::size_t device) const

  32-bit registers available per multiprocessor
- int getManagedMemory (std::size_t device) const

  Device supports allocating managed memory on this system.
- int getIsMultiGpuBoard (std::size_t device) const

  Device is on a multi-GPU board.
- int getMultiGpuBoardGroupID (std::size_t device) const

  Unique identifier for a group of devices on the same multi-GPU board.
- int getCUDADeviceCount () const

  Device count.
- **CUDADriverInfo** (const CUDADriverInfo &)=delete
- **CUDADriverInfo** (CUDADriverInfo &&)=delete
- CUDADriverInfo & **operator=** (const CUDADriverInfo &)=delete
- CUDADriverInfo & **operator=** (CUDADriverInfo &&)=delete

Private Member Functions

- void **reportCUDAPlatformVersions** () const
- void **reportCUDADeviceCapabilities** (std::size_t device) const

Private Attributes

- int **_cudaDriverVersion** = 0
- int **_cudaRuntimeVersion** = 0
- int **_cudaDeviceCount** = 0
- cudaDeviceProp ∗ **_allCudaDevicesProperties** = nullptr

5.13.1 Detailed Description

This class encapsulates CUDA driver info for detection & reporting.

CUDADriverInfo.h:

This class encapsulates CUDA driver info for detection & reporting.

Author

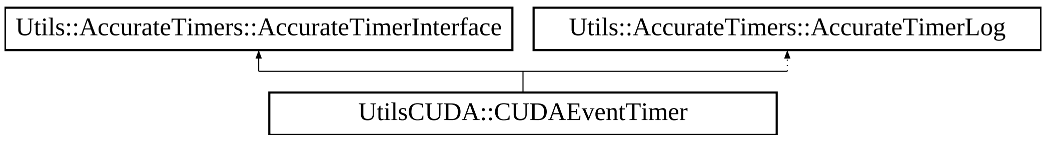
Thanos Theo, 2018

5.14 UtilsCUDA::CUDAEventTimer Class Reference

This class contains an AccurateTimers encapsulation of CUDA event timers.

```
#include <CUDAEventTimer.h>
```

Inheritance diagram for UtilsCUDA::CUDAEventTimer:


Utils::AccurateTimers::AccurateTimerInterface
Utils::AccurateTimers::AccurateTimerLog
UtilsCUDA::CUDAEventTimer

Public Member Functions

- void **startTimer** () override
- void **stopTimer** () override
- double **getElapsedTimeInNanoSecs** () override
- double **getElapsedTimeInMicroSecs** () override
- double **getElapsedTimeInMilliSecs** () override
- double **getElapsedTimeInSecs** () override
- double **getMeanTimeInNanoSecs** () override
- double **getMeanTimeInMicroSecs** () override
- double **getMeanTimeInMilliSecs** () override
- double **getMeanTimeInSecs** () override
- double **getDecimalElapsedTimeInMicroSecs** () override
- double **getDecimalElapsedTimeInMilliSecs** () override
- double **getDecimalElapsedTimeInSecs** () override
- double **getDecimalMeanTimeInMicroSecs** () override
- double **getDecimalMeanTimeInMilliSecs** () override
- double **getDecimalMeanTimeInSecs** () override
- **CUDAEventTimer** (const CUDAEventTimer &)=delete
- **CUDAEventTimer** (CUDAEventTimer &&)=delete
- CUDAEventTimer & **operator=** (const CUDAEventTimer &)=delete
- CUDAEventTimer & **operator=** (CUDAEventTimer &&)=delete

Private Member Functions

- float **getElapsedTime** ()

Private Attributes

- cudaEvent_t **_start** {}
- cudaEvent_t **_stop** {}

Additional Inherited Members

5.14.1 Detailed Description

This class contains an AccurateTimers encapsulation of CUDA event timers.

CUDAEventTimer.h:

This class contains an AccurateTimers encapsulation of CUDA event timers. CUDA Events provides a timer with a resolution of around 0.5 microseconds. Note: no virtual destructor is needed for data-oriented design ie no up-casting should ever be used.

Author

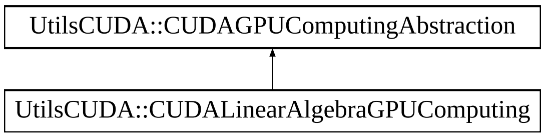
Thanos Theo, 2018

5.15 UtilsCUDA::CUDAGPUComputingAbstraction Class Reference

This class encapsulates a basic abstraction layer for CUDA GPU Computing.

```
#include <CUDAGPUComputingAbstraction.h>
```

Inheritance diagram for UtilsCUDA::CUDAGPUComputingAbstraction:


UtilsCUDA::CUDAGPUComputingAbstraction
UtilsCUDA::CUDALinearAlgebraGPUComputing

Public Member Functions

- virtual void initializeGPUMemory ()=0

  Initializes GPU memory.
- virtual void performGPUComputing ()=0

  Performs the GPU Computing calculations.
- virtual void retrieveGPUResults ()=0

  Retrieves the results from the GPU.
- virtual bool verifyComputingResults () const =0

  Verifies the computing results between the CPU and the GPU.
- virtual void releaseGPUComputingResources ()=0

  Releases the GPU Computing resources.

Protected Member Functions

- **CUDAGPUComputingAbstraction** (const CUDADriverInfo &cudaDriverInfo, int device) noexcept
- **CUDAGPUComputingAbstraction** (const CUDAGPUComputingAbstraction &)=delete
- **CUDAGPUComputingAbstraction** (CUDAGPUComputingAbstraction &&)=delete
- CUDAGPUComputingAbstraction & **operator=** (const CUDAGPUComputingAbstraction &)=delete
- CUDAGPUComputingAbstraction & **operator=** (CUDAGPUComputingAbstraction &&)=delete

Protected Attributes

- const CUDADriverInfo & **_cudaDriverInfo**
- int **_device** = 0
- int **_deviceCount** = 0
- double **_totalTimeTakenInMs** = 0.0

5.15.1 Detailed Description

This class encapsulates a basic abstraction layer for CUDA GPU Computing.

CUDAGPUComputingAbstraction.h:

This class encapsulates a basic abstraction layer for CUDA GPU Computing (abstract class CUDAGPU↩ ComputingAbstraction, ie no direct instantiation allowed). Note: no virtual destructor is needed for data-oriented design ie no up-casting should ever be used.

Author

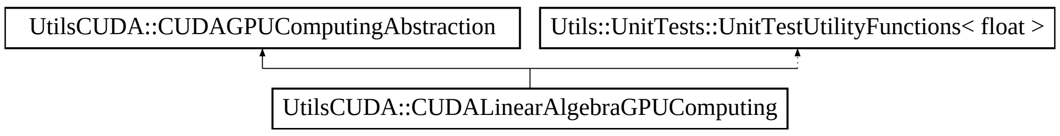
Thanos Theo, 2018

5.16 UtilsCUDA::CUDALinearAlgebraGPUComputing Class Reference

This class contains a basic Linear Algebra GPU Computing test case in CUDA.

```
#include <CUDALinearAlgebraGPUComputing.h>
```

Inheritance diagram for UtilsCUDA::CUDALinearAlgebraGPUComputing:


UtilsCUDA::CUDAGPUComputingAbstraction
Utils::UnitTests::UnitTestUtilityFunctions< float >
UtilsCUDA::CUDALinearAlgebraGPUComputing

Public Member Functions

- void initializeGPUMemory () override

  Initializes GPU memory.
- void performGPUComputing () override

  Performs the GPU Computing calculations.
- void retrieveGPUResults () override

  Retrieves the results from the GPU.
- bool verifyComputingResults () const override

  Verifies the computing results between the CPU and the GPU.
- void releaseGPUComputingResources () override

  Releases the GPU Computing resources.
- **CUDALinearAlgebraGPUComputing** (const CUDADriverInfo &cudaDriverInfo, int device=0, bool use↩ UnifiedMemoryIfAvailable=false, std::uint32_t arraySize=16384) noexcept
- **CUDALinearAlgebraGPUComputing** (const CUDALinearAlgebraGPUComputing &)=delete
- **CUDALinearAlgebraGPUComputing** (CUDALinearAlgebraGPUComputing &&)=delete
- CUDALinearAlgebraGPUComputing & **operator=** (const CUDALinearAlgebraGPUComputing &)=delete
- CUDALinearAlgebraGPUComputing & **operator=** (CUDALinearAlgebraGPUComputing &&)=delete

Private Attributes

- std::uint32_t **_arraySizeXY** = 16384 $*$ 16384
- bool **_useUnifiedMemoryIfAvailable** = false
- CUDAMemoryRegistry **_cudaMemoryRegistry**
- const CUDAStreamsHandler **_cudaStreamsHandler**
- std::unique_ptr< int32_t[ ]> **_hostArrayA** = nullptr
- std::unique_ptr< int32_t[ ]> **_hostArrayB** = nullptr
- std::unique_ptr< int32_t[ ]> **_hostArrayC** = nullptr

Additional Inherited Members

5.16.1 Detailed Description

This class contains a basic Linear Algebra GPU Computing test case in CUDA.

CUDALinearAlgebraGPUComputing.h:

This class contains a basic Linear Algebra GPU Computing test case in CUDA.

Author

Thanos Theo, 2018

5.17 UtilsCUDA::CUDAMemoryRegistry Class Reference

This class encapsulates CUDA memory registry functionality for both host & device with reporting.

```
#include <CUDAMemoryRegistry.h>
```

Public Types

- enum MemoryRegistryTypes : std::size_t { **HOST_MEMORY** = 0, **CUDA_MEMORY** = 1 }

  enum for Memory Registry Types

Public Member Functions

- void registerHostMemoryRegistry (unsigned int flags)

  Registers host memory in the Host Memory Registry (thread-safe function)
- void allocateCUDAMemoryRegistry (bool useUnifiedMemory=false)

  Allocates GPU-side memory in the CUDA Memory Registry (thread-safe function)
- void unregisterAndClearHostMemoryRegistry ()

  Unregisters host memory from the Host Memory Registry (thread-safe function)
- void freeAndClearCUDAMemoryRegistry ()

  Frees (de-allocates) GPU-side memory & clears the CUDA Memory Registry (thread-safe function)
- void clearHostMemoryRegistry ()

  Clears the Host Memory Registry (thread-safe function)
- void clearCUDAMemoryRegistry ()

  Clears the CUDA Memory Registry (thread-safe function)
- bool unregisterAndEraseFromHostMemoryRegistry (const std::string &name)

  Unregisters host memory & erases a given name from the Host Memory Registry (thread-safe function)
- bool freeAndEraseFromCUDAMemoryRegistry (const std::string &name)

  Frees (de-allocates) GPU-side memory & erases a given name from the CUDA Memory Registry (thread-safe function)
- bool eraseFromHostMemoryRegistry (const std::string &name)

  Erases a given name from the Host Memory Registry (thread-safe function)
- bool eraseFromCUDAMemoryRegistry (const std::string &name)

Erases a given name from the CUDA Memory Registry (thread-safe function)

- template<typename T >
  bool addToHostMemoryRegistry (const std::string &name, T ∗ptr, std::size_t size)

  Adds to the Host Memory Registry (wrapping a thread-safe non-template function) a T∗ ptr-based tuple.
- template<typename T >
  bool addToCUDAMemoryRegistry (const std::string &name, T ∗ptr, std::size_t size, int device)

  Adds to the CUDA Memory Registry (wrapping a thread-safe non-template function) a T∗ ptr-based tuple.
- template<typename T >
  bool addToCUDAMemoryRegistry (const std::string &name, std::size_t size, int device)

  Adds to the CUDA Memory Registry (wrapping a thread-safe non-template function) only by name a void∗ ptr-based tuple.
- template<typename T >
  std::tuple< T ∗, std::size_t, std::size_t, int > getPtrTupleFromHostMemoryRegistry (const std::string &name) const

  Gets from the Host Memory Registry (wrapping a thread-safe non-template function) a T∗ ptr-based tuple.
- template<typename T >
  std::tuple< T ∗, std::size_t, std::size_t, int > getPtrTupleFromCUDAMemoryRegistry (const std::string &name) const

  Gets from the CUDA Memory Registry (wrapping a thread-safe non-template function) a T∗ ptr-based tuple.
- template<typename T >
  T ∗ getPtrFromHostMemoryRegistry (const std::string &name) const

  Gets from the Host Memory Registry (wrapping a thread-safe non-template function) a T∗ ptr.
- template<typename T >
  T ∗ getPtrFromCUDAMemoryRegistry (const std::string &name) const

  Gets from the CUDA Memory Registry (wrapping a thread-safe non-template function) a T∗ ptr.
- std::size_t getHostMemoryRegistrySize () const

  Gets the Host Memory Registry size (thread-safe function)
- std::size_t getCUDAMemoryRegistrySize () const

  Gets the CUDA Memory Registry size (thread-safe function)
- std::vector< std::string > getHostMemoryNamesRegistry () const

  Gets the Host Memory Registry names (thread-safe function)
- std::vector< std::string > getCUDAMemoryNamesRegistry () const

  Gets the CUDA Memory Registry names (thread-safe function)
- void reportHostMemoryRegistryInformation () const

  Reports information from the Host Memory Registry.
- void reportCUDAMemoryRegistryInformation () const

  Reports information from the CUDA Memory Registry.
- **CUDAMemoryRegistry** (const CUDAMemoryRegistry &)=delete
- **CUDAMemoryRegistry** (CUDAMemoryRegistry &&)=delete
- CUDAMemoryRegistry & **operator=** (const CUDAMemoryRegistry &)=delete
- CUDAMemoryRegistry & **operator=** (CUDAMemoryRegistry &&)=delete

Private Member Functions

- bool addToMemoryRegistryVoidPtr (const std::string &name, void ∗ptr, std::size_t size, std::size_t sizeOf↩
  Object, int device, MemoryRegistryTypes type)

  Adds to the CUDA Memory Registry (thread-safe function) a void∗ ptr-based tuple.
- std::tuple< void ∗, std::size_t, std::size_t, int > getFromMemoryRegistryVoidPtr (const std::string &name, MemoryRegistryTypes type) const

  Gets from the CUDA Memory Registry (thread-safe function) a void∗ ptr-based tuple.

Private Attributes

- std::vector< std::string > _hostMemoryNamesRegistry
  The Host Memory Names Registry is stored in a vector.
- std::unordered_map< std::string, std::tuple< void ∗, std::size_t, std::size_t, int > > _hostMemoryRegistry
  The Host Memory Registry is stored in an unordered map.
- std::vector< std::string > _cudaMemoryNamesRegistry
  The CUDA Memory Names Registry is stored in a vector.
- std::unordered_map< std::string, std::tuple< void ∗, std::size_t, std::size_t, int > > _cudaMemoryRegistry
  The CUDA Memory Registry is stored in an unordered map.

5.17.1 Detailed Description

This class encapsulates CUDA memory registry functionality for both host & device with reporting.

CUDAMemoryRegistry.h:

This class encapsulates CUDA memory registry functionality for both host & device with reporting.

Author

Thanos Theo, 2018

5.18 UtilsCUDA::CUDASpinLock Class Reference

This class is based on the book'The CUDA Handbook - A comprehensive Guide to GPU Programming'.

```
#include <CUDASpinLock.h>
```

Public Member Functions

- __device__ __forceinline__ void **acquire** ()
- __device__ __forceinline__ void **release** ()
- __device__ __forceinline__ **CUDASpinLock** (int ∗address) noexcept
- **CUDASpinLock** (const CUDASpinLock &)=delete
- **CUDASpinLock** (CUDASpinLock &&)=delete
- CUDASpinLock & **operator=** (const CUDASpinLock &)=delete
- CUDASpinLock & **operator=** (CUDASpinLock &&)=delete

Private Attributes

- int ∗ **_address** = nullptr

5.18.1 Detailed Description

This class is based on the book'The CUDA Handbook - A comprehensive Guide to GPU Programming'.

CUDASpinLock.h:

Note from the book: The CUDA execution model imposes restrictions on the use of global memory atomics for synchronization, like for this CUDASpinLock class. Unlike CPU threads, some CUDAthreads within a kernel launch may not begin execution until other threads in the same kernel have exited. On CUDA hardware, each SM can context switch a limited number of thread blocks, so any kernel launch with more than MaxThreadBlocksPerSM ∗ NumSMs requires the first thread blocks to exit before more thread blocks can begin execution. As a result, it is important that developers not assume all of the threads in a given kernel launch are active.

Additionally, the CUDASpinLock::acquire() function below is prone to deadlock if used for INTRABLOCK synchronization. Expected usage is for one thread in each block to attempt to acquire the CUDASpinLock, otherwise the divergent code execution tends to deadlock. This is unsuitable in any case, since the hardware supports so many better ways for threads within the same block to communicate and synchronize with one another, for example shared memory and __syncthreads(), respectively.

Example code usage: **device forceinline** void sumDoubles(double∗ pSum, int∗ spinlock, const double∗ in, size_t N, int∗ acquireCount) { SharedMemory<double> shared; CUDASpinLock globalSpinlock(spinlock); for (size_↩
t i = blockIdx.x∗blockDim.x + threadIdx.x; i < N; i += blockDim.x∗gridDim.x) { shared[threadIdx.x] = in[i]; __↩
syncthreads(); double blockSum = Reduce_block<double, double>(); __syncthreads();

if (threadIdx.x == 0) { globalSpinlock.acquire(); ∗pSum += blockSum; __threadfence(); // function stalls current thread until its writes to global memory are guaranteed to be visible by all other threads in the grid globalSpinlock.↩
release(); } } }

Author

Thanos Theo, 2018

Version

14.0.0.0

5.19 UtilsCUDA::CUDAStreamsHandler Class Reference

This class encapsulates usage of a collection of CUDA streams & the RAII C++ idiom.

```
#include <CUDAStreamsHandler.h>
```

Public Member Functions

- **CUDAStreamsHandler** (const CUDADriverInfo &cudaDriverInfo, int device=0, size_t numberOfStreams=1, bool useStreamPriorities=true, int priorityType=cudaStreamNonBlocking) noexcept
- const cudaStream_t & **operator[ ]** (std::size_t index) const noexcept
- **CUDAStreamsHandler** (const CUDAStreamsHandler &)=delete
- **CUDAStreamsHandler** (CUDAStreamsHandler &&)=delete
- CUDAStreamsHandler & **operator=** (const CUDAStreamsHandler &)=delete
- CUDAStreamsHandler & **operator=** (CUDAStreamsHandler &&)=delete

Private Member Functions

- void **initialize** () noexcept
- void **uninitialize** () const noexcept

Private Attributes

- std::unique_ptr< cudaStream_t[ ]> **cudaStreams** = nullptr
- std::unique_ptr< bool[ ]> **cudaStreamsInitialized** = nullptr
- std::size_t **numberOfStreams** = 0
- bool **useStreamPriorities** = false
- int **priorityType** = cudaStreamNonBlocking
- int **priorityHigh** = 0
- int **priorityLow** = 0

5.19.1 Detailed Description

This class encapsulates usage of a collection of CUDA streams & the RAII C++ idiom.

Author

Thanos Theo, 2018

Version

14.0.0.0

5.20 UtilsCUDA::CUDAUtilityFunctions Struct Reference

This class encapsulates all the CUDA related utility functions.

```
#include <CUDAUtilityFunctions.h>
```

Public Member Functions

- **CUDAUtilityFunctions** (const CUDAUtilityFunctions &)=delete
- **CUDAUtilityFunctions** (CUDAUtilityFunctions &&)=delete
- CUDAUtilityFunctions & **operator=** (const CUDAUtilityFunctions &)=delete
- CUDAUtilityFunctions & **operator=** (CUDAUtilityFunctions &&)=delete

Static Public Member Functions

- template<typename... Args>
  static void __host__ __device__ **printfCUDAImpl** (const char ∗format, Args... args)
- static bool __host__ __device__ equal (float left, float right)
  GLSL-style equal function (float version).
- static bool __host__ __device__ equal (double left, double right)
  GLSL-style equal function (double version).
- static float __host__ __device__ **sign** (float x)
- static double __host__ __device__ **sign** (double x)
- static float __host__ __device__ fractf (float x)
  GLSL-style fract function (float version).
- static double __host__ __device__ fract (double x)

GLSL-style fract function (double version).
- static float __host__ __device__ toRadians (float degrees)

  Conversion function from degrees to radians (float version).
- static double __host__ __device__ toRadians (double degrees)

  Conversion function from degrees to radians (double version).
- static float __host__ __device__ toDegrees (float radians)

  Conversion function from radians to degrees (float version).
- static double __host__ __device__ toDegrees (double radians)

  Conversion function from radians to degrees (double version).
- static float __host__ __device__ dot (const float2 &a, const float2 &b)

  GLSL-style dot function (float version).
- static double __host__ __device__ dot (const double2 &a, const double2 &b)

  GLSL-style dot function (double version).
- static float __host__ __device__ rand1 (const float2 &seed)

  This function returns uniformly distributed float values in the range [0, 1] (float version).
- static double __host__ __device__ rand1 (const double2 &seed)

  This function returns uniformly distributed double values in the range [0, 1] (double version).
- static float2 __host__ __device__ rand2 (const float2 &seed)

  This function returns uniformly distributed float2 values in the range [0, 1] (float version).
- static double2 __host__ __device__ rand2 (const double2 &seed)

  This function returns uniformly distributed double2 values in the range [0, 1] (double version).
- static float3 __host__ __device__ rand3 (const float2 &seed)

  This function returns uniformly distributed float3 values in the range [0, 1] (float version).
- static double3 __host__ __device__ rand3 (const double2 &seed)

  This function returns uniformly distributed double3 values in the range [0, 1] (double version).
- static float4 __host__ __device__ rand4 (const float2 &seed)

  This function returns uniformly distributed float4 values in the range [0, 1] (float version).
- static double4 __host__ __device__ rand4 (const double2 &seed)

  This function returns uniformly distributed double4 values in the range [0, 1] (double version).
- template<std::uint32_t N>
  static std::uint32_t __host__ __device__ seedGenerator (std::uint32_t value0, std::uint32_t value1)

  Seed generator for the Linear Congruential Generator (LGC).
- static std::uint32_t __host__ __device__ rand1u (std::uint32_t &seed)

  Generate random uint32_t values in the [0, 2^24) range with the Linear Congruential Generator (LGC).
- static float __host__ __device__ rand1f (std::uint32_t &seed)

  Generate random float values in the [0, 1) range with the Linear Congruential Generator (LGC).
- static float2 __host__ __device__ rand2f (std::uint32_t &seed)

  Generate random float2 values in the [0, 1) range with the Linear Congruential Generator (LGC).
- static float3 __host__ __device__ rand3f (std::uint32_t &seed)

  Generate random float3 values in the [0, 1) range with the Linear Congruential Generator (LGC).
- static float4 __host__ __device__ rand4f (std::uint32_t &seed)

  Generate random float4 values in the [0, 1) range with the Linear Congruential Generator (LGC).
- template<typename T >
  static bool __host__ __device__ checkAbsoluteError (T a, T b, T epsilon)

  This function is the GPU version checkAbsoluteError to be used on CUDA.
- static std::uint32_t __host__ __device__ asUint32 (float value)

  Get the float32 bit representation to a uint32.
- static float __host__ __device__ asFloat32 (std::uint32_t value)

  Get the uint32 bit representation to a float32.
- static std::uint32_t __host__ __device__ float32Flip (float unflippedFloatValue)

Flip a float32 for make it sortable: finds SIGN of fp number, so: if it's 1 (negative float32) it flips all bits, if it's 0 (positive float32) it flips the sign only.

- static float __host__ __device__ float32Unflip (std::uint32_t flippedFloatValue)

  Unflip a float32 back (invert float32Flip() above): signed was flipped from above, so: if sign is 1 (negative) it flips the sign bit back, if if sign is 0 (positive) it flips all bits back.
- static std::uint64_t __host__ __device__ asUint64 (double value)

  Get the float64 bit representation to a uint64.
- static double __host__ __device__ asFloat64 (std::uint64_t value)

  Get the uint64 bit representation to a float64.
- static std::uint64_t __host__ __device__ float64Flip (double unflippedFloatValue)

  Flip a float64 for make it sortable: finds SIGN of fp number, so: if it's 1 (negative float64) it flips all bits, if it's 0 (positive float64) it flips the sign only.
- static double __host__ __device__ float64Unflip (std::uint64_t flippedFloatValue)

  Unflip a float64 back (invert float64Flip() above): signed was flipped from above, so: if sign is 1 (negative) it flips the sign bit back, if if sign is 0 (positive) it flips all bits back.
- static void **checkCUDAErrorImpl** (const cudaError_t &errnum, const char ∗file, const char ∗function, int line, bool abort=true)
- static std::uint32_t **getWarpSize** ()
- static dim3 **getDefaultTheads2DDimensions** ()
- static std::uint32_t powerOfTwoDimension2D (std::uint32_t arraySize)

  The powerOfTwoDimension2D() function finds the next power-of-two dimension for a 2D kernel given a 1D array size.
- static std::tuple< dim3, dim3, std::uint32_t > calculateCUDA2DKernelDimensions (std::uint32_t arraySize, const dim3 &threads2D=getDefaultTheads2DDimensions())

  The calculateCUDA2DKernelDimensions() function efficiently calculates the dimensions for a CUDA 2D kernel.

5.20.1 Detailed Description

This class encapsulates all the CUDA related utility functions.

Author

Thanos Theo, 2018

5.20.2 Member Function Documentation

5.20.2.1 asFloat32()

```
static float __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::asFloat32 (
             std::uint32_t value )  [inline], [static]
```

Get the uint32 bit representation to a float32.

Author

Thanos Theo, 2018

5.20.2.2 asFloat64()

```
static double __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::asFloat64 (
             std::uint64_t value )  [inline], [static]
```

Get the uint64 bit representation to a float64.

Author

Thanos Theo, 2018

5.20.2.3 asUint32()

```
static std::uint32_t __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::asUint32 (
             float value )  [inline], [static]
```

Get the float32 bit representation to a uint32.

Author

Thanos Theo, 2018

5.20.2.4 asUint64()

```
static std::uint64_t __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::asUint64 (
             double value )  [inline], [static]
```

Get the float64 bit representation to a uint64.

Author

Thanos Theo, 2018

5.20.2.5 calculateCUDA2DKernelDimensions()

```
tuple< dim3, dim3, uint32_t > CUDAUtilityFunctions::calculateCUDA2DKernelDimensions (
             std::uint32_t arraySize,
             const dim3 & threads2D = getDefaultTheads2DDimensions() )  [static]
```

The calculateCUDA2DKernelDimensions() function efficiently calculates the dimensions for a CUDA 2D kernel.

Author

Thanos Theo, 2018

Version

14.0.0.0

5.20.2.6 checkAbsoluteError()

```
template<typename T >
static bool __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::checkAbsoluteError (
             T a,
             T b,
             T epsilon ) [inline], [static]
```

This function is the GPU version checkAbsoluteError to be used on CUDA.

Author

Thanos Theo, 2018

5.20.2.7 float32Flip()

```
static std::uint32_t __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::float32Flip (
             float unflippedFloatValue ) [inline], [static]
```

Flip a float32 for make it sortable: finds SIGN of fp number, so: if it's 1 (negative float32) it flips all bits, if it's 0 (positive float32) it flips the sign only.

Needs IEEE 754 hardware compliance. Based on `http://stereopsis.com/radix.html`.

Author

Thanos Theo, 2018

5.20.2.8 float32Unflip()

```
static float __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::float32Unflip (
             std::uint32_t flippedFloatValue ) [inline], [static]
```

Unflip a float32 back (invert float32Flip() above): signed was flipped from above, so: if sign is 1 (negative) it flips the sign bit back, if if sign is 0 (positive) it flips all bits back.

Needs IEEE 754 hardware compliance. Based on `http://stereopsis.com/radix.html`.

Author

Thanos Theo, 2018

5.20.2.9 float64Flip()

```
static std::uint64_t __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::float64Flip (
             double unflippedFloatValue ) [inline], [static]
```

Flip a float64 for make it sortable: finds SIGN of fp number, so: if it's 1 (negative float64) it flips all bits, if it's 0 (positive float64) it flips the sign only.

Needs IEEE 754 hardware compliance. Based on `http://stereopsis.com/radix.html`.

Author

Thanos Theo, 2018

5.20.2.10 float64Unflip()

```
static double __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::float64Unflip (
             std::uint64_t flippedFloatValue )  [inline], [static]
```

Unflip a float64 back (invert float64Flip() above): signed was flipped from above, so: if sign is 1 (negative) it flips the sign bit back, if if sign is 0 (positive) it flips all bits back.

Needs IEEE 754 hardware compliance. Based on `http://stereopsis.com/radix.html`.

Author

Thanos Theo, 2018

5.20.2.11 powerOfTwoDimension2D()

```
uint32_t CUDAUtilityFunctions::powerOfTwoDimension2D (
             std::uint32_t arraySize )  [static]
```

The powerOfTwoDimension2D() function finds the next power-of-two dimension for a 2D kernel given a 1D array size.

Split the 1D array size to a power-of-two 2D kernel for efficient kernel execution with large 1D array sizes:

1. Take the sqrt() and ceil() it to int.
2. Check if it is a power-of-two number.
3. If not, find the next power-of-two number.

Author

Thanos Theo, 2018

Version

14.0.0.0

5.20.2.12 rand1() [1/2]

```
static float __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::rand1 (
             const float2 & seed )  [inline], [static]
```

This function returns uniformly distributed float values in the range [0, 1] (float version).

Author

Thanos Theo, 2018

5.20.2.13 rand1() [2/2]

```
static double __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::rand1 (
            const double2 & seed )  [inline], [static]
```

This function returns uniformly distributed double values in the range [0, 1] (double version).

Author

Thanos Theo, 2018

5.20.2.14 rand1f()

```
static float __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::rand1f (
            std::uint32_t & seed )  [inline], [static]
```

Generate random float values in the [0, 1) range with the Linear Congruential Generator (LGC).

Author

Thanos Theo, 2018

5.20.2.15 rand1u()

```
static std::uint32_t __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::rand1u (
            std::uint32_t & seed )  [inline], [static]
```

Generate random uint32_t values in the [0, 2^24) range with the Linear Congruential Generator (LGC).

Author

Thanos Theo, 2018

5.20.2.16 rand2() [1/2]

```
static float2 __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::rand2 (
            const float2 & seed )  [inline], [static]
```

This function returns uniformly distributed float2 values in the range [0, 1] (float version).

Author

Thanos Theo, 2018

5.20.2.17 rand2() [2/2]

```
static double2 __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::rand2 (
             const double2 & seed )  [inline], [static]
```

This function returns uniformly distributed double2 values in the range [0, 1] (double version).

Author

Thanos Theo, 2018

5.20.2.18 rand2f()

```
static float2 __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::rand2f (
             std::uint32_t & seed )  [inline], [static]
```

Generate random float2 values in the [0, 1) range with the Linear Congruential Generator (LGC).

Author

Thanos Theo, 2018

5.20.2.19 rand3() [1/2]

```
static float3 __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::rand3 (
             const float2 & seed )  [inline], [static]
```

This function returns uniformly distributed float3 values in the range [0, 1] (float version).

Author

Thanos Theo, 2018

5.20.2.20 rand3() [2/2]

```
static double3 __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::rand3 (
             const double2 & seed )  [inline], [static]
```

This function returns uniformly distributed double3 values in the range [0, 1] (double version).

Author

Thanos Theo, 2018

5.20.2.21 rand3f()

```
static float3 __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::rand3f (
             std::uint32_t & seed )  [inline], [static]
```

Generate random float3 values in the [0, 1) range with the Linear Congruential Generator (LGC).

Author

Thanos Theo, 2018

5.20.2.22 rand4() [1/2]

```
static float4 __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::rand4 (
             const float2 & seed )  [inline], [static]
```

This function returns uniformly distributed float4 values in the range [0, 1] (float version).

Author

Thanos Theo, 2018

5.20.2.23 rand4() [2/2]

```
static double4 __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::rand4 (
             const double2 & seed )  [inline], [static]
```

This function returns uniformly distributed double4 values in the range [0, 1] (double version).

Author

Thanos Theo, 2018

5.20.2.24 rand4f()

```
static float4 __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::rand4f (
             std::uint32_t & seed )  [inline], [static]
```

Generate random float4 values in the [0, 1) range with the Linear Congruential Generator (LGC).

Author

Thanos Theo, 2018

5.20.2.25 seedGenerator()

```
template<std::uint32_t N>
static std::uint32_t __host__ __device__ UtilsCUDA::CUDAUtilityFunctions::seedGenerator (
            std::uint32_t value0,
            std::uint32_t value1 )  [inline], [static]
```

Seed generator for the Linear Congruential Generator (LGC).

Author

Thanos Theo, 2018

5.21 Utils::UtilityFunctions::DebugConsole Class Reference

The DebugConsole class provides debugging & logging functionality.

```
#include <UtilityFunctions.h>
```

Public Member Functions

- **DebugConsole** (const DebugConsole &)=delete
- **DebugConsole** (DebugConsole &&)=delete
- DebugConsole & **operator=** (const DebugConsole &)=delete
- DebugConsole & **operator=** (DebugConsole &&)=delete

Static Public Member Functions

- static void **setLogFileName** (const std::string &givenLogFileName)
- static void **setUseLogFile** (bool givenUseLogFile)
- template<typename... Args>
  static void **printfConsoleOutLineImpl** (const char ∗format, const Args... args)
- template<typename... Args>
  static void **printfFileOutLineImpl** (const char ∗format, const Args... args)
- static void **consoleOutLineImpl** ()
- template<typename... Args>
  static void **consoleOutLineImpl** (const Args... args)
- static void **fileOutLineImpl** ()
- template<typename... Args>
  static void **fileOutLineImpl** (const Args... args)
- static void **writeLogFileImpl** (const std::string &msg)

Static Private Member Functions

- static std::string **getLogFileName** ()
- static bool **getUseLogFile** ()

Static Private Attributes

- static constexpr std::size_t **STRING_BUFFER_SIZE** = 2048

5.21.1 Detailed Description

The DebugConsole class provides debugging & logging functionality.

Author

Thanos Theo, 2009-2018

Version

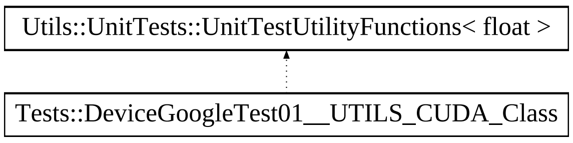
14.0.0.0

5.22 Tests::DeviceGoogleTest01__UTILS_CUDA_Class Struct Reference

Device Google Test 01 for the UtilsCUDA::CUDADriverInfo class.

```
#include <DeviceUnitTests.h>
```

Inheritance diagram for Tests::DeviceGoogleTest01__UTILS_CUDA_Class:


Utils::UnitTests::UnitTestUtilityFunctions< float >
Tests::DeviceGoogleTest01__UTILS_CUDA_Class

Public Member Functions

- **DeviceGoogleTest01__UTILS_CUDA_Class** (const DeviceGoogleTest01__UTILS_CUDA_Class &)=delete
- **DeviceGoogleTest01__UTILS_CUDA_Class** (DeviceGoogleTest01__UTILS_CUDA_Class &&)=delete
- DeviceGoogleTest01__UTILS_CUDA_Class & **operator=** (const DeviceGoogleTest01__UTILS_CUDA_↩ Class &)=delete
- DeviceGoogleTest01__UTILS_CUDA_Class & **operator=** (DeviceGoogleTest01__UTILS_CUDA_Class &&)=delete

Additional Inherited Members

5.22.1 Detailed Description

Device Google Test 01 for the UtilsCUDA::CUDADriverInfo class.

Author

Thanos Theo, 2018

Version

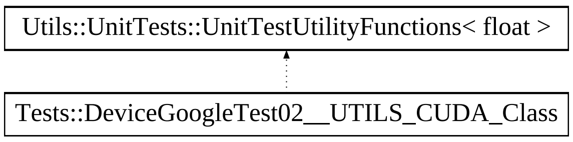
14.0.0.0

5.23 Tests::DeviceGoogleTest02__UTILS_CUDA_Class Struct Reference

Device Google Test 02 for the UtilsCUDA::CUDALinearAlgebraGPUComputing class.

```
#include <DeviceUnitTests.h>
```

Inheritance diagram for Tests::DeviceGoogleTest02__UTILS_CUDA_Class:


Utils::UnitTests::UnitTestUtilityFunctions< float >
Tests::DeviceGoogleTest02__UTILS_CUDA_Class

Public Member Functions

- **DeviceGoogleTest02__UTILS_CUDA_Class** (const DeviceGoogleTest02__UTILS_CUDA_Class &)=delete
- **DeviceGoogleTest02__UTILS_CUDA_Class** (DeviceGoogleTest02__UTILS_CUDA_Class &&)=delete
- DeviceGoogleTest02__UTILS_CUDA_Class & **operator=** (const DeviceGoogleTest02__UTILS_CUDA_↩ Class &)=delete
- DeviceGoogleTest02__UTILS_CUDA_Class & **operator=** (DeviceGoogleTest02__UTILS_CUDA_Class &&)=delete

Additional Inherited Members

5.23.1 Detailed Description

Device Google Test 02 for the UtilsCUDA::CUDALinearAlgebraGPUComputing class.

Author

Thanos Theo, 2018

Version

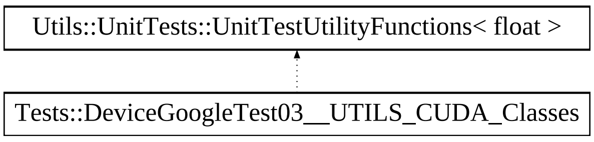
14.0.0.0

5.24 Tests::DeviceGoogleTest03__UTILS_CUDA_Classes Struct Reference

Device Google Test 03 for the UtilsCUDA::CUDADriverInfo class CUDA Memory Registry functionality.

```
#include <DeviceUnitTests.h>
```

Inheritance diagram for Tests::DeviceGoogleTest03__UTILS_CUDA_Classes:


Utils::UnitTests::UnitTestUtilityFunctions< float >
Tests::DeviceGoogleTest03__UTILS_CUDA_Classes

Public Member Functions

- **DeviceGoogleTest03__UTILS_CUDA_Classes** (const DeviceGoogleTest03__UTILS_CUDA_Classes &)=delete
- **DeviceGoogleTest03__UTILS_CUDA_Classes** (DeviceGoogleTest03__UTILS_CUDA_Classes &&)=delete
- DeviceGoogleTest03__UTILS_CUDA_Classes & **operator=** (const DeviceGoogleTest03__UTILS_CUDA_↩ Classes &)=delete
- DeviceGoogleTest03__UTILS_CUDA_Classes & **operator=** (DeviceGoogleTest03__UTILS_CUDA_Classes &&)=delete

Additional Inherited Members

5.24.1 Detailed Description

Device Google Test 03 for the UtilsCUDA::CUDADriverInfo class CUDA Memory Registry functionality.

Author

Thanos Theo, 2018

Version

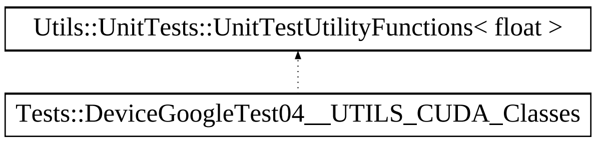
14.0.0.0

5.25 Tests::DeviceGoogleTest04__UTILS_CUDA_Classes Struct Reference

Device Google Test 04 for the UtilsCUDA::CUDAMemoryHandler set of classes functionality.

```
#include <DeviceUnitTests.h>
```

Inheritance diagram for Tests::DeviceGoogleTest04__UTILS_CUDA_Classes:


Utils::UnitTests::UnitTestUtilityFunctions< float >
Tests::DeviceGoogleTest04__UTILS_CUDA_Classes

Public Member Functions

- **DeviceGoogleTest04__UTILS_CUDA_Classes** (const DeviceGoogleTest04__UTILS_CUDA_Classes &)=delete
- **DeviceGoogleTest04__UTILS_CUDA_Classes** (DeviceGoogleTest04__UTILS_CUDA_Classes &&)=delete
- DeviceGoogleTest04__UTILS_CUDA_Classes & **operator=** (const DeviceGoogleTest04__UTILS_CUDA_↩ Classes &)=delete
- DeviceGoogleTest04__UTILS_CUDA_Classes & **operator=** (DeviceGoogleTest04__UTILS_CUDA_Classes &&)=delete

Additional Inherited Members

5.25.1 Detailed Description

Device Google Test 04 for the UtilsCUDA::CUDAMemoryHandler set of classes functionality.

Author

Thanos Theo, 2018

Version

14.0.0.0

5.26 UtilsCUDA::DeviceMemory< T > Class Template Reference

This class encapsulates usage of a collection of CUDA memory handling techniques (device only) & the RAII C++ idiom.

```
#include <CUDAMemoryHandler.h>
```

Public Member Functions

- void **allocate** (std::size_t numberOfElements, int device=0, bool useUnifiedMemory=false) noexcept
- std::future< void > **allocateAsync** (std::size_t numberOfElements, int device=0, bool useUnified↩ Memory=false) noexcept
- void **memset** (std::size_t numberOfElements, int value) noexcept
- std::future< void > **memsetAsync** (std::size_t numberOfElements, int value) noexcept
- void **copyHostToDevice** (const void ∗hostPtr, std::size_t size) noexcept
- void **copyHostToDeviceAsync** (const void ∗hostPtr, std::size_t size, const cudaStream_t &stream) noexcept
- void **copyDeviceToHost** (void ∗hostPtr, std::size_t size) const noexcept
- void **copyDeviceToHostAsync** (void ∗hostPtr, std::size_t size, const cudaStream_t &stream) const noexcept
- T ∗ **device** ()
- const T ∗ **device** () const
- **DeviceMemory** (const DeviceMemory &)=delete
- **DeviceMemory** (DeviceMemory &&)=delete
- DeviceMemory & **operator=** (const DeviceMemory &)=delete
- DeviceMemory & **operator=** (DeviceMemory &&)=delete

Private Attributes

- DeviceUniquePtr< T > **devicePtr_** = nullptr

5.26.1 Detailed Description

template<typename T>
class UtilsCUDA::DeviceMemory< T >

This class encapsulates usage of a collection of CUDA memory handling techniques (device only) & the RAII C++ idiom.

Author

David Lenz, Thanos Theo, 2018

Version

14.0.0.0

5.27 Utils::VectorTypes::double2 Struct Reference

The double2 class provides double2 functionality.

```
#include <VectorTypes.h>
```

Public Member Functions

- **double2** (double x, double y) noexcept
- **double2** (const double2 &)=default
- **double2** (double2 &&other)=default
- double2 & **operator=** (const double2 &)=default
- double2 & **operator=** (double2 &&other)=default

Public Attributes

- double **x** = 0.0
- double **y** = 0.0

5.27.1 Detailed Description

The double2 class provides double2 functionality.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.28 Utils::VectorTypes::double3 Struct Reference

The double3 class provides double3 functionality.

```
#include <VectorTypes.h>
```

Public Member Functions

- **double3** (double x, double y, double z) noexcept
- **double3** (const double3 &)=default
- **double3** (double3 &&other)=default
- double3 & **operator=** (const double3 &)=default
- double3 & **operator=** (double3 &&other)=default

Public Attributes

- double **x** = 0.0
- double **y** = 0.0
- double **z** = 0.0

5.28.1 Detailed Description

The double3 class provides double3 functionality.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.29 Utils::VectorTypes::double4 Struct Reference

The double4 class provides double4 functionality.

```
#include <VectorTypes.h>
```

Public Member Functions

- **double4** (double x, double y, double z, double w) noexcept
- **double4** (const double4 &)=default
- **double4** (double4 &&other)=default
- double4 & **operator=** (const double4 &)=default
- double4 & **operator=** (double4 &&other)=default

Public Attributes

- double **x** = 0.0
- double **y** = 0.0
- double **z** = 0.0
- double **w** = 0.0

5.29.1 Detailed Description

The double4 class provides double4 functionality.

Author

Thanos Theo, 2009-2018

Version

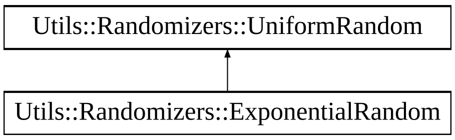
14.0.0.0

5.30 Utils::Randomizers::ExponentialRandom Class Reference

The ExponentialRandom class provides a exponential random number generator.

```
#include <Randomizers.h>
```

Inheritance diagram for Utils::Randomizers::ExponentialRandom:


Utils::Randomizers::UniformRandom
Utils::Randomizers::ExponentialRandom

Public Member Functions

- double **getExponentialFloat** ()
- double **operator()** ()
- **ExponentialRandom** (const ExponentialRandom &)=delete
- **ExponentialRandom** (ExponentialRandom &&)=delete
- ExponentialRandom & **operator=** (const ExponentialRandom &)=delete
- ExponentialRandom & **operator=** (ExponentialRandom &&)=delete

Private Attributes

- std::exponential_distribution< double > **_exponentialDistribution**

Additional Inherited Members

5.30.1 Detailed Description

The ExponentialRandom class provides a exponential random number generator.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.31 Utils::VectorTypes::float2 Struct Reference

The float2 class provides float2 functionality.

```
#include <VectorTypes.h>
```

Public Member Functions

- **float2** (float x, float y) noexcept
- **float2** (const float2 &)=default
- **float2** (float2 &&other)=default
- float2 & **operator=** (const float2 &)=default
- float2 & **operator=** (float2 &&other)=default

Public Attributes

- float **x** = 0.0f
- float **y** = 0.0f

5.31.1 Detailed Description

The float2 class provides float2 functionality.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.32 Utils::VectorTypes::float3 Struct Reference

The float3 class provides float3 functionality.

```
#include <VectorTypes.h>
```

Public Member Functions

- **float3** (float x, float y, float z) noexcept
- **float3** (const float3 &)=default
- **float3** (float3 &&other)=default
- float3 & **operator=** (const float3 &)=default
- float3 & **operator=** (float3 &&other)=default

Public Attributes

- float **x** = 0.0f
- float **y** = 0.0f
- float **z** = 0.0f

5.32.1 Detailed Description

The float3 class provides float3 functionality.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.33 Utils::VectorTypes::float4 Struct Reference

The float4 class provides float4 functionality.

```
#include <VectorTypes.h>
```

Public Member Functions

- **float4** (float x, float y, float z, float w) noexcept
- **float4** (const float4 &)=default
- **float4** (float4 &&other)=default
- float4 & **operator=** (const float4 &)=default
- float4 & **operator=** (float4 &&other)=default

Public Attributes

- float **x** = 0.0f
- float **y** = 0.0f
- float **z** = 0.0f
- float **w** = 0.0f

5.33.1 Detailed Description

The float4 class provides float4 functionality.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.34 Utils::FunctionView< Fn > Class Template Reference

This class encapsulates usage of a function view (lightweight replacement of std::function).

```
#include <FunctionView.h>
```

5.34.1 Detailed Description

template<typename Fn>
class Utils::FunctionView< Fn >

This class encapsulates usage of a function view (lightweight replacement of std::function).

FunctionView<R(T...)> is a lightweight non-owning generic callable object view, similar to a std::function<R(T...)>, but with much less overhead.

A FunctionView invocation should have the same cost as a function pointer (which it basically is underneath). The function-like object that the FunctionView refers to MUST have a lifetime that outlasts any use of the FunctionView.

In contrast, a full std::function<> is an owning container for a callable object. It's more robust, especially with respect to object lifetimes, but the call overhead is quite high. So use a FunctionView when you can.

This implementation comes from LLVM: `https://github.com/llvm-mirror/llvm/blob/master/include/llvm/` `ADT/STLExtras.h`

For more information & profiling tests: `https://vittorioromeo.info/index/blog/passing_↩` `functions_to_functions.html`

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.35 Utils::FunctionView< Ret(Params...)> Class Template Reference

Public Member Functions

- **FunctionView** (std::nullptr_t)
- template<typename Callable >
  FunctionView (Callable &&callable, std::enable_if_t<!std::is_same< std::remove_reference_t< Callable >, FunctionView >::value > ∗=nullptr)
- Ret **operator()** (Params ...params) const
- **operator bool** () const

Static Private Member Functions

- template<typename Callable >
  static Ret **callback_fn** (intptr_t callable, Params... params)

Private Attributes

- Ret(∗ **callback**)(intptr_t callable, Params... params) = nullptr
- intptr_t **callable** = 0

5.36 OpenGLRenderingEngine::OpenGLUtilityFunctions::GLAuxiliaryFunctions Struct Reference

This class contains only static CG & OpenGL related methods.

```
#include <OpenGLUtilityFunctions.h>
```

Public Member Functions

- **GLAuxiliaryFunctions** (const GLAuxiliaryFunctions &)=delete
- **GLAuxiliaryFunctions** (GLAuxiliaryFunctions &&)=delete
- GLAuxiliaryFunctions & **operator=** (const GLAuxiliaryFunctions &)=delete
- GLAuxiliaryFunctions & **operator=** (GLAuxiliaryFunctions &&)=delete

Static Public Member Functions

- template<typename T >
  static void flipPixels (std::size_t bytesPerPixel, std::size_t width, std::size_t height, T ∗__restrict pixelData)

  Flips the given pixel data to adhere to OpenGL's bottom-top coordinate system.
- static void findCurrentActiveTextureUnit (int textureValues[3])

  Finds and returns the currently active texture unit.
- static int currentTexEnvModeGLConstantToShaderEnum ()

  Returns the current texture environment GL constant to a shader enum after polling the GL state.
- static int convertTexEnvModeGLConstantToShaderEnum (int texEnvMode)

  Converts the texture environment GL constant to a shader enum.
- static int getCurrentGLState (GLenum mode)

Gets the current state of the given GL mode.
- static void setVSynch (int enableVSynch, bool isVSyncSupported)

  Set the VSync on/off state.
- static void prepareHighQualityRendering (bool isNvidia)

  Prepare GL high quality rendering (may have a minor speed-hit on older GPUs).
- static void prepareLowQualityRendering (bool isNvidia)

  Prepare GL low quality rendering (may have a minor speed-up on older GPUs).
- static void createFullScreenQuad ()

  Create a fullscreen quad for fullscreen & FBO related effects.
- static void createFullScreenQuadWithDummyVAO (bool isNvidia, GLuint dummyVao)

  Create a fullscreen quad for fullscreen & FBO related effects with a dummy VAO.
- static uint32_t packNormalToUInt (float x, float y, float z)

  Pack given XYZ normal to UInt32 type (used for the GL_INT_2_10_10_10_REV normal packing conversion).
- static void checkGLErrorImpl (const char ∗file, const char ∗function, int line, GLenum errnum=glGetError())

  A simple OpenGL error checking routine.

5.36.1 Detailed Description

This class contains only static CG & OpenGL related methods.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.36.2 Member Function Documentation

5.36.2.1 checkGLErrorImpl()

```
void GLAuxiliaryFunctions::checkGLErrorImpl (
            const char * file,
            const char * function,
            int line,
            GLenum errnum = glGetError() )  [static]
```

A simple OpenGL error checking routine.

This compiles away to a no-op inline method if the GPU_FRAMEWORK_GL_CONSOLE preprocessor symbol is not defined during compilation.

- The first parameter is a GLenum.
- The second parameter (optional) is a string that can be used to indicate the location where the error check occurs.
- The third parameter (optional) determines the destination of the error message. It defaults to cout, but could also be a file.

5.37 UtilsCUDA::HostDeviceMemory< T > Class Template Reference

This class encapsulates usage of a collection of host & CUDA memory handling techniques (host & device) & the RAII C++ idiom.

```
#include <CUDAMemoryHandler.h>
```

Public Member Functions

- void **allocate** (std::size_t numberOfElements, int device=0) noexcept
- std::future< void > **allocateAsync** (std::size_t numberOfElements, int device=0) noexcept
- void **memset** (int value, bool memsetHost=true) noexcept
- std::future< void > **memsetAsync** (int value, bool memsetHost=true) noexcept
- void **copyHostToDevice** () noexcept
- void **copyHostToDeviceAsync** (const cudaStream_t &stream) noexcept
- void **copyDeviceToHost** () const noexcept
- void **copyDeviceToHostAsync** (const cudaStream_t &stream) const noexcept
- T ∗ **device** ()
- const T ∗ **device** () const
- T ∗ **host** ()
- const T ∗ **host** () const
- std::size_t **getNumberOfElements** () const
- **HostDeviceMemory** (const HostDeviceMemory &)=delete
- **HostDeviceMemory** (HostDeviceMemory &&)=delete
- HostDeviceMemory & **operator=** (const HostDeviceMemory &)=delete
- HostDeviceMemory & **operator=** (HostDeviceMemory &&)=delete

Private Attributes

- DeviceUniquePtr< T > **devicePtr_** = nullptr
- PinnedUniquePtr< T > **hostPtr_** = nullptr
- std::size_t **numberOfElements_** = 0

5.37.1 Detailed Description

template<typename T>
class UtilsCUDA::HostDeviceMemory< T >

This class encapsulates usage of a collection of host & CUDA memory handling techniques (host & device) & the RAII C++ idiom.

Author

David Lenz, Thanos Theo, 2018

Version

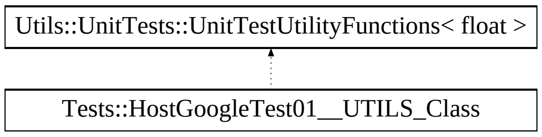
14.0.0.0

5.38 Tests::HostGoogleTest01__UTILS_Class Struct Reference

Host Google Test 01 for the Utils::AccurateTimers::AccurateCPUTimer class.

`#include <HostUnitTests.h>`

Inheritance diagram for Tests::HostGoogleTest01__UTILS_Class:


Utils::UnitTests::UnitTestUtilityFunctions< float >
Tests::HostGoogleTest01__UTILS_Class

Public Member Functions

- **HostGoogleTest01__UTILS_Class** (const HostGoogleTest01__UTILS_Class &)=delete
- **HostGoogleTest01__UTILS_Class** (HostGoogleTest01__UTILS_Class &&)=delete
- HostGoogleTest01__UTILS_Class & **operator=** (const HostGoogleTest01__UTILS_Class &)=delete
- HostGoogleTest01__UTILS_Class & **operator=** (HostGoogleTest01__UTILS_Class &&)=delete

Additional Inherited Members

5.38.1 Detailed Description

Host Google Test 01 for the Utils::AccurateTimers::AccurateCPUTimer class.

Author

Thanos Theo, 2018

Version

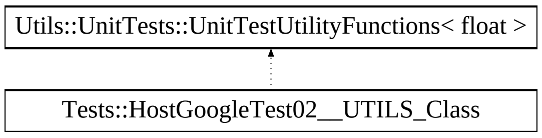
14.0.0.0

5.39 Tests::HostGoogleTest02__UTILS_Class Struct Reference

Host Google Test 02 for the Utils::Randomizers::RandomRNGWELL512 class.

`#include <HostUnitTests.h>`

Inheritance diagram for Tests::HostGoogleTest02__UTILS_Class:


Utils::UnitTests::UnitTestUtilityFunctions< float >
Tests::HostGoogleTest02__UTILS_Class

Public Member Functions

- **HostGoogleTest02__UTILS_Class** (const HostGoogleTest02__UTILS_Class &)=delete
- **HostGoogleTest02__UTILS_Class** (HostGoogleTest02__UTILS_Class &&)=delete
- HostGoogleTest02__UTILS_Class & **operator=** (const HostGoogleTest02__UTILS_Class &)=delete
- HostGoogleTest02__UTILS_Class & **operator=** (HostGoogleTest02__UTILS_Class &&)=delete

Additional Inherited Members

5.39.1 Detailed Description

Host Google Test 02 for the Utils::Randomizers::RandomRNGWELL512 class.

Author

Thanos Theo, 2018

Version

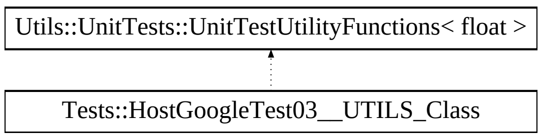
14.0.0.0

5.40 Tests::HostGoogleTest03__UTILS_Class Struct Reference

Host Google Test 03 for the Utils::SIMDVectorizations classes.

```
#include <HostUnitTests.h>
```

Inheritance diagram for Tests::HostGoogleTest03__UTILS_Class:


Utils::UnitTests::UnitTestUtilityFunctions< float >
Tests::HostGoogleTest03__UTILS_Class

Public Member Functions

- **HostGoogleTest03__UTILS_Class** (const HostGoogleTest03__UTILS_Class &)=delete
- **HostGoogleTest03__UTILS_Class** (HostGoogleTest03__UTILS_Class &&)=delete
- HostGoogleTest03__UTILS_Class & **operator=** (const HostGoogleTest03__UTILS_Class &)=delete
- HostGoogleTest03__UTILS_Class & **operator=** (HostGoogleTest03__UTILS_Class &&)=delete

Additional Inherited Members

5.40.1 Detailed Description

Host Google Test 03 for the Utils::SIMDVectorizations classes.

Author

Thanos Theo, 2018

Version

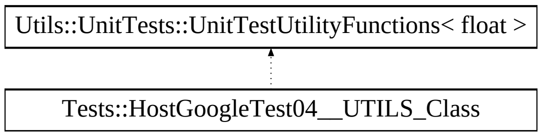
14.0.0.0

5.41 Tests::HostGoogleTest04__UTILS_Class Struct Reference

Host Google Test 04 for the Utils::UtilityFunctions::BitManipulationFunctions class.

```
#include <HostUnitTests.h>
```

Inheritance diagram for Tests::HostGoogleTest04__UTILS_Class:


Utils::UnitTests::UnitTestUtilityFunctions< float >
Tests::HostGoogleTest04__UTILS_Class

Public Member Functions

- **HostGoogleTest04__UTILS_Class** (const HostGoogleTest04__UTILS_Class &)=delete
- **HostGoogleTest04__UTILS_Class** (HostGoogleTest04__UTILS_Class &&)=delete
- HostGoogleTest04__UTILS_Class & **operator=** (const HostGoogleTest04__UTILS_Class &)=delete
- HostGoogleTest04__UTILS_Class & **operator=** (HostGoogleTest04__UTILS_Class &&)=delete

Additional Inherited Members

5.41.1 Detailed Description

Host Google Test 04 for the Utils::UtilityFunctions::BitManipulationFunctions class.

Author

Thanos Theo, 2018

Version

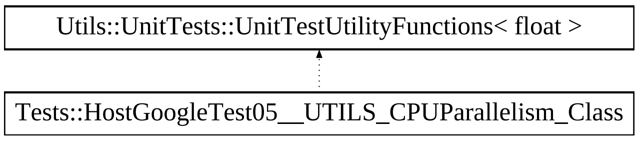
14.0.0.0

5.42 Tests::HostGoogleTest05__UTILS_CPUParallelism_Class Struct Reference

Host Google Test 05 for the Utils::CPUParallelism parallelFor() functionality.

`#include <HostUnitTests.h>`

Inheritance diagram for Tests::HostGoogleTest05__UTILS_CPUParallelism_Class:


Utils::UnitTests::UnitTestUtilityFunctions< float >
Tests::HostGoogleTest05__UTILS_CPUParallelism_Class

Public Member Functions

- **HostGoogleTest05__UTILS_CPUParallelism_Class** (const HostGoogleTest05__UTILS_CPUParallelism←_Class &)=delete
- **HostGoogleTest05__UTILS_CPUParallelism_Class** (HostGoogleTest05__UTILS_CPUParallelism_Class &&)=delete
- HostGoogleTest05__UTILS_CPUParallelism_Class & **operator=** (const HostGoogleTest05__UTILS_CPU←Parallelism_Class &)=delete
- HostGoogleTest05__UTILS_CPUParallelism_Class & **operator=** (HostGoogleTest05__UTILS_CPU←Parallelism_Class &&)=delete

Additional Inherited Members

5.42.1 Detailed Description

Host Google Test 05 for the Utils::CPUParallelism parallelFor() functionality.

Author

Thanos Theo, 2018

Version

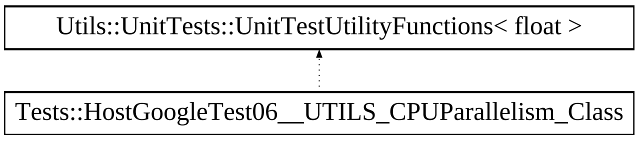
14.0.0.0

5.43 Tests::HostGoogleTest06__UTILS_CPUParallelism_Class Struct Reference

Host Google Test 06 for the Utils::CPUParallelism::CPUParallelismUnitTests class for the parallelFor() functionality.

`#include <HostUnitTests.h>`

Inheritance diagram for Tests::HostGoogleTest06__UTILS_CPUParallelism_Class:


Utils::UnitTests::UnitTestUtilityFunctions< float >
Tests::HostGoogleTest06__UTILS_CPUParallelism_Class

Public Member Functions

- **HostGoogleTest06__UTILS_CPUParallelism_Class** (const HostGoogleTest06__UTILS_CPUParallelism↩ _Class &)=delete
- **HostGoogleTest06__UTILS_CPUParallelism_Class** (HostGoogleTest06__UTILS_CPUParallelism_Class &&)=delete
- HostGoogleTest06__UTILS_CPUParallelism_Class & **operator=** (const HostGoogleTest06__UTILS_CPU↩ Parallelism_Class &)=delete
- HostGoogleTest06__UTILS_CPUParallelism_Class & **operator=** (HostGoogleTest06__UTILS_CPU↩ Parallelism_Class &&)=delete

Additional Inherited Members

5.43.1 Detailed Description

Host Google Test 06 for the Utils::CPUParallelism::CPUParallelismUnitTests class for the parallelFor() functionality.

Author

Thanos Theo, 2018

Version

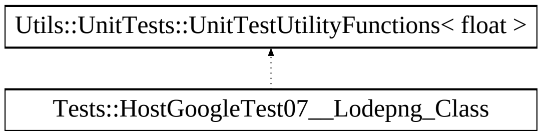
14.0.0.0

5.44 Tests::HostGoogleTest07__Lodepng_Class Struct Reference

Host Google Test 07 for the lodepng class for png encoding/decoding functionality.

```
#include <HostUnitTests.h>
```

Inheritance diagram for Tests::HostGoogleTest07__Lodepng_Class:


Utils::UnitTests::UnitTestUtilityFunctions< float >
Tests::HostGoogleTest07__Lodepng_Class

Public Member Functions

- **HostGoogleTest07__Lodepng_Class** (const HostGoogleTest07__Lodepng_Class &)=delete
- **HostGoogleTest07__Lodepng_Class** (HostGoogleTest07__Lodepng_Class &&)=delete
- HostGoogleTest07__Lodepng_Class & **operator=** (const HostGoogleTest07__Lodepng_Class &)=delete
- HostGoogleTest07__Lodepng_Class & **operator=** (HostGoogleTest07__Lodepng_Class &&)=delete

Additional Inherited Members

5.44.1 Detailed Description

Host Google Test 07 for the lodepng class for png encoding/decoding functionality.

Author

Thanos Theo, 2018

Version

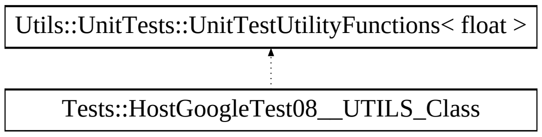
14.0.0.0

5.45 Tests::HostGoogleTest08__UTILS_Class Struct Reference

Host Google Test 08 for the Utils::UtilityFunctions::MathFunctions class.

```
#include <HostUnitTests.h>
```

Inheritance diagram for Tests::HostGoogleTest08__UTILS_Class:


Utils::UnitTests::UnitTestUtilityFunctions< float >
Tests::HostGoogleTest08__UTILS_Class

Public Member Functions

- **HostGoogleTest08__UTILS_Class** (const HostGoogleTest08__UTILS_Class &)=delete
- **HostGoogleTest08__UTILS_Class** (HostGoogleTest08__UTILS_Class &&)=delete
- HostGoogleTest08__UTILS_Class & **operator=** (const HostGoogleTest08__UTILS_Class &)=delete
- HostGoogleTest08__UTILS_Class & **operator=** (HostGoogleTest08__UTILS_Class &&)=delete

Additional Inherited Members

5.45.1 Detailed Description

Host Google Test 08 for the Utils::UtilityFunctions::MathFunctions class.

Author

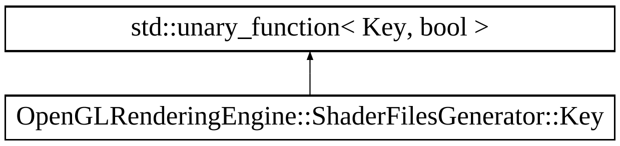
Thanos Theo, 2018

Version

14.0.0.0

5.46 OpenGLRenderingEngine::ShaderFilesGenerator::Key Class Reference

Inheritance diagram for OpenGLRenderingEngine::ShaderFilesGenerator::Key:


std::unary_function< Key, bool >
OpenGLRenderingEngine::ShaderFilesGenerator::Key

Public Member Functions

- **Key** (const std::string &first, const BitsetType &second) noexcept
- **Key** (const Key &)=default
- **Key** (Key &&)=default
- Key & **operator=** (const Key &)=default
- Key & **operator=** (Key &&)=default
- bool **operator**< (const Key &other) const
- const std::string & **getFirst** () const
- const BitsetType & **getSecond** () const

Private Attributes

- std::string **_first** = ""
- BitsetType **_second** = 0

5.47 Utils::UtilityFunctions::MathFunctions Struct Reference

The MathFunctions class provides some needed mathematical functions functionality (note that some functions emulate GLSL-style CPU functionality).

```
#include <UtilityFunctions.h>
```

Public Member Functions

- **MathFunctions** (const MathFunctions &)=delete
- **MathFunctions** (MathFunctions &&)=delete
- MathFunctions & **operator=** (const MathFunctions &)=delete
- MathFunctions & **operator=** (MathFunctions &&)=delete

Static Public Member Functions

- template<typename T >
  static bool equal (const T left, const T right, std::enable_if_t< std::is_arithmetic< T >::value > *=nullptr)

  GLSL-style equal function.
- template<typename T >
  static T **sign** (const T x, std::enable_if_t< std::is_arithmetic< T >::value &&std::is_signed< T >::value > *=nullptr)
- template<typename T >
  static T fract (const T x, std::enable_if_t< std::is_integral< T >::value > *=nullptr)

  GLSL-style fract function (integral version).
- template<typename T >
  static T fract (const T x, std::enable_if_t< std::is_floating_point< T >::value > *=nullptr)

  GLSL-style fract function (float/double version).
- template<typename T >
  static T clamp (const T &value, const T &minVal, const T &maxVal, std::enable_if_t< std::is_arithmetic< T >::value > *=nullptr)

  GLSL-style clamp function.
- template<typename T >
  static T reinterval (const T &inVal, const T &oldMin, const T &oldMax, const T &newMin, const T &newMax, std::enable_if_t< std::is_arithmetic< T >::value > *=nullptr)

  GLSL-style reinterval function.
- template<typename T >
  static T reintervalClamped (const T &inVal, const T &oldMin, const T &oldMax, const T &newMin, const T &newMax, std::enable_if_t< std::is_arithmetic< T >::value > *=nullptr)

  GLSL-style reintervalClamped function.
- template<typename T , typename I >
  static T mix (const T &left, const T &right, const I &t, std::enable_if_t< std::is_arithmetic< T >::value > *=nullptr)

  GLSL-style mix function.
- template<typename T >
  static T smoothstep (const T &edge0, const T &edge1, T x, std::enable_if_t< std::is_arithmetic< T >::value > *=nullptr)

  GLSL-style smoothstep function.
- template<typename T >
  static T smootherstep (const T &edge0, const T &edge1, T x, std::enable_if_t< std::is_arithmetic< T >::value > *=nullptr)

  Prof.
- template<typename T >
  static T toRadians (const T degrees, std::enable_if_t< std::is_floating_point< T >::value > *=nullptr)

  Conversion function from degrees to radians.
- template<typename T >
  static T toDegrees (const T radians, std::enable_if_t< std::is_floating_point< T >::value > *=nullptr)

  Conversion function from radians to degrees.
- template<typename T >
  static T matlabMOD (const T a, const T b, std::enable_if_t< std::is_integral< T >::value > *=nullptr)

  Matlab MOD function emulation (integral version).
- template<typename T >
  static T matlabMOD (const T a, const T b, std::enable_if_t< std::is_floating_point< T >::value > *=nullptr)

  Matlab MOD function emulation (float/double version).
- static float dot (const VectorTypes::float2 &a, const VectorTypes::float2 &b)

  GLSL-style dot function (float version).
- static double dot (const VectorTypes::double2 &a, const VectorTypes::double2 &b)

  GLSL-style dot function (double version).

- static float rand1 (const VectorTypes::float2 &seed)

  This function returns uniformly distributed float values in the range [0, 1] (float version).
- static double rand1 (const VectorTypes::double2 &seed)

  This function returns uniformly distributed double values in the range [0, 1] (double version).
- static VectorTypes::float2 rand2 (const VectorTypes::float2 &seed)

  This function returns uniformly distributed float2 values in the range [0, 1] (float version).
- static VectorTypes::double2 rand2 (const VectorTypes::double2 &seed)

  This function returns uniformly distributed double2 values in the range [0, 1] (double version).
- static VectorTypes::float3 rand3 (const VectorTypes::float2 &seed)

  This function returns uniformly distributed float3 values in the range [0, 1] (float version).
- static VectorTypes::double3 rand3 (const VectorTypes::double2 &seed)

  This function returns uniformly distributed double3 values in the range [0, 1] (double version).
- static VectorTypes::float4 rand4 (const VectorTypes::float2 &seed)

  This function returns uniformly distributed float4 values in the range [0, 1] (float version).
- static VectorTypes::double4 rand4 (const VectorTypes::double2 &seed)

  This function returns uniformly distributed double4 values in the range [0, 1] (double version).
- template<std::uint32_t N>
  static std::uint32_t seedGenerator (std::uint32_t value0, std::uint32_t value1)

  Seed generator for the Linear Congruential Generator (LGC).
- static std::uint32_t rand1u (std::uint32_t &seed)

  Generate random uint32_t values in the [0, 2^24) range with the Linear Congruential Generator (LGC).
- static float rand1f (std::uint32_t &seed)

  Generate random float values in the [0, 1) range with the Linear Congruential Generator (LGC).
- static VectorTypes::float2 rand2f (std::uint32_t &seed)

  Generate random float2 values in the [0, 1) range with the Linear Congruential Generator (LGC).
- static VectorTypes::float3 rand3f (std::uint32_t &seed)

  Generate random float3 values in the [0, 1) range with the Linear Congruential Generator (LGC).
- static VectorTypes::float4 rand4f (std::uint32_t &seed)

  Generate random float4 values in the [0, 1) range with the Linear Congruential Generator (LGC).
- static std::uint32_t asUint32 (float value)

  Get the float32 bit representation to a uint32.
- static float asFloat32 (std::uint32_t value)

  Get the uint32 bit representation to a float32.
- static std::uint32_t float32Flip (float unflippedFloatValue)

  Flip a float32 for make it sortable: finds SIGN of fp number, so: if it's 1 (negative float32) it flips all bits, if it's 0 (positive float32) it flips the sign only.
- static float float32Unflip (std::uint32_t flippedFloatValue)

  Unflip a float32 back (invert float32Flip() above): signed was flipped from above, so: if sign is 1 (negative) it flips the sign bit back, if if sign is 0 (positive) it flips all bits back.
- static std::uint64_t asUint64 (double value)

  Get the float64 bit representation to a uint64.
- static double asFloat64 (std::uint64_t value)

  Get the uint64 bit representation to a float64.
- static std::uint64_t float64Flip (double unflippedFloatValue)

  Flip a float64 for make it sortable: finds SIGN of fp number, so: if it's 1 (negative float64) it flips all bits, if it's 0 (positive float64) it flips the sign only.
- static double float64Unflip (std::uint64_t flippedFloatValue)

  Unflip a float64 back (invert float64Flip() above): signed was flipped from above, so: if sign is 1 (negative) it flips the sign bit back, if if sign is 0 (positive) it flips all bits back.

5.47.1 Detailed Description

The MathFunctions class provides some needed mathematical functions functionality (note that some functions emulate GLSL-style CPU functionality).

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.47.2 Member Function Documentation

5.47.2.1 asFloat32()

```
static float Utils::UtilityFunctions::MathFunctions::asFloat32 (
             std::uint32_t value )  [inline], [static]
```

Get the uint32 bit representation to a float32.

Author

Thanos Theo, 2018

5.47.2.2 asFloat64()

```
static double Utils::UtilityFunctions::MathFunctions::asFloat64 (
             std::uint64_t value )  [inline], [static]
```

Get the uint64 bit representation to a float64.

Author

Thanos Theo, 2018

5.47.2.3 asUint32()

```
static std::uint32_t Utils::UtilityFunctions::MathFunctions::asUint32 (
             float value )  [inline], [static]
```

Get the float32 bit representation to a uint32.

Author

Thanos Theo, 2018

5.47.2.4 asUint64()

```
static std::uint64_t Utils::UtilityFunctions::MathFunctions::asUint64 (
            double value ) [inline], [static]
```

Get the float64 bit representation to a uint64.

Author

Thanos Theo, 2018

5.47.2.5 float32Flip()

```
static std::uint32_t Utils::UtilityFunctions::MathFunctions::float32Flip (
            float unflippedFloatValue ) [inline], [static]
```

Flip a float32 for make it sortable: finds SIGN of fp number, so: if it's 1 (negative float32) it flips all bits, if it's 0 (positive float32) it flips the sign only.

Needs IEEE 754 hardware compliance. Based on `http://stereopsis.com/radix.html`.

Author

Thanos Theo, 2018

5.47.2.6 float32Unflip()

```
static float Utils::UtilityFunctions::MathFunctions::float32Unflip (
            std::uint32_t flippedFloatValue ) [inline], [static]
```

Unflip a float32 back (invert float32Flip() above): signed was flipped from above, so: if sign is 1 (negative) it flips the sign bit back, if if sign is 0 (positive) it flips all bits back.

Needs IEEE 754 hardware compliance. Based on `http://stereopsis.com/radix.html`.

Author

Thanos Theo, 2018

5.47.2.7 float64Flip()

```
static std::uint64_t Utils::UtilityFunctions::MathFunctions::float64Flip (
            double unflippedFloatValue ) [inline], [static]
```

Flip a float64 for make it sortable: finds SIGN of fp number, so: if it's 1 (negative float64) it flips all bits, if it's 0 (positive float64) it flips the sign only.

Needs IEEE 754 hardware compliance. Based on `http://stereopsis.com/radix.html`.

Author

Thanos Theo, 2018

5.47.2.8 float64Unflip()

```
static double Utils::UtilityFunctions::MathFunctions::float64Unflip (
             std::uint64_t flippedFloatValue )  [inline], [static]
```

Unflip a float64 back (invert float64Flip() above): signed was flipped from above, so: if sign is 1 (negative) it flips the sign bit back, if if sign is 0 (positive) it flips all bits back.

Needs IEEE 754 hardware compliance. Based on `http://stereopsis.com/radix.html`.

Author

Thanos Theo, 2018

5.47.2.9 rand1() [1/2]

```
static float Utils::UtilityFunctions::MathFunctions::rand1 (
             const VectorTypes::float2 & seed )  [inline], [static]
```

This function returns uniformly distributed float values in the range [0, 1] (float version).

Author

Thanos Theo, 2018

5.47.2.10 rand1() [2/2]

```
static double Utils::UtilityFunctions::MathFunctions::rand1 (
             const VectorTypes::double2 & seed )  [inline], [static]
```

This function returns uniformly distributed double values in the range [0, 1] (double version).

Author

Thanos Theo, 2018

5.47.2.11 rand1f()

```
static float Utils::UtilityFunctions::MathFunctions::rand1f (
             std::uint32_t & seed )  [inline], [static]
```

Generate random float values in the [0, 1) range with the Linear Congruential Generator (LGC).

Author

Thanos Theo, 2018

5.47.2.12 rand1u()

```
static std::uint32_t Utils::UtilityFunctions::MathFunctions::rand1u (
             std::uint32_t & seed )  [inline], [static]
```

Generate random uint32_t values in the [0, 2^24) range with the Linear Congruential Generator (LGC).

Author

Thanos Theo, 2018

5.47.2.13 rand2() [1/2]

```
static VectorTypes::float2 Utils::UtilityFunctions::MathFunctions::rand2 (
             const VectorTypes::float2 & seed )  [inline], [static]
```

This function returns uniformly distributed float2 values in the range [0, 1] (float version).

Author

Thanos Theo, 2018

5.47.2.14 rand2() [2/2]

```
static VectorTypes::double2 Utils::UtilityFunctions::MathFunctions::rand2 (
             const VectorTypes::double2 & seed )  [inline], [static]
```

This function returns uniformly distributed double2 values in the range [0, 1] (double version).

Author

Thanos Theo, 2018

5.47.2.15 rand2f()

```
static VectorTypes::float2 Utils::UtilityFunctions::MathFunctions::rand2f (
             std::uint32_t & seed )  [inline], [static]
```

Generate random float2 values in the [0, 1) range with the Linear Congruential Generator (LGC).

Author

Thanos Theo, 2018

5.47.2.16 rand3() [1/2]

```
static VectorTypes::float3 Utils::UtilityFunctions::MathFunctions::rand3 (
            const VectorTypes::float2 & seed ) [inline], [static]
```

This function returns uniformly distributed float3 values in the range [0, 1] (float version).

Author

Thanos Theo, 2018

5.47.2.17 rand3() [2/2]

```
static VectorTypes::double3 Utils::UtilityFunctions::MathFunctions::rand3 (
            const VectorTypes::double2 & seed ) [inline], [static]
```

This function returns uniformly distributed double3 values in the range [0, 1] (double version).

Author

Thanos Theo, 2018

5.47.2.18 rand3f()

```
static VectorTypes::float3 Utils::UtilityFunctions::MathFunctions::rand3f (
            std::uint32_t & seed ) [inline], [static]
```

Generate random float3 values in the [0, 1) range with the Linear Congruential Generator (LGC).

Author

Thanos Theo, 2018

5.47.2.19 rand4() [1/2]

```
static VectorTypes::float4 Utils::UtilityFunctions::MathFunctions::rand4 (
            const VectorTypes::float2 & seed ) [inline], [static]
```

This function returns uniformly distributed float4 values in the range [0, 1] (float version).

Author

Thanos Theo, 2018

5.47.2.20 rand4() [2/2]

```
static VectorTypes::double4 Utils::UtilityFunctions::MathFunctions::rand4 (
             const VectorTypes::double2 & seed )  [inline], [static]
```

This function returns uniformly distributed double4 values in the range [0, 1] (double version).

Author

Thanos Theo, 2018

5.47.2.21 rand4f()

```
static VectorTypes::float4 Utils::UtilityFunctions::MathFunctions::rand4f (
             std::uint32_t & seed )  [inline], [static]
```

Generate random float4 values in the [0, 1) range with the Linear Congruential Generator (LGC).

Author

Thanos Theo, 2018

5.47.2.22 seedGenerator()

```
template<std::uint32_t N>
static std::uint32_t Utils::UtilityFunctions::MathFunctions::seedGenerator (
             std::uint32_t value0,
             std::uint32_t value1 )  [inline], [static]
```

Seed generator for the Linear Congruential Generator (LGC).

Author

Thanos Theo, 2018

5.47.2.23 smootherstep()

```
template<typename T >
static T Utils::UtilityFunctions::MathFunctions::smootherstep (
             const T & edge0,
             const T & edge1,
             T x,
             std::enable_if_t< std::is_arithmetic< T >::value > *  = nullptr )  [inline],
[static]
```

Prof.

Ken Perlin suggests an improved version of the smoothstep function which has zero 1st and 2nd order derivatives at t=0 and t=1. Scale, and clamp x to 0...1 (first line) range & evaluate polynomial (second line). Look at http↩ ://en.wikipedia.org/wiki/Smoothstep -> smootherstep. GLSL-style smootherstep function.

5.48 Utils::NewHandlerSupport< T >::NewHandlerHolder Class Reference

Public Member Functions

- **NewHandlerHolder** (std::new_handler newHandler)
- **NewHandlerHolder** (const NewHandlerHolder &)=delete
- **NewHandlerHolder** (NewHandlerHolder &&)=delete
- NewHandlerHolder & **operator=** (const NewHandlerHolder &)=delete
- NewHandlerHolder & **operator=** (NewHandlerHolder &&)=delete

Private Attributes

- std::new_handler **handler**

5.49 Utils::NewHandlerSupport< T > Class Template Reference

"Mixin-style" base class for class-specific std::set_new_handler support.

```
#include <NewHandlerSupport.h>
```

Classes

- class NewHandlerHolder

Public Member Functions

- **NewHandlerSupport** (const NewHandlerSupport &)=delete
- **NewHandlerSupport** (NewHandlerSupport &&)=delete
- NewHandlerSupport & **operator=** (const NewHandlerSupport &)=delete
- NewHandlerSupport & **operator=** (NewHandlerSupport &&)=delete

Static Public Member Functions

- static std::new_handler **set_new_handler** (std::new_handler newHandler) noexcept
- static void ∗ **operator new** (std::size_t size) noexcept(false)
- static void **operator delete** (void ∗pMemory) noexcept
- static void ∗ **operator new** (std::size_t size, void ∗ptr) noexcept
- static void **operator delete** (void ∗pMemory, void ∗ptr) noexcept
- static void ∗ **operator new** (std::size_t size, const std::nothrow_t &nt) noexcept
- static void **operator delete** (void ∗pMemory, const std::nothrow_t &nt) noexcept

Static Private Attributes

- static std::new_handler **currentHandler** = nullptr

5.49.1 Detailed Description

template<typename T>
class Utils::NewHandlerSupport< T >

"Mixin-style" base class for class-specific std::set_new_handler support.

Author

Thanos Theo, 2009-2018

Version

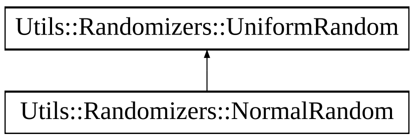
14.0.0.0

5.50 Utils::Randomizers::NormalRandom Class Reference

The NormalRandom class provides a normal random number generator.

```
#include <Randomizers.h>
```

Inheritance diagram for Utils::Randomizers::NormalRandom:


Utils::Randomizers::UniformRandom
Utils::Randomizers::NormalRandom

Public Member Functions

- double **getNormalFloat** ()
- double **operator()** ()
- **NormalRandom** (const NormalRandom &)=delete
- **NormalRandom** (NormalRandom &&)=delete
- NormalRandom & **operator=** (const NormalRandom &)=delete
- NormalRandom & **operator=** (NormalRandom &&)=delete

Private Attributes

- std::normal_distribution< double > **_normalDistribution**

Additional Inherited Members

5.50.1 Detailed Description

The NormalRandom class provides a normal random number generator.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.51 Utils::SIMDVectorizations::not_vec4 Class Reference

The not_vec4 class is an internal class: not be used directly.

```
#include <SIMDVectorizations.h>
```

Public Member Functions

- **not_vec4** (__m128 value)
- __m128 **get** () const

Private Attributes

- __m128 **_v**

5.51.1 Detailed Description

The not_vec4 class is an internal class: not be used directly.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.52 Utils::SIMDVectorizations::not_vec8 Class Reference

The not_vec8 class is an internal class: not be used directly.

```
#include <SIMDVectorizations.h>
```

Public Member Functions

- **not_vec8** (__m256 value)
- __m256 **get** () const

Private Attributes

- __m256 **_v**

5.52.1 Detailed Description

The not_vec8 class is an internal class: not be used directly.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.53 OpenGLRenderingEngine::OpenGLAssetManager Struct Reference

This class encapsulates usage of an OpenGL Asset Manager.

```
#include <OpenGLAssetManager.h>
```

Public Types

- enum **ShaderTypes** : std::size_t {
  VS = (1 << 0), **TCS** = (1 << 1), **TES** = (1 << 2), **GS** = (1 << 3),
  FS = (1 << 4), **CS** = (1 << 5) }
- enum **CharacterEncodingMethods** : std::size_t {
  NONE = 0, **BASE64** = 1, **FLIP_BITS** = 2, **FLIP_XOR_SWAP_BITS** = 3,
  BASE64_FLIP_XOR_SWAP_BITS = 4, **BASE64_COMPRESSION** = 5, **BASE64_FLIP_XOR_SWAP_BI↩
  TS_COMPRESSION** = 6 }

Public Member Functions

- **OpenGLAssetManager** (const OpenGLAssetManager &)=delete
- **OpenGLAssetManager** (OpenGLAssetManager &&)=delete
- OpenGLAssetManager & **operator=** (const OpenGLAssetManager &)=delete
- OpenGLAssetManager & **operator=** (OpenGLAssetManager &&)=delete

Static Public Member Functions

- static std::string **getGLSLInternalDirectory** ()
- static std::string **getVertexShadersFileName** ()
- static std::string **getTessellationControlShadersFileName** ()
- static std::string **getTessellationEvaluationShadersFileName** ()
- static std::string **getGeometryShadersFileName** ()
- static std::string **getFragmentShadersFileName** ()
- static std::string **getComputeShadersFileName** ()
- static std::string **getVertexShadersFileNameExtension** ()
- static std::string **getTessellationControlShadersFileNameExtension** ()
- static std::string **getTessellationEvaluationShadersFileNameExtension** ()
- static std::string **getGeometryShadersFileNameExtension** ()
- static std::string **getFragmentShadersFileNameExtension** ()
- static std::string **getComputeShadersFileNameExtension** ()
- static std::string **getAssetsDefaultDirectory** ()
- static std::string **getGLSLDefaultDirectory** ()
- static std::string **getImagesDefaultDirectory** ()
- static std::string **getModelsDefaultDirectory** ()
- static std::string **getTexturesDefaultDirectory** ()
- static std::string **getDefaultScreenshotFormat** ()

Static Public Attributes

- static const size_t **NUMBER_OF_TOTAL_SHADER_TYPES** = 6
- static const CharacterEncodingMethods **CHARACTER_ENCODING_METHOD** = BASE64_FLIP_XOR_S↩ WAP_BITS_COMPRESSION

5.53.1 Detailed Description

This class encapsulates usage of an OpenGL Asset Manager.

Author

Thanos Theo, 2009-2018

Version

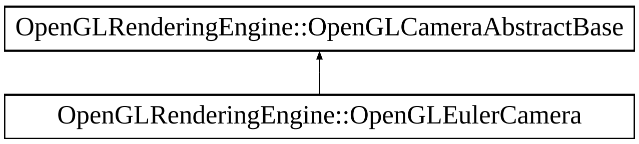
14.0.0.0

5.54 OpenGLRenderingEngine::OpenGLCameraAbstractBase Class Reference

This abstract class encapsulates usage of an OpenGL camera.

`#include <OpenGLCameraAbstractBase.h>`

Inheritance diagram for OpenGLRenderingEngine::OpenGLCameraAbstractBase:


OpenGLRenderingEngine::OpenGLCameraAbstractBase
OpenGLRenderingEngine::OpenGLEulerCamera

Public Member Functions

- float **getFieldOfView** () const
- float **getRatio** () const
- float **getViewNear** () const
- float **getViewFar** () const
- void **setFieldOfView** (float fieldOfView)
- void **setRatio** (float ratio)
- void **setViewNear** (float viewNear)
- void **setViewFar** (float viewFar)
- virtual void **setMatrices** () const =0
- **OpenGLCameraAbstractBase** (const OpenGLCameraAbstractBase &)=delete
- **OpenGLCameraAbstractBase** (OpenGLCameraAbstractBase &&)=delete
- OpenGLCameraAbstractBase & **operator=** (const OpenGLCameraAbstractBase &)=delete
- OpenGLCameraAbstractBase & **operator=** (OpenGLCameraAbstractBase &&)=delete

Protected Member Functions

- **OpenGLCameraAbstractBase** (float fieldOfView) noexcept

Protected Attributes

- float **_fieldOfView** = 0.0f
- float **_ratio** = 0.0f
- float **_viewNear** = 0.01f
- float **_viewFar** = std::numeric_limits<float>::max()

5.54.1 Detailed Description

This abstract class encapsulates usage of an OpenGL camera.

To be inherited from usage-specific sub-classes.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.55 OpenGLRenderingEngine::OpenGLDriverInfo Class Reference

Gets GL vendor, version, supported extensions and other states using glGet∗ functions and store them in Open↩
GLDriverInfo class variables.

```
#include <OpenGLDriverInfo.h>
```

Public Member Functions

- std::string getVendor () const

    all getter functions inlined return-by-value instead return-by-reference so as to avoid changing original string
- std::string getRenderer () const

    return-by-value instead return-by-reference so as to avoid changing original string
- std::string getVersion () const

    return-by-value instead return-by-reference so as to avoid changing original string
- std::string getShadingLanguageVersion () const

    return-by-value instead return-by-reference so as to avoid changing original string
- std::set< std::string > getExtensions () const

    return-by-value instead return-by-reference so as to avoid changing original string
- bool **isNvidia** () const
- bool **isAMDATI** () const
- bool **isIntel** () const
- bool **isMesa** () const
- bool **isMicrosoft** () const
- bool **supports120Shaders** () const
- bool **supports330Shaders** () const
- bool **supports400Shaders** () const
- bool **supports420Shaders** () const
- bool **supports430Shaders** () const
- bool **supports440Shaders** () const
- bool **supports450Shaders** () const
- bool **isVSynchSupported** () const
- GLint **getRedBits** () const
- GLint **getGreenBits** () const
- GLint **getBlueBits** () const
- GLint **getAlphaBits** () const
- GLint **getDepthBits** () const
- GLint **getStencilBits** () const
- GLint **getAccumRedBits** () const
- GLint **getAccumGreenBits** () const
- GLint **getAccumBlueBits** () const
- GLint **getAccumAlphaBits** () const
- GLint **getSampleBuffers** () const
- GLint **getMaxTextureSize** () const
- GLint **getMaxTextureBufferSize** () const
- GLint **getMaxTextureMaxAnisotropy** () const
- GLint **getMaxRenderBufferSize** () const
- GLint **getMaxColorAttachments** () const
- GLint **getMaxLights** () const
- GLint **getMaxAttribStacks** () const
- GLint **getMaxModelViewStacks** () const
- GLint **getMaxProjectionStacks** () const
- GLint **getMaxClipPlanes** () const
- GLint **getMaxTextureStacks** () const
- GLint **getMaxGeometryOutputVertices** () const
- GLint **getMaxTessellationControlOutputComponents** () const
- GLint **getMaxTessellationGenerationLevel** () const
- GLint **getGPUMemoryInfoDedicatedVidmemNVX** () const
- GLint **getGPUMemoryInfoTotalAvailableMemoryNVX** () const
- GLint **getGPUMemoryInfoCurrentAvailableMemoryNVX** () const
- GLint **getGPUMemoryInfoEvictionCountNVX** () const

- GLint **getGPUMemoryInfoEvictedMemoryNVX** () const
- const GLint ∗ **getVBOFreeMemoryATI** () const
- const GLint ∗ **getTextureFreeMemoryATI** () const
- const GLint ∗ **getRenderBufferFreeMemoryATI** () const
- bool **supports_GL_ARB_texture_rectangle** () const
- bool **supports_GL_ARB_texture_buffer_object** () const
- bool **supports_GL_EXT_texture_filter_anisotropic** () const
- bool **supports_GL_EXT_framebuffer_object** () const
- bool **supports_GL_EXT_framebuffer_multisample** () const
- bool **supports_GL_EXT_framebuffer_blit** () const
- bool **supports_GL_EXT_packed_depth_stencil** () const
- bool **supports_GL_EXT_gpu_shader4** () const
- bool **supports_GL_ARB_geometry_shader4** () const
- bool **supports_GL_ARB_tessellation_shader** () const
- bool **supports_GL_ARB_compute_shader** () const
- bool **supports_GL_ARB_gpu_shader5** () const
- bool **supports_GL_ARB_gpu_shader_fp64** () const
- bool **supports_GL_ARB_vertex_type_2_10_10_10_rev** () const
- bool **supports_GL_NVX_gpu_memory_info** () const
- bool **supports_GL_ATI_meminfo** () const
- void getGLMemoryInfo ()

  extract GL memory info
- void printGLInfo () const

  print GL info
- void printGLMemoryInfo () const

  print GL memory info
- bool isGLExtensionSupported (const std::string &extension) const

  check if a GL extension is supported
- std::string getConciseGLDriverInfo () const

  get a concise GL driver info string
- **OpenGLDriverInfo** (const OpenGLDriverInfo &)=delete
- **OpenGLDriverInfo** (OpenGLDriverInfo &&)=delete
- OpenGLDriverInfo & **operator=** (const OpenGLDriverInfo &)=delete
- OpenGLDriverInfo & **operator=** (OpenGLDriverInfo &&)=delete

Static Public Member Functions

- static float **getMinimumGLVersionForQualityRenderingAndShaders** ()
- static std::string **getMinimumGLSLVersionFor120Shaders** ()
- static std::string **getMinimumGLSLVersionFor330Shaders** ()
- static std::string **getMinimumGLSLVersionFor400Shaders** ()
- static std::string **getMinimumGLSLVersionFor420Shaders** ()
- static std::string **getMinimumGLSLVersionFor430Shaders** ()
- static std::string **getMinimumGLSLVersionFor440Shaders** ()
- static std::string **getMinimumGLSLVersionFor450Shaders** ()
- static std::string **getGLSLLanguageMode** ()

Private Member Functions

- void getGLInfo ()

  extract GL info

Private Attributes

- std::string **_vendor**
- std::string **_renderer**
- std::string **_version**
- std::string **_shadingLanguageVersion**
- std::set< std::string > **_extensions**
- bool **_isNvidia** = false
- bool **_isAMDATI** = false
- bool **_isIntel** = false
- bool **_isMesa** = false
- bool **_isMicrosoft** = false
- bool **_use120Shaders** = false
- bool **_use330Shaders** = false
- bool **_use400Shaders** = false
- bool **_use420Shaders** = false
- bool **_use430Shaders** = false
- bool **_use440Shaders** = false
- bool **_use450Shaders** = false
- bool **_isVSynchSupported** = false
- GLint **_redBits** = 0
- GLint **_greenBits** = 0
- GLint **_blueBits** = 0
- GLint **_alphaBits** = 0
- GLint **_depthBits** = 0
- GLint **_stencilBits** = 0
- GLint **_accumRedBits** = 0
- GLint **_accumGreenBits** = 0
- GLint **_accumBlueBits** = 0
- GLint **_accumAlphaBits** = 0
- GLint **_sampleBuffers** = 0
- GLint **_samples** = 0
- GLint **_maxTextureSize** = 0
- GLint **_maxTextureBufferSize** = 0
- GLint **_maxTextureMaxAnisotropy** = 0
- GLint **_maxRenderBufferSize** = 0
- GLint **_maxColorAttachments** = 0
- GLint **_maxLights** = 0
- GLint **_maxAttribStacks** = 0
- GLint **_maxModelViewStacks** = 0
- GLint **_maxProjectionStacks** = 0
- GLint **_maxClipPlanes** = 0
- GLint **_maxTextureStacks** = 0
- GLint **_maxGeometryOutputVertices** = 0
- GLint **_maxTessellationPatchVertices** = 0
- GLint **_maxTessellationControlOutputComponents** = 0
- GLint **_maxTessellationGenerationLevel** = 0
- GLint **_maxUniformBufferBindings** = 0
- GLint **_maxUniformBlockSize** = 0
- GLint **_maxCombinedVertexUniformComponents** = 0
- GLint **_maxCombinedGeometryUniformComponents** = 0
- GLint **_maxCombinedTessellationControlUniformComponents** = 0
- GLint **_maxCombinedTessellationEvaluationUniformComponents** = 0
- GLint **_maxCombinedFragmentUniformComponents** = 0
- GLint **_GPUMemoryInfoDedicatedVidmemNVX** = 0

- GLint **_GPUMemoryInfoTotalAvailableMemoryNVX** = 0
- GLint **_GPUMemoryInfoCurrentAvailableMemoryNVX** = 0
- GLint **_GPUMemoryInfoEvictionCountNVX** = 0
- GLint **_GPUMemoryInfoEvictedMemoryNVX** = 0
- GLint **_vboFreeMemoryATI** [4] = { 0 }
- GLint **_textureFreeMemoryATI** [4] = { 0 }
- GLint **_renderBufferFreeMemoryATI** [4] = { 0 }

5.55.1 Detailed Description

Gets GL vendor, version, supported extensions and other states using glGet∗ functions and store them in Open←GLDriverInfo class variables.

get valid OpenGL infos, an OpenGL rendering context (RC) must be opened before calling OpenGLDriverInfo←::getGLInfo(). Otherwise it returns false.

Author

Thanos Theo, 2009-2018

Version

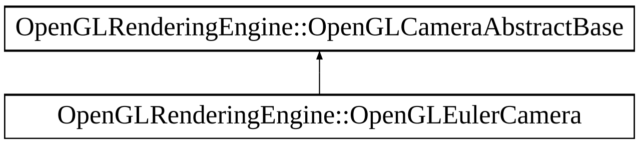
14.0.0.0

5.56 OpenGLRenderingEngine::OpenGLEulerCamera Class Reference

This class encapsulates usage of an OpenGL Euler camera.

```
#include <OpenGLEulerCamera.h>
```

Inheritance diagram for OpenGLRenderingEngine::OpenGLEulerCamera:


OpenGLRenderingEngine::OpenGLCameraAbstractBase
OpenGLRenderingEngine::OpenGLEulerCamera

Public Member Functions

- void **setMatrices** () const override
- void **setEulerCamera** (double cameraDistanceX, double cameraDistanceY, double cameraDistanceZ, double cameraAngleX, double cameraAngleY, double sceneScaleFactor=0.0, double sceneCenterX=0.0, double sceneCenterY=0.0, double sceneCenterZ=0.0) const
- **OpenGLEulerCamera** (float fieldOfView) noexcept
- **OpenGLEulerCamera** (const OpenGLEulerCamera &)=delete
- **OpenGLEulerCamera** (OpenGLEulerCamera &&)=delete
- OpenGLEulerCamera & **operator=** (const OpenGLEulerCamera &)=delete
- OpenGLEulerCamera & **operator=** (OpenGLEulerCamera &&)=delete

Additional Inherited Members

5.56.1 Detailed Description

This class encapsulates usage of an OpenGL Euler camera.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.57 OpenGLRenderingEngine::OpenGLFrameBufferObject Class Reference

This class provides Frame Buffer Object support using the GL_EXT_framebuffer_object OpenGL extension.

```
#include <OpenGLFrameBufferObject.h>
```

Public Member Functions

- GLsizei **getWidth** () const
- GLsizei **getHeight** () const
- GLuint **getTextureID** () const
- bool **getUseTexture** () const
- bool **getDepthRenderBuffer** () const
- bool **getUseDepthTexture** () const
- bool **getDepthStencilRenderBuffer** () const
- GLint **getTextureFormat1** () const
- GLenum **getTextureFormat2** () const
- GLenum **getTextureFormatType** () const
- bool **getGenerateMipmap** () const
- std::pair< bool, GLint > **getMultisampleFBO** () const
- void **setUseTexture** (bool useTexture)
- void setDepthRenderBuffer (bool depthRenderBuffer)

  Sets the depthRenderBuffer value.
- void setUseDepthTexture (bool useDepthTexture)

  Sets the shadowMap value.
- void setDepthStencilRenderBuffer (bool depthStencilRenderBuffer)

  Sets the depthStencilRenderBuffer value.
- void **setTextureFormat1** (GLint textureFormat1)
- void **setTextureFormat2** (GLenum textureFormat2)
- void **setTextureFormatType** (GLenum textureFormatType)
- void **setGenerateMipmap** (bool generateMipmap)
- void setMultisampleFBO (bool multisampleFBO, GLint numberOfSamples)

  Sets the multisampleFBO value.
- void initFrameBufferObjectResources (GLsizei width, GLsizei height, GLenum textureUnit=0, GLenum depthTextureUnit=1)

  Initializes all Frame Buffer Object resources.

- void startRender () const

  Binds the framebuffer & sets the viewport to the given texture dimensions (uses glPushAttrib).
- void finishRender () const

  Unbinds the framebuffer & returns to default state.
- void enable () const

  Enable the fbo texture.
- void disable () const

  Disable the fbo texture.
- void bindTexture (GLenum textureUnit=0) const

  Binds the fbo texture with a given active texture unit.
- void bindDepthTexture (GLenum depthTextureUnit=0) const

  Binds the fbo depth texture with a given active depth texture unit.
- void unbind (GLenum textureUnit=0) const

  Unbinds the fbo texture with a given active texture unit.
- void **renderTextureToFullScreenQuad** (GLenum textureUnit, bool isNvidia, bool useDummyVAO=true) const
- void **renderDepthTextureToFullScreenQuad** (GLenum depthTextureUnit, bool isNvidia, bool useDummy↩ VAO=true) const
- void **disposeFrameBufferObjectResources** ()
- **OpenGLFrameBufferObject** (OpenGLDriverInfo ∗openGLDriverInfo, bool useTexture=true, bool depth↩ RenderBuffer=false, bool useDepthTexture=false, bool depthStencilRenderBuffer=false, GLint texture↩ Format1=GL_RGBA8, GLenum textureFormat2=GL_RGBA, GLenum textureFormatType=GL_UNSIGNE↩ D_BYTE, bool generateMipmap=false, bool multisampleFBO=false, GLint numberOfSamples=4) noexcept
- **OpenGLFrameBufferObject** (const OpenGLFrameBufferObject &)=delete
- **OpenGLFrameBufferObject** (OpenGLFrameBufferObject &&)=delete
- OpenGLFrameBufferObject & **operator=** (const OpenGLFrameBufferObject &)=delete
- OpenGLFrameBufferObject & **operator=** (OpenGLFrameBufferObject &&)=delete

Private Member Functions

- void initTextureResouces (GLenum textureUnit=0)

  Initializes the Frame Buffer Object texture resources.
- void initDepthTextureResouces (GLenum depthTextureUnit=0)

  Initializes the Frame Buffer Object depth texture resources.
- void initTextureParameters () const

  Initializes the Frame Buffer Object texture parameters.
- void printFramebufferInfo () const

  Prints information about the Frame Buffer Object (FBO).
- std::string getTextureParameters (GLuint id) const

  Returns the texture parameters as string using glGetTexLevelParameteriv().
- std::string getRenderbufferParameters (GLuint id) const

  Returns the renderbuffer parameters as string using glGetRenderbufferParameteriv().

Private Attributes

- OpenGLDriverInfo ∗ **_openGLDriverInfo** = nullptr
- GLuint **_fboID** = 0
- GLuint **_fboMultiSampleID** = 0
- GLuint **_renderBufferMultiSampleID** = 0
- GLuint **_renderBufferID** = 0
- GLsizei **_width** = 0

- GLsizei **_height** = 0
- GLuint **_textureID** = 0
- GLuint **_depthTextureID** = 0
- bool **_depthRenderBuffer** = false
- bool **_useTexture** = false
- bool **_useDepthTexture** = false
- bool **_depthStencilRenderBuffer** = false
- GLint **_textureFormat1** = 0
- GLenum **_textureFormat2** = 0
- GLenum **_textureFormatType** = 0
- bool **_generateMipmap** = false
- bool **_multisampleFBO** = false
- GLint **_numberOfSamples** = 0
- GLuint **_dummyVao** = 0

5.57.1 Detailed Description

This class provides Frame Buffer Object support using the GL_EXT_framebuffer_object OpenGL extension.

also supports Frame Buffer Object multisampling via the GL_EXT_framebuffer_multisample & GL_EXT_↩ framebuffer_blit extensions.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.57.2 Member Function Documentation

5.57.2.1 finishRender()

```
void OpenGLFrameBufferObject::finishRender ( ) const
```

Unbinds the framebuffer & returns to default state.

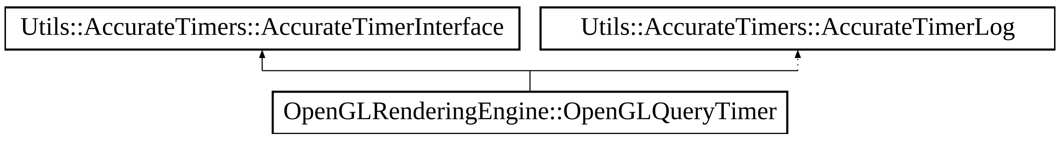
Always restore the viewport when ready to render to the screen (uses glPopAttrib).

5.58 OpenGLRenderingEngine::OpenGLQueryTimer Class Reference

This class contains an AccurateTimers encapsulation of OpenGL query timers.

`#include <OpenGLQueryTimer.h>`

Inheritance diagram for OpenGLRenderingEngine::OpenGLQueryTimer:


Utils::AccurateTimers::AccurateTimerInterface
Utils::AccurateTimers::AccurateTimerLog
OpenGLRenderingEngine::OpenGLQueryTimer

Public Member Functions

- void **startTimer** () override
- void **stopTimer** () override
- double **getElapsedTimeInNanoSecs** () override
- double **getElapsedTimeInMicroSecs** () override
- double **getElapsedTimeInMilliSecs** () override
- double **getElapsedTimeInSecs** () override
- double **getMeanTimeInNanoSecs** () override
- double **getMeanTimeInMicroSecs** () override
- double **getMeanTimeInMilliSecs** () override
- double **getMeanTimeInSecs** () override
- double **getDecimalElapsedTimeInMicroSecs** () override
- double **getDecimalElapsedTimeInMilliSecs** () override
- double **getDecimalElapsedTimeInSecs** () override
- double **getDecimalMeanTimeInMicroSecs** () override
- double **getDecimalMeanTimeInMilliSecs** () override
- double **getDecimalMeanTimeInSecs** () override
- **OpenGLQueryTimer** (const OpenGLQueryTimer &)=delete
- **OpenGLQueryTimer** (OpenGLQueryTimer &&)=delete
- OpenGLQueryTimer & **operator=** (const OpenGLQueryTimer &)=delete
- OpenGLQueryTimer & **operator=** (OpenGLQueryTimer &&)=delete

Private Member Functions

- double **getElapsedTime_** ()

Private Attributes

- bool **stopped_** = false
- GLuint **query_** = 0
- GLuint64 **renderingTime_** = 0

Additional Inherited Members

5.58.1 Detailed Description

This class contains an AccurateTimers encapsulation of OpenGL query timers.

OpenGLQueryTimer.h:

This class contains an AccurateTimers encapsulation of OpenGL query timers. Note: no virtual destructor is needed for data-oriented design ie no up-casting should ever be used.

Author

Thanos Theo, 2018

5.59 OpenGLRenderingEngine::OpenGLShaderCompileAndLink Class Reference

This class encapsulates loading, compilation & linking of a GLSL program.

`#include <OpenGLShaderCompileAndLink.h>`

Public Member Functions

- void addShaderLibraryToProgram (const std::string &shaderLibraryPathName, const std::string &shader↩ LibraryName, int shaderType)

  add shader library to program
- void linkShaderProgram (GLint inputTopology=GL_TRIANGLES, GLint outputTopology=GL_TRIANGLE_S↩ TRIP, GLint maxVerticesOut=256)

  link shader program
- **OpenGLShaderCompileAndLink** (OpenGLDriverInfo ∗openGLDriverInfo, OpenGLShaderGLSLPre↩ ProcessorCommands ∗openGLShaderGLSLPreProcessorCommands, GLuint shaderProgram) noexcept
- **OpenGLShaderCompileAndLink** (const OpenGLShaderCompileAndLink &)=delete
- **OpenGLShaderCompileAndLink** (OpenGLShaderCompileAndLink &&)=delete
- OpenGLShaderCompileAndLink & **operator=** (const OpenGLShaderCompileAndLink &)=delete
- OpenGLShaderCompileAndLink & **operator=** (OpenGLShaderCompileAndLink &&)=delete

Private Member Functions

- void compileShader (const std::string &shaderLibraryPathName, const std::string &shaderLibraryName, const std::string &shaderFileNameExtension, const std::string &shaderFileName, int shaderTypeEnum, const OpenGLShaderObjects ∗openGLShaderObjects, const std::string &shaderTypeString) const

  compile shader
- bool checkUsageOfGeometryShaderObject ()

  check if a geometry shader object was created
- void checkInfoLog (const std::string &shaderName, GLuint obj) const

  check GL info log function
- void releaseAllShaderObjects ()

  release all shader objects

Private Attributes

- OpenGLDriverInfo ∗ **_openGLDriverInfo** = nullptr
- OpenGLShaderGLSLPreProcessorCommands ∗ **_openGLShaderGLSLPreProcessorCommands** = nullptr
- GLuint **_shaderProgram** = 0
- std::string **_mergedShaderLibraryName** = ""
- std::unordered_map< std::string, OpenGLShaderObjects ∗ > **_allOpenGLShaderObjectsMap**

5.59.1 Detailed Description

This class encapsulates loading, compilation & linking of a GLSL program.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.59.2 Member Function Documentation

5.59.2.1 checkInfoLog()

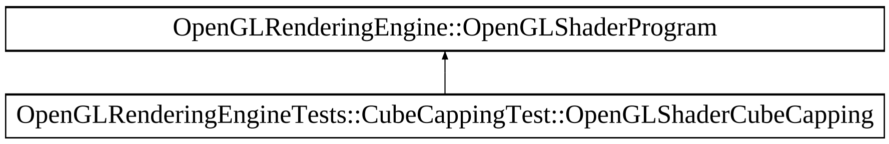
```
void OpenGLShaderCompileAndLink::checkInfoLog (
             const std::string & shaderName,
             GLuint obj ) const  [private]
```

check GL info log function

Checks the OpenGL info log of the shader loading process.

5.60 OpenGLRenderingEngineTests::CubeCappingTest::OpenGLShaderCubeCapping Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::CubeCappingTest::OpenGLShaderCubeCapping:


OpenGLRenderingEngine::OpenGLShaderProgram
OpenGLRenderingEngineTests::CubeCappingTest::OpenGLShaderCubeCapping

Public Member Functions

- **OpenGLShaderCubeCapping** (OpenGLRenderingEngine::OpenGLDriverInfo ∗openGLDriverInfo)

Private Member Functions

- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.61 OpenGLRenderingEngine::OpenGLShaderGLSLPreProcessorCommands Class Reference

This class is responsible for the GLSL shader preprocessor process.

`#include <OpenGLShaderGLSLPreProcessorCommands.h>`

Public Member Functions

- void addHighestGLVersionDefinition ()
  add highest GL version definition
- void addGL21VersionDefinition ()
  add GL 2.1 version definition
- void addGL33VersionDefinition ()
  add GL 3.3 version definition
- void addGL42VersionDefinition ()
  add GL 4.2 version definition
- void addGL43VersionDefinition ()
  add GL 4.3 version definition
- void addGL44VersionDefinition ()
  add GL 4.4 version definition
- void addGL45VersionDefinition ()
  add GL 4.5 version definition
- void addDefinition (const std::string &definition)
  add definition
- void addDefinitionAndCondition (const std::string &definition, int condition)
  add definition and condition
- void addPreprocessorLine (const std::string &GLSLPreProcessorLine)
  add preprocessor line
- void addDefinitionForStartingLine ()
  add definition for starting line
- std::string getCurrentGLSLPreProcessorCommands () const
  get current GLSL preprocessor commands
- std::string getFinalizedGLSLPreProcessorCommands ()
  get GLSL preprocessor commands
- void clearGLSLPreProcessorCommands ()
  clear GLSL preprocessor commands
- **OpenGLShaderGLSLPreProcessorCommands** (OpenGLDriverInfo ∗openGLDriverInfo) noexcept
- **OpenGLShaderGLSLPreProcessorCommands** (const OpenGLShaderGLSLPreProcessorCommands &)=delete
- **OpenGLShaderGLSLPreProcessorCommands** (OpenGLShaderGLSLPreProcessorCommands &&)=delete
- OpenGLShaderGLSLPreProcessorCommands & **operator=** (const OpenGLShaderGLSLPreProcessor←Commands &)=delete
- OpenGLShaderGLSLPreProcessorCommands & **operator=** (OpenGLShaderGLSLPreProcessorCommands &&)=delete

Private Attributes

- OpenGLDriverInfo ∗ **_openGLDriverInfo** = nullptr
- std::ostringstream **_allGLSLPreProcessorCommands**

5.61.1 Detailed Description

This class is responsible for the GLSL shader preprocessor process.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.62 OpenGLRenderingEngine::OpenGLShaderObjects Class Reference

This class is holding all shader objects GL handles and type information.

```
#include <OpenGLShaderObjects.h>
```

Public Member Functions

- bool isUsingShaderType (int shaderTypeEnum) const

  get the shader type
- void setShaderObject (GLuint shaderObject, int shaderTypeEnum)

  set the shader object by shader type
- GLuint getShaderObject (int shaderTypeEnum) const

  get the shader object by shader type
- void attachShaderObjectsToProgram (GLuint shaderProgram)

  attach shader objects to program (only if the attach shader objects state was already set to false)
- void detachShaderObjectsFromProgram (GLuint shaderProgram)

  detach shader objects from program (only if the attach shader objects state was already set to true)
- std::size_t numberOfShaderTypeProgrammableStages () const

  number of shader type programmable stages
- std::size_t numberOfCreatedShaderObjects () const

  number of created shader objects
- bool hasCreatedShaderObjects () const

  get if any shader objects were created
- bool isEqualNumberOfShaderTypeProgrammableStagesAndCreatedShaderObjects () const

  get if the number of the shader type programmable stages equals the created shader objects
- **OpenGLShaderObjects** (int shaderType) noexcept
- **OpenGLShaderObjects** (const OpenGLShaderObjects &)=delete
- **OpenGLShaderObjects** (OpenGLShaderObjects &&)=delete
- OpenGLShaderObjects & **operator=** (const OpenGLShaderObjects &)=delete
- OpenGLShaderObjects & **operator=** (OpenGLShaderObjects &&)=delete

Private Attributes

- std::bitset< OpenGLAssetManager::NUMBER_OF_TOTAL_SHADER_TYPES > ∗ _shaderType = nullptr

  shader type flag storage variables
- GLuint _shaderObjects [OpenGLAssetManager::NUMBER_OF_TOTAL_SHADER_TYPES] = { 0 }

  all shader objects GL handles

5.62.1 Detailed Description

This class is holding all shader objects GL handles and type information.

Author

Thanos Theo, 2009-2018

Version

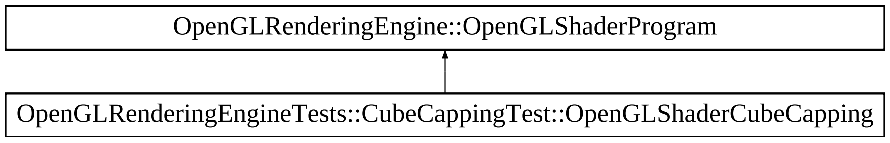
14.0.0.0

5.63 OpenGLRenderingEngine::OpenGLShaderProgram Class Reference

This abstract class encapsulates usage of a GLSL program.

```
#include <OpenGLShaderProgram.h>
```

Inheritance diagram for OpenGLRenderingEngine::OpenGLShaderProgram:


OpenGLRenderingEngine::OpenGLShaderProgram
OpenGLRenderingEngineTests::CubeCappingTest::OpenGLShaderCubeCapping

Public Member Functions

- void setUniform1i (const std::string &name, GLint value)

  integer uniform setter auxiliary function
- void setUniform1iv (const std::string &name, GLsizei count, const GLint ∗values)

  integer uniform setter auxiliary function
- void setUniform2iv (const std::string &name, GLsizei count, const GLint ∗values)

  integer uniform setter auxiliary function
- void setUniform3iv (const std::string &name, GLsizei count, const GLint ∗values)

  integer uniform setter auxiliary function
- void setUniform4iv (const std::string &name, GLsizei count, const GLint ∗values)

  integer uniform setter auxiliary function
- void setUniform1ui (const std::string &name, GLuint value)

  unsigned integer uniform setter auxiliary function
- void setUniform1uiv (const std::string &name, GLsizei count, const GLuint ∗values)

  unsigned integer uniform setter auxiliary function
- void setUniform2uiv (const std::string &name, GLsizei count, const GLuint ∗values)

  unsigned integer uniform setter auxiliary function
- void setUniform3uiv (const std::string &name, GLsizei count, const GLuint ∗values)

  unsigned integer uniform setter auxiliary function
- void setUniform4uiv (const std::string &name, GLsizei count, const GLuint ∗values)

  unsigned integer uniform setter auxiliary function
- void setUniform1i64NV (const std::string &name, GLint64EXT value)

  64bit integer (GL 4.0+ only) uniform setter auxiliary function
- void setUniform1i64vNV (const std::string &name, GLsizei count, const GLint64EXT ∗values)

  64bit integer (GL 4.0+ only) uniform setter auxiliary function
- void setUniform2i64vNV (const std::string &name, GLsizei count, const GLint64EXT ∗values)

  64bit integer (GL 4.0+ only) uniform setter auxiliary function
- void setUniform3i64vNV (const std::string &name, GLsizei count, const GLint64EXT ∗values)

  64bit integer (GL 4.0+ only) uniform setter auxiliary function
- void setUniform4i64vNV (const std::string &name, GLsizei count, const GLint64EXT ∗values)

  64bit integer (GL 4.0+ only) uniform setter auxiliary function
- void setUniform1ui64NV (const std::string &name, GLuint64EXT value)

  64bit unsigned integer (GL 4.0+ only) uniform setter auxiliary function
- void setUniform1ui64vNV (const std::string &name, GLsizei count, const GLuint64EXT ∗values)

  64bit unsigned integer (GL 4.0+ only) uniform setter auxiliary function

- void setUniform2ui64vNV (const std::string &name, GLsizei count, const GLuint64EXT ∗values)

    64bit unsigned integer (GL 4.0+ only) uniform setter auxiliary function
- void setUniform3ui64vNV (const std::string &name, GLsizei count, const GLuint64EXT ∗values)

    64bit unsigned integer (GL 4.0+ only) uniform setter auxiliary function
- void setUniform4ui64vNV (const std::string &name, GLsizei count, const GLuint64EXT ∗values)

    64bit unsigned integer (GL 4.0+ only) uniform setter auxiliary function
- void setUniform1f (const std::string &name, GLfloat value)

    float uniform setter auxiliary function
- void setUniform1fv (const std::string &name, GLsizei count, const GLfloat ∗values)

    float uniform setter auxiliary function
- void setUniform2fv (const std::string &name, GLsizei count, const GLfloat ∗values)

    float uniform setter auxiliary function
- void setUniform3fv (const std::string &name, GLsizei count, const GLfloat ∗values)

    float uniform setter auxiliary function
- void setUniform4fv (const std::string &name, GLsizei count, const GLfloat ∗values)

    float uniform setter auxiliary function
- void setUniform1d (const std::string &name, GLdouble value)

    double (GL 4.0+ only) uniform setter auxiliary function
- void setUniform1dv (const std::string &name, GLsizei count, const GLdouble ∗values)

    double (GL 4.0+ only) uniform setter auxiliary function
- void setUniform2dv (const std::string &name, GLsizei count, const GLdouble ∗values)

    double (GL 4.0+ only) uniform setter auxiliary function
- void setUniform3dv (const std::string &name, GLsizei count, const GLdouble ∗values)

    double (GL 4.0+ only) uniform setter auxiliary function
- void setUniform4dv (const std::string &name, GLsizei count, const GLdouble ∗values)

    double (GL 4.0+ only) uniform setter auxiliary function
- void setAttribute1i (const std::string &name, GLint value)

    generic vertex integer attribute setter auxiliary function
- void setAttribute1iv (const std::string &name, const GLint ∗values)

    generic vertex integer attribute setter auxiliary function
- void setAttribute2iv (const std::string &name, const GLint ∗values)

    generic vertex integer attribute setter auxiliary function
- void setAttribute3iv (const std::string &name, const GLint ∗values)

    generic vertex integer attribute setter auxiliary function
- void setAttribute4iv (const std::string &name, const GLint ∗values)

    generic vertex integer attribute setter auxiliary function
- void setAttribute1ui (const std::string &name, GLuint value)

    generic vertex unsigned integer attribute setter auxiliary function
- void setAttribute1uiv (const std::string &name, const GLuint ∗values)

    generic vertex integer attribute setter auxiliary function
- void setAttribute2uiv (const std::string &name, const GLuint ∗values)

    generic vertex integer attribute setter auxiliary function
- void setAttribute3uiv (const std::string &name, const GLuint ∗values)

    generic vertex integer attribute setter auxiliary function
- void setAttribute4uiv (const std::string &name, const GLuint ∗values)

    generic vertex integer attribute setter auxiliary function
- void setAttributeL1i64NV (const std::string &name, GLint64EXT value)

    generic vertex 64bit integer (GL 4.0+ only) attribute setter auxiliary function
- void setAttributeL1i64vNV (const std::string &name, const GLint64EXT ∗values)

    generic vertex 64bit integer (GL 4.0+ only) attribute setter auxiliary function
- void setAttributeL2i64vNV (const std::string &name, const GLint64EXT ∗values)

generic vertex 64bit integer (GL 4.0+ only) attribute setter auxiliary function
- void setAttributeL3i64vNV (const std::string &name, const GLint64EXT ∗values)

  generic vertex 64bit integer (GL 4.0+ only) attribute setter auxiliary function
- void setAttributeL4i64vNV (const std::string &name, const GLint64EXT ∗values)

  generic vertex 64bit integer (GL 4.0+ only) attribute setter auxiliary function
- void setAttributeL1ui64NV (const std::string &name, GLuint64EXT value)

  generic vertex 64bit unsigned integer (GL 4.0+ only) attribute setter auxiliary function
- void setAttributeL1ui64vNV (const std::string &name, const GLuint64EXT ∗values)

  generic vertex 64bit unsigned integer (GL 4.0+ only) attribute setter auxiliary function
- void setAttributeL2ui64vNV (const std::string &name, const GLuint64EXT ∗values)

  generic vertex 64bit unsigned integer (GL 4.0+ only) attribute setter auxiliary function
- void setAttributeL3ui64vNV (const std::string &name, const GLuint64EXT ∗values)

  generic vertex 64bit unsigned integer (GL 4.0+ only) attribute setter auxiliary function
- void setAttributeL4ui64vNV (const std::string &name, const GLuint64EXT ∗values)

  generic vertex 64bit unsigned integer (GL 4.0+ only) attribute setter auxiliary function
- void setAttributeP1ui (const std::string &name, GLenum type, GLboolean normalized, GLuint value)

  generic vertex unsigned integer packed attribute setter auxiliary function type parameter: type of packing used on the data.
- void setAttributeP1uiv (const std::string &name, GLenum type, GLboolean normalized, const GLuint ∗values)

  generic vertex unsigned integer packed attribute setter auxiliary function type parameter: type of packing used on the data.
- void setAttributeP2uiv (const std::string &name, GLenum type, GLboolean normalized, const GLuint ∗values)

  generic vertex unsigned integer packed attribute setter auxiliary function type parameter: type of packing used on the data.
- void setAttributeP3uiv (const std::string &name, GLenum type, GLboolean normalized, const GLuint ∗values)

  generic vertex unsigned integer packed attribute setter auxiliary function type parameter: type of packing used on the data.
- void setAttributeP4uiv (const std::string &name, GLenum type, GLboolean normalized, const GLuint ∗values)

  generic vertex unsigned integer packed attribute setter auxiliary function type parameter: type of packing used on the data.
- void setAttribute1f (const std::string &name, GLfloat value)

  generic vertex float attribute setter auxiliary function
- void setAttribute1fv (const std::string &name, const GLfloat ∗values)

  generic vertex float attribute setter auxiliary function
- void setAttribute2fv (const std::string &name, const GLfloat ∗values)

  generic vertex float attribute setter auxiliary function
- void setAttribute3fv (const std::string &name, const GLfloat ∗values)

  generic vertex float attribute setter auxiliary function
- void setAttribute4fv (const std::string &name, const GLfloat ∗values)

  generic vertex float attribute setter auxiliary function
- void setAttribute1d (const std::string &name, GLdouble value)

  generic vertex double (GL 4.0+ only) attribute setter auxiliary function
- void setAttribute1dv (const std::string &name, const GLdouble ∗values)

  generic vertex double (GL 4.0+ only) attribute setter auxiliary function
- void setAttribute2dv (const std::string &name, const GLdouble ∗values)

  generic vertex double (GL 4.0+ only) attribute setter auxiliary function
- void setAttribute3dv (const std::string &name, const GLdouble ∗values)

  generic vertex double (GL 4.0+ only) attribute setter auxiliary function
- void setAttribute4dv (const std::string &name, const GLdouble ∗values)

  generic vertex double (GL 4.0+ only) attribute setter auxiliary function
- virtual void initializeShaderProgram ()=0

  initialize shader program function to override

- void enableShaderProgram () const

  enable shader program
- void disableShaderProgram () const

  disable shader program
- **OpenGLShaderProgram** (const OpenGLShaderProgram &)=delete
- **OpenGLShaderProgram** (OpenGLShaderProgram &&)=delete
- OpenGLShaderProgram & **operator=** (const OpenGLShaderProgram &)=delete
- OpenGLShaderProgram & **operator=** (OpenGLShaderProgram &&)=delete

Protected Member Functions

- **OpenGLShaderProgram** (OpenGLDriverInfo ∗openGLDriverInfo, bool enableVertexProgramTwoSided↩Lighting=true) noexcept

Protected Attributes

- OpenGLDriverInfo ∗ **_openGLDriverInfo** = nullptr
- OpenGLShaderGLSLPreProcessorCommands ∗ **_openGLShaderGLSLPreProcessorCommands** = nullptr
- OpenGLShaderCompileAndLink ∗ **_openGLShaderCompileAndLink** = nullptr

Private Member Functions

- void createGLShaderProgram ()

  create shader program
- GLint getUniformLocation (const std::string &name)

  get uniform location
- GLint getAttributeLocation (const std::string &name)

  get attribute location
- void releaseGLShaderProgram () const

  release shader program

Private Attributes

- GLuint **_shaderProgram** = 0
- std::unordered_map< std::string, GLint > **_allUniformLocationsMap**
- std::unordered_map< std::string, GLint > **_allAttribLocationsMap**

5.63.1 Detailed Description

This abstract class encapsulates usage of a GLSL program.

To be inherited from usage-specific sub-classes.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.63.2 Member Function Documentation

5.63.2.1 setAttributeP1ui()

```
void OpenGLShaderProgram::setAttributeP1ui (
            const std::string & name,
            GLenum type,
            GLboolean normalized,
            GLuint value )
```

generic vertex unsigned integer packed attribute setter auxiliary function type parameter: type of packing used on the data.

This parameter must be GL_INT_10_10_10_2 or GL_UNSIGNED_INT_10_10_10_2 to specify signed or unsigned data, respectively normalized parameter: if GL_TRUE, then the values are to be converted to floating point values by normalizing. Otherwise, they are converted directly to floating point values

5.63.2.2 setAttributeP1uiv()

```
void OpenGLShaderProgram::setAttributeP1uiv (
            const std::string & name,
            GLenum type,
            GLboolean normalized,
            const GLuint * values )
```

generic vertex unsigned integer packed attribute setter auxiliary function type parameter: type of packing used on the data.

This parameter must be GL_INT_10_10_10_2 or GL_UNSIGNED_INT_10_10_10_2 to specify signed or unsigned data, respectively normalized parameter: if GL_TRUE, then the values are to be converted to floating point values by normalizing. Otherwise, they are converted directly to floating point values

5.63.2.3 setAttributeP2uiv()

```
void OpenGLShaderProgram::setAttributeP2uiv (
            const std::string & name,
            GLenum type,
            GLboolean normalized,
            const GLuint * values )
```

generic vertex unsigned integer packed attribute setter auxiliary function type parameter: type of packing used on the data.

This parameter must be GL_INT_10_10_10_2 or GL_UNSIGNED_INT_10_10_10_2 to specify signed or unsigned data, respectively normalized parameter: if GL_TRUE, then the values are to be converted to floating point values by normalizing. Otherwise, they are converted directly to floating point values

5.63.2.4 setAttributeP3uiv()

```
void OpenGLShaderProgram::setAttributeP3uiv (
            const std::string & name,
            GLenum type,
            GLboolean normalized,
            const GLuint * values )
```

generic vertex unsigned integer packed attribute setter auxiliary function type parameter: type of packing used on the data.

This parameter must be GL_INT_10_10_10_2 or GL_UNSIGNED_INT_10_10_10_2 to specify signed or unsigned data, respectively normalized parameter: if GL_TRUE, then the values are to be converted to floating point values by normalizing. Otherwise, they are converted directly to floating point values

5.63.2.5 setAttributeP4uiv()

```
void OpenGLShaderProgram::setAttributeP4uiv (
            const std::string & name,
            GLenum type,
            GLboolean normalized,
            const GLuint * values )
```

generic vertex unsigned integer packed attribute setter auxiliary function type parameter: type of packing used on the data.

This parameter must be GL_INT_10_10_10_2 or GL_UNSIGNED_INT_10_10_10_2 to specify signed or unsigned data, respectively normalized parameter: if GL_TRUE, then the values are to be converted to floating point values by normalizing. Otherwise, they are converted directly to floating point values

5.64 UtilsCUDA::OutputTypes Struct Reference

Usage of a C-style enum (not typesafe C++11 enum class) to be able to use a viz-style bitwise flag OR API on enum values.

```
#include <OutputTypes.h>
```

Public Types

- enum **OutputType** : std::uint32_t {
  WRITE_TO_NOTHING = (1 << 0), **WRITE_TO_CPU_MEMORY** = (1 << 1), **WRITE_TO_BINARY** = (1 << 2), **WRITE_TO_ZIP** = (1 << 3),
  WRITE_TO_TEXT = (1 << 4), **WRITE_TO_GPU0_MEMORY** = (1 << 5), **WRITE_TO_GPU1_MEMORY** = (1 << 6), **WRITE_TO_GPU2_MEMORY** = (1 << 7),
  WRITE_TO_GPU3_MEMORY = (1 << 8), **WRITE_TO_GPU4_MEMORY** = (1 << 9), **WRITE_TO_GPU5↩_MEMORY** = (1 << 10), **WRITE_TO_GPU6_MEMORY** = (1 << 11),
  WRITE_TO_GPU7_MEMORY = (1 << 12) }

Public Member Functions

- **OutputTypes** (const OutputTypes &)=delete
- **OutputTypes** (OutputTypes &&)=delete
- OutputTypes & **operator=** (const OutputTypes &)=delete
- OutputTypes & **operator=** (OutputTypes &&)=delete

5.64.1 Detailed Description

Usage of a C-style enum (not typesafe C++11 enum class) to be able to use a viz-style bitwise flag OR API on enum values.

OutputTypes.h:

Usage of a C-style enum (not typesafe C++11 enum class) to be able to use a viz-style bitwise flag OR API on enum values.

Author

Thanos Theo, 2018

Version

14.0.0.0

5.65 UtilsCUDA::PinnedDeleter< T > Struct Template Reference

Public Member Functions

- void **operator()** (T ∗ptr) noexcept

5.66 Utils::Randomizers::RandomRNGWELL512 Class Reference

The RandomRNGWELL512 class provides the very fast RNG WELL512 algorithm random number generator initialized with a random integer.

```
#include <Randomizers.h>
```

Public Member Functions

- std::uint64_t **getRandomInteger** ()
- double **getRandomFloat** ()
- double **operator()** ()
- **RandomRNGWELL512** (const RandomRNGWELL512 &)=delete
- **RandomRNGWELL512** (RandomRNGWELL512 &&)=delete
- RandomRNGWELL512 & **operator=** (const RandomRNGWELL512 &)=delete
- RandomRNGWELL512 & **operator=** (RandomRNGWELL512 &&)=delete

Static Public Member Functions

- static std::uint64_t **getRandomMax** ()

Private Attributes

- std::uint64_t **_index** = 0
- std::array< std::uint64_t, STATE_SIZE > **_state** { { 0 } }

Static Private Attributes

- static constexpr std::size_t **STATE_SIZE** = 16

5.66.1 Detailed Description

The RandomRNGWELL512 class provides the very fast RNG WELL512 algorithm random number generator initialized with a random integer.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.67 Utils::ReverseIterationWrapper< Container > Struct Template Reference

The ReverseIterationWrapper dummy struct provides additional generic functionality which std doesn't still provide.

```
#include <UtilityFunctions.h>
```

Public Attributes

- Container & **iterable**

5.67.1 Detailed Description

template<typename Container>
struct Utils::ReverseIterationWrapper< Container >

The ReverseIterationWrapper dummy struct provides additional generic functionality which std doesn't still provide.

Note that ReverseIterationWrapper with its related functions have to reside in namespace scope.

Usage: for (const auto& value : Utils::reverse(container))

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.68 OpenGLRenderingEngine::ShaderFilesGenerator Class Reference

This class includes shader files header/implementation generator related functionality.

```
#include <ShaderFilesGenerator.h>
```

Classes

- class Key

Public Member Functions

- **ShaderFilesGenerator** (const ShaderFilesGenerator &)=delete
- **ShaderFilesGenerator** (ShaderFilesGenerator &&)=delete
- ShaderFilesGenerator & **operator=** (const ShaderFilesGenerator &)=delete
- ShaderFilesGenerator & **operator=** (ShaderFilesGenerator &&)=delete
- void **generateAllShaderFilesCode** (const std::string &absolutePath)

Private Types

- using **BitsetType** = std::bitset< OpenGLAssetManager::NUMBER_OF_TOTAL_SHADER_TYPES >

Private Member Functions

- void readAllShaderFiles (const std::string &absolutePath, const std::string &pathName, const std::string &shaderName, int shaderType)

  core shader generator function
- void writeAllGLSLHeaderFilesForShaders (const std::string &absolutePath)

  core shader generator function
- void writeMainGLSLClass (const std::string &absolutePath)

  core shader generator function

Private Attributes

- std::map< Key, std::vector< std::list< std::string > > > _allShaderFiles

  standard map for saving shader file names sorted

5.68.1 Detailed Description

This class includes shader files header/implementation generator related functionality.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.69 Utils::UtilityFunctions::StdAuxiliaryFunctions Struct Reference

The StdAuxiliaryFunctions class provides additional generic functionality which std doesn't (currently) still provide.

```
#include <UtilityFunctions.h>
```

Public Member Functions

- **StdAuxiliaryFunctions** (const StdAuxiliaryFunctions &)=delete
- **StdAuxiliaryFunctions** (StdAuxiliaryFunctions &&)=delete
- StdAuxiliaryFunctions & **operator=** (const StdAuxiliaryFunctions &)=delete
- StdAuxiliaryFunctions & **operator=** (StdAuxiliaryFunctions &&)=delete

Static Public Member Functions

- template<typename T , std::size_t N>
  static constexpr std::size_t arraySize (T(&)[N]) noexcept

  Returns the size of an array as a compile-time constant (the array parameter has no name, because we care only about the number of elements it contains).
- template<typename E >
  static constexpr auto toUnsignedType (E enumerator) noexcept

  Returns the size_t value from a given enumerator.
- template<typename T , std::size_t N>
  static void insertionSort (T ∗__restrict arrayData)

  Sort an array using insertion sort with a constant small size of N.

5.69.1 Detailed Description

The StdAuxiliaryFunctions class provides additional generic functionality which std doesn't (currently) still provide.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.69.2 Member Function Documentation

5.69.2.1 insertionSort()

```
template<typename T , std::size_t N>
static void Utils::UtilityFunctions::StdAuxiliaryFunctions::insertionSort (
            T ∗__restrict arrayData )  [inline], [static]
```

Sort an array using insertion sort with a constant small size of N.

While some divide-and-conquer algorithms such as quicksort and mergesort outperform insertion sort for larger arrays, non-recursive sorting algorithms such as insertion sort or selection sort are generally faster for very small arrays (the exact size varies by environment and implementation, but is typically between seven and fifty elements). Therefore, a useful optimization in the implementation of those algorithms is a hybrid approach, using the simpler algorithm when the array has been divided to a small size.

5.70 Utils::UtilityFunctions::StdReadWriteFileFunctions Class Reference

The StdReadWriteFileFunctions class provides additional i/o functionality.

```
#include <UtilityFunctions.h>
```

Public Member Functions

- **StdReadWriteFileFunctions** (const StdReadWriteFileFunctions &)=delete
- **StdReadWriteFileFunctions** (StdReadWriteFileFunctions &&)=delete
- StdReadWriteFileFunctions & **operator=** (const StdReadWriteFileFunctions &)=delete
- StdReadWriteFileFunctions & **operator=** (StdReadWriteFileFunctions &&)=delete

Static Public Member Functions

- static bool assure (const std::ios &stream, const std::string &fullpathWithFileName)

  Checks if stream is open.
- static bool assure (std::size_t numberOfElements, const std::string &fullpathWithFileName)

  Checks if file is empty.
- static std::list< std::string > readTextFile (const std::string &fullpathWithFileName, bool trimString=true)

  Reads a text file into a list of line strings.
- static void writeTextFile (const std::string &fullpathWithFileName, const std::string &textToWrite, std::ios_↩
base::openmode mode=std::ios::out)

  Writes a text file with a given text.
- static bool pathExists (const std::string &fullpath)

  Checks if a given path exists using the C++17 <filesystem>.
- static std::size_t getFileSize (const std::string &fullpathWithFileName)

  Gets the file size of a given file using the C++17 <filesystem>.
- static std::string getCurrentPath ()

  Gets the current path using the C++17 <filesystem>.
- static bool removeFile (const std::string &fullpathWithFileName)

  Removes the given file using the C++17 <filesystem>.
- static bool removeAllFilesWithExtension (const std::string &fullpath, const std::string &fileExtension)

  Removes all files with given extension in given directory using the C++17 <filesystem>.
- static bool createDirectory (const std::string &fullpath)

  Creates the given directory using the C++17 <filesystem>.
- static std::uintmax_t removeDirectory (const std::string &fullpath)

  Removes the given directory with anything in it recursively using the C++17 <filesystem>.
- template<typename T >
static char ∗ asBytes (const T ∗obj)

  Cast T as bytes.
- template<typename T >
static T ∗ asObject (void ∗data)

  Cast void∗ as object T.
- template<typename T >
static bool writeBinaryFile (const std::string &fullpathWithFileName, const T ∗__restrict ptr, std::size_t array↩
Size)

  Write a binary file from the given T∗ pointer array.

- template<typename T >
  static bool writeZipFile (const std::string &fullpathWithFileName, const std::string &archiveName, const T ∗↩
  __restrict ptr, std::size_t arraySize)

  Write a zip file from the given T∗ pointer array.
- template<typename T >
  static bool readBinaryFile (const std::string &fullpathWithFileName, std::vector< T > &vec)

  Read the given binary file to an std::vector<T>.
- template<typename T >
  static std::tuple< bool, std::size_t > readBinaryFile (const std::string &fullpathWithFileName, std::unique_↩
  ptr< T[ ]> &ptr)

  Read the given binary file to an std::unique_ptr<T[ ]>.
- template<typename T >
  static bool readZipFile (const std::string &fullpathWithFileName, const std::string &archiveName, std::vector<
  T > &vec)

  Read the given zip file to an std::vector<T>.
- template<typename T >
  static std::tuple< bool, std::size_t > readZipFile (const std::string &fullpathWithFileName, const std::string
  &archiveName, std::unique_ptr< T[ ]> &ptr)

  Read the given zip file to an std::unique_ptr<T[ ]>.

Static Private Member Functions

- static bool zipAddMemoryToArchiveFileInPlace (const std::string &fullpathWithFileName, const std::string &archiveName, const void ∗bufferPtr, std::size_t bufferSize)

  zipAddMemoryToArchiveFileInPlace() efficiently (but not atomically) appends a memory blob to a ZIP archive.
- static bool zipExtractArchiveFileToHeap (const std::string &fullpathWithFileName, const std::string &archive↩
  Name, void ∗&data, std::size_t &dataSize)

  zipExtractArchiveFileToHeap() reads a single file from an archive into a heap block.

5.70.1 Detailed Description

The StdReadWriteFileFunctions class provides additional i/o functionality.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.70.2 Member Function Documentation

5.70.2.1 zipAddMemoryToArchiveFileInPlace()

```
bool StdReadWriteFileFunctions::zipAddMemoryToArchiveFileInPlace (
             const std::string & fullpathWithFileName,
             const std::string & archiveName,
             const void ∗ bufferPtr,
             std::size_t bufferSize )  [static], [private]
```

zipAddMemoryToArchiveFileInPlace() efficiently (but not atomically) appends a memory blob to a ZIP archive.

C++ wrapper encapsulation of the mz_zip_add_mem_to_archive_file_in_place() C library function.

5.70.2.2 zipExtractArchiveFileToHeap()

```
bool StdReadWriteFileFunctions::zipExtractArchiveFileToHeap (
            const std::string & fullpathWithFileName,
            const std::string & archiveName,
            void *& data,
            std::size_t & dataSize )  [static], [private]
```

zipExtractArchiveFileToHeap() reads a single file from an archive into a heap block.

Returns NULL on failure. C++ wrapper encapsulation of the mz_zip_extract_archive_file_to_heap() C library function.

5.71 Utils::UtilityFunctions::StringAuxiliaryFunctions Class Reference

The StringAuxiliaryFunctions class provides additional string functionality which std doesn't (currently) still provide.

`#include <UtilityFunctions.h>`

Public Member Functions

- **StringAuxiliaryFunctions** (const StringAuxiliaryFunctions &)=delete
- **StringAuxiliaryFunctions** (StringAuxiliaryFunctions &&)=delete
- StringAuxiliaryFunctions & **operator=** (const StringAuxiliaryFunctions &)=delete
- StringAuxiliaryFunctions & **operator=** (StringAuxiliaryFunctions &&)=delete

Static Public Member Functions

- template<typename T >
  static std::string toString (const bool value, std::enable_if_t< std::is_same< T, bool >::value > *=nullptr)

  String manipulation auxiliary function (bool version).
- template<typename T >
  static std::string toString (const T value, std::enable_if_t<!std::is_same< T, bool >::value &&std::is_integral< T >::value &&std::is_signed< T >::value > *=nullptr)

  String manipulation auxiliary function (integral signed version).
- template<typename T >
  static std::string toString (const T value, std::enable_if_t<!std::is_same< T, bool >::value &&std::is_integral< T >::value &&std::is_unsigned< T >::value > *=nullptr)

  String manipulation auxiliary function (integral unsigned version).
- template<typename T >
  static std::string toString (const T value, std::enable_if_t< std::is_floating_point< T >::value > *=nullptr)

  String manipulation auxiliary function (float/double version).
- template<typename T >
  static std::string toString (const T &value, std::enable_if_t<!std::is_arithmetic< T >::value &&std::is_same< T, std::string >::value > *=nullptr)

  String manipulation auxiliary function (T 'as a string' version).
- template<typename T >
  static std::string toString (const T &value, std::enable_if_t<!std::is_arithmetic< T >::value &&!std::is_same< T, std::string >::value > *=nullptr)

  String manipulation auxiliary function (T 'as a generic writable object' version).

- template<typename T >
  static T fromString (const std::string &str, std::enable_if_t< std::is_same< T, bool >::value > ∗=nullptr)

  String manipulation auxiliary function (bool version).
- template<typename T >
  static T fromString (const std::string &str, std::enable_if_t<!std::is_same< T, bool >::value &&std::is_↩ integral< T >::value &&std::is_signed< T >::value > ∗=nullptr)

  String manipulation auxiliary function (integral signed version).
- template<typename T >
  static T fromString (const std::string &str, std::enable_if_t<!std::is_same< T, bool >::value &&std::is_↩ integral< T >::value &&std::is_unsigned< T >::value > ∗=nullptr)

  String manipulation auxiliary function (integral unsigned version).
- template<typename T >
  static T fromString (const std::string &str, std::enable_if_t< std::is_floating_point< T >::value > ∗=nullptr)

  String manipulation auxiliary function (float/double version).
- template<typename T >
  static T fromString (const std::string &str, std::enable_if_t<!std::is_arithmetic< T >::value &&std::is_same< T, std::string >::value > ∗=nullptr)

  String manipulation auxiliary function (T 'as a string' version).
- template<typename T >
  static T fromString (const std::string &str, std::enable_if_t<!std::is_arithmetic< T >::value &&!std::is_same< T, std::string >::value > ∗=nullptr)

  String manipulation auxiliary function (T 'as a generic writable object' version).
- static bool startsWith (const std::string &str, const std::string &starting)

  String manipulation auxiliary function.
- static bool endsWith (const std::string &str, const std::string &ending)

  String manipulation auxiliary function.
- static std::string trimLeft (const std::string &str)

  String manipulation auxiliary function.
- static std::string trimRight (const std::string &str)

  String manipulation auxiliary function.
- static std::string trim (const std::string &str)

  String manipulation auxiliary function.
- static std::string toUpperCase (const std::string &str)

  String manipulation auxiliary function.
- static std::string toLowerCase (const std::string &str)

  String manipulation auxiliary function.
- static std::string formatNumberString (std::size_t number, std::size_t totalNumbers)

  String manipulation auxiliary function.
- template<typename Container >
  static Container tokenize (const std::string &str, const std::string &delimiters=" ")

  String manipulation auxiliary function.

Static Private Member Functions

- template<typename T >
  static std::string parseNumberCStyle (const char ∗format, const T value)

  String manipulation auxiliary function.

Static Private Attributes

- static constexpr std::size_t **STRING_BUFFER_SIZE** = 64

5.71.1 Detailed Description

The StringAuxiliaryFunctions class provides additional string functionality which std doesn't (currently) still provide.

Author

Thanos Theo, 2009-2018

Version

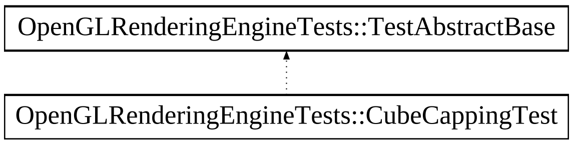
14.0.0.0

5.72 OpenGLRenderingEngineTests::TestAbstractBase Class Reference

TestAbstractBase is the abstract base class for all GLUT tests.

```
#include <TestAbstractBase.h>
```

Inheritance diagram for OpenGLRenderingEngineTests::TestAbstractBase:


OpenGLRenderingEngineTests::TestAbstractBase
OpenGLRenderingEngineTests::CubeCappingTest

Public Member Functions

- **TestAbstractBase** (const TestAbstractBase &)=delete
- **TestAbstractBase** (TestAbstractBase &&)=delete
- TestAbstractBase & **operator=** (const TestAbstractBase &)=delete
- TestAbstractBase & **operator=** (TestAbstractBase &&)=delete

Protected Member Functions

- void **writeScreenshotToFile** () const
- void **releaseAllGLResources** ()
- **TestAbstractBase** (int thisScreenWidth, int thisScreenHeight, const std::string &thisModelFileName, const std::string &thisModelLoaderDescriptorFileName, bool thisMultisample)
- **TestAbstractBase** (int thisScreenWidth, int thisScreenHeight, const std::string &thisTextureFileName, const std::string &thisModelFileName, const std::string &thisModelLoaderDescriptorFileName, bool this↩ Multisample)

Protected Attributes

- OpenGLRenderingEngine::OpenGLDriverInfo ∗ **openGLDriverInfo** = nullptr
- OpenGLRenderingEngine::OpenGLEulerCamera ∗ **openGLEulerCamera** = nullptr
- int **screenWidth** = 0
- int **screenHeight** = 0
- int **vsynch** = 1
- int **autoRotate** = 0
- bool **blackOrWhiteBackground** = true
- int **wireframe** = 0
- bool **useMotionBlurForScene** = false
- float **motionBlurSize** = 0.6f
- bool **reInitMotionBlurForScene** = false
- int **useUIInformation** = 1
- GLuint **useUIInformationDisplayList** = 0
- bool **reInitUIInformation** = true
- bool **useFXAA_Antialias** = false
- bool **takeScreenshot** = false
- bool **mouseLeftDown** = false
- bool **mouseMiddleDown** = false
- bool **mouseRightDown** = false
- double **mouseX** = 0.0
- double **mouseY** = 0.0
- double **cameraAngleX** = 0.0
- double **cameraAngleY** = 0.0
- double **cameraDistanceX** = 0.0
- double **cameraDistanceY** = 0.0
- double **cameraDistanceZ** = 15.0
- Utils::Randomizers::RandomRNGWELL512 **random**
- Utils::AccurateTimers::AccurateCPUTimer **timer**
- int **fpsCounter** = 59
- std::string **fpsString** = ""
- std::string **textureFileName** = ""
- std::string **modelFileName** = ""
- std::string **modelLoaderDescriptorFileName** = ""
- bool **multisample** = false
- GLfloat **shaderTimer** = 0.0f

Static Protected Attributes

- static const std::size_t **GLUT_TEXT_WIDTH** = 9
- static const std::size_t **GLUT_TEXT_HEIGHT** = 15
- static const std::size_t **NUMBER_OF_LIGHTS** = 2
- static const bool **USE_COLOR_MATERIAL** = true
- static const std::size_t **ENVIRONMENT_MAPPING_RATIO_FACTOR** = 1
- static const std::size_t **DEPTH_STENCIL_RATIO_FACTOR** = 1
- static const std::size_t **DEPTH_STENCIL_MULTIPLICATION_FACTOR** = 2
- static const std::size_t **FULLSCREEN_MAPPING_RATIO_FACTOR** = 1
- static const std::size_t **A_BUFFER_3D_MAX_SIZE** = 256
- static const GLuint **ACTIVE_TEXTURE_UNIT_FOR_2D_TEXTURE** = 0
- static const GLuint **ACTIVE_TEXTURE_UNIT_FOR_BLUR_XY_TEXTURE** = 1
- static const GLuint **ACTIVE_TEXTURE_UNIT_FOR_PERLIN_NOISE_3D_TEXTURE** = 2
- static const GLuint **ACTIVE_TEXTURE_UNIT_FOR_A_BUFFER_3D_COUNTER** = 3
- static const GLuint **ACTIVE_TEXTURE_UNIT_FOR_A_BUFFER_3D** = 4

- static const GLuint **ACTIVE_TEXTURE_UNIT_FOR_A_BUFFER_3D_LINKED_LIST_ATOMIC_COUNTER** = 5
- static const GLuint **ACTIVE_TEXTURE_UNIT_FOR_A_BUFFER_3D_LINKED_LIST_OFFSET** = 6
- static const GLuint **ACTIVE_TEXTURE_UNIT_FOR_A_BUFFER_3D_LINKED_LIST** = 7
- static const bool **DEPTH_OF_FIELD_DEBUG_MODE** = false
- static const GLuint **NORMAL_SHADING_BUFFER_ELEMENTS** = 3
- static const GLuint **NORMAL_SHADING_LINKED_LIST_BUFFER_ELEMENTS** = 4
- static const GLuint **DEFERRED_SHADING_BUFFER_ELEMENTS** = 9
- static const GLuint **DEFERRED_SHADING_LINKED_LIST_BUFFER_ELEMENTS** = 10

Private Member Functions

- void **performAllGLInitializations** ()

5.72.1 Detailed Description

TestAbstractBase is the abstract base class for all GLUT tests.

Its constructors & virtual destructor are protected for this reason.

Author

Thanos Theo, 2009-2018

Version

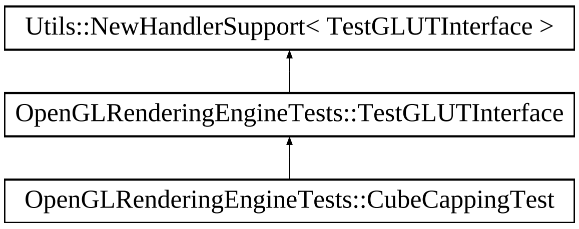
14.0.0.0

5.73 OpenGLRenderingEngineTests::TestGLUTInterface Struct Reference

TestGLUTInterface is the interface (pure abstract class) for all GLUT tests (FreeGlut pure virtual void function to be implemented in sub-classes).

```
#include <TestGLUTInterface.h>
```

Inheritance diagram for OpenGLRenderingEngineTests::TestGLUTInterface:


Utils::NewHandlerSupport< TestGLUTInterface >
OpenGLRenderingEngineTests::TestGLUTInterface
OpenGLRenderingEngineTests::CubeCappingTest

Public Member Functions

- virtual void **renderScene** ()=0
- virtual void **changeSize** (int w, int h)=0
- virtual void **keyboard** (unsigned char key, int x, int y)=0
- virtual void **specialKeysKeyboard** (int key, int x, int y)=0
- virtual void **mouse** (int button, int state, int x, int y)=0
- virtual void **mouseMotion** (int x, int y)=0
- virtual void **closeFunc** ()=0
- **TestGLUTInterface** (const TestGLUTInterface &)=delete
- **TestGLUTInterface** (TestGLUTInterface &&)=delete
- TestGLUTInterface & **operator=** (const TestGLUTInterface &)=delete
- TestGLUTInterface & **operator=** (TestGLUTInterface &&)=delete

Additional Inherited Members

5.73.1 Detailed Description

TestGLUTInterface is the interface (pure abstract class) for all GLUT tests (FreeGlut pure virtual void function to be implemented in sub-classes).

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.74 Utils::CPUParallelism::ThreadBarrier Class Reference

This class encapsulates usage of a thread barrier.

```
#include <ThreadBarrier.h>
```

Public Member Functions

- **ThreadBarrier** (std::size_t threadCount) noexcept
- bool **wait** ()
- **ThreadBarrier** (const ThreadBarrier &)=delete
- **ThreadBarrier** (ThreadBarrier &&)=delete
- ThreadBarrier & **operator=** (const ThreadBarrier &)=delete
- ThreadBarrier & **operator=** (ThreadBarrier &&)=delete

Private Attributes

- std::mutex **_mutex**
- std::condition_variable **_conditionVariable**
- std::size_t **_threshold** = 0
- std::size_t **_threadCount** = 0
- std::size_t **_generation** = 0

5.74.1 Detailed Description

This class encapsulates usage of a thread barrier.

ThreadBarrier.h:

This class encapsulates usage of a thread barrier.
CPUParallelism libraries originally based on with further extensions: `http://www.manning.com/williams/`.
The N-CP idea was based on: `http://www.biolayout.org/wp-content/uploads/2013/01/↩ Manuscript.pdf`.
Further inspiration was found here: `http://jcip.net.s3-website-us-east-1.amazonaws.com/`.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.75 Utils::CPUParallelism::ThreadGuard Class Reference

This class encapsulates usage of a thread guard using std::move() & the RAII C++ idiom.

```
#include <ThreadGuard.h>
```

Public Types

- enum **DestructorAction** : std::size_t { **JOIN**, **DETACH** }

Public Member Functions

- **ThreadGuard** (std::thread &&thread, DestructorAction action=DestructorAction::JOIN) noexcept
- **ThreadGuard** (const ThreadGuard &)=delete
- **ThreadGuard** (ThreadGuard &&)=delete
- ThreadGuard & **operator=** (const ThreadGuard &)=delete
- ThreadGuard & **operator=** (ThreadGuard &&)=delete
- std::thread & **get** ()

Private Attributes

- DestructorAction **_action** = DestructorAction::JOIN
- std::thread **_thread**

5.75.1 Detailed Description

This class encapsulates usage of a thread guard using std::move() & the RAII C++ idiom.

ThreadGuard.h:

This class encapsulates usage of a thread guard using std::move() & the RAII C++ idiom.
CPUParallelism libraries originally based on with further extensions: `http://www.manning.com/williams/`.
The N-CP idea was based on: `http://www.biolayout.org/wp-content/uploads/2013/01/↵ Manuscript.pdf`.
Further inspiration was found here: `http://jcip.net.s3-website-us-east-1.amazonaws.com/`.
This class also derives its inspiration from Scott Meyers C++11/14 book.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.76 Utils::CPUParallelism::ThreadJoiner Class Reference

This class encapsulates usage of a vector<thread> joiner using the RAII C++ idiom.

```
#include <ThreadJoiner.h>
```

Public Member Functions

- **ThreadJoiner** (std::thread ∗__restrict threads, std::size_t numberOfThreads) noexcept
- **ThreadJoiner** (const ThreadJoiner &)=delete
- **ThreadJoiner** (ThreadJoiner &&)=delete
- ThreadJoiner & **operator=** (const ThreadJoiner &)=delete
- ThreadJoiner & **operator=** (ThreadJoiner &&)=delete

Private Attributes

- std::thread ∗__restrict **_threads** = nullptr
- std::size_t **_numberOfThreads** = 0

5.76.1 Detailed Description

This class encapsulates usage of a vector<thread> joiner using the RAII C++ idiom.

ThreadJoiner.h:

This class encapsulates usage of a vector<thread> joiner using the RAII C++ idiom.
CPUParallelism libraries originally based on with further extensions: http://www.manning.com/williams/.
The N-CP idea was based on: http://www.biolayout.org/wp-content/uploads/2013/01/↩
Manuscript.pdf.
Further inspiration was found here: http://jcip.net.s3-website-us-east-1.amazonaws.com/.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.77 Utils::CPUParallelism::ThreadPool Class Reference

This class encapsulates usage of a thread pool.

```
#include <ThreadPool.h>
```

Public Member Functions

- **ThreadPool** (std::size_t numberOfThreads=numberOfHardwareThreads()) noexcept
- template<typename FunctionType >
  void **submit** (FunctionType function)
- void **runPendingTask** ()
- **ThreadPool** (const ThreadPool &)=delete
- **ThreadPool** (ThreadPool &&)=delete
- ThreadPool & **operator=** (const ThreadPool &)=delete
- ThreadPool & **operator=** (ThreadPool &&)=delete

Private Member Functions

- void **workerThread** ()

Private Attributes

- std::unique_ptr< std::thread[ ]> **_threads** = nullptr
- std::size_t **_numberOfThreads** = 0
- ThreadJoiner **_joiner** = { nullptr, 0 }
- std::atomic< bool > **_done** = { false }
- ConcurrentBlockingQueue< std::function< void()> > **_workQueue**

5.77.1 Detailed Description

This class encapsulates usage of a thread pool.

ThreadPool.h:

This class encapsulates usage of a thread pool.
CPUParallelism libraries originally based on with further extensions: `http://www.manning.com/williams/`.
The N-CP idea was based on: `http://www.biolayout.org/wp-content/uploads/2013/01/↩ Manuscript.pdf`.
Further inspiration was found here: `http://jcip.net.s3-website-us-east-1.amazonaws.com/`.

Author

Thanos Theo, 2009-2018

Version

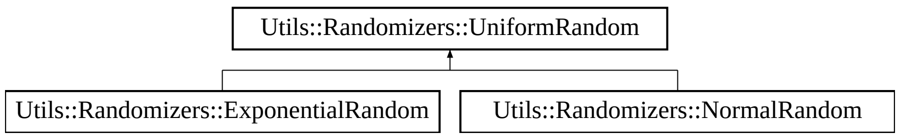
14.0.0.0

5.78 Utils::Randomizers::UniformRandom Class Reference

The UniformRandom class provides a uniform random number generator.

```
#include <Randomizers.h>
```

Inheritance diagram for Utils::Randomizers::UniformRandom:


Utils::Randomizers::UniformRandom
Utils::Randomizers::ExponentialRandom
Utils::Randomizers::NormalRandom

Public Member Functions

- std::uint64_t **getUniformInteger** ()
- double **getUniformFloat** ()
- double **operator()** ()
- void **setSeed** (std::uint64_t value=5489U)
- **UniformRandom** (const UniformRandom &)=delete
- **UniformRandom** (UniformRandom &&)=delete
- UniformRandom & **operator=** (const UniformRandom &)=delete
- UniformRandom & **operator=** (UniformRandom &&)=delete

Protected Attributes

- std::mt19937_64 **_rng**

Private Attributes

- std::uniform_int_distribution< std::uint64_t > **_uniformIntegerDistribution**
- std::uniform_real_distribution< double > **_uniformRealDistribution**

5.78.1 Detailed Description

The UniformRandom class provides a uniform random number generator.

Author

Thanos Theo, 2009-2018

Version

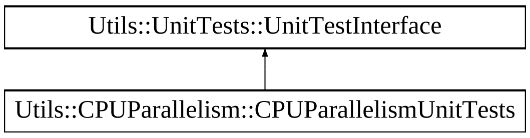
14.0.0.0

5.79 Utils::UnitTests::UnitTestInterface Struct Reference

The UnitTestInterface struct encapsulate a basic unit test interface.

```
#include <UnitTests.h>
```

Inheritance diagram for Utils::UnitTests::UnitTestInterface:


Utils::UnitTests::UnitTestInterface
Utils::CPUParallelism::CPUParallelismUnitTests

Public Member Functions

- virtual void **resetTests** ()=0
- virtual bool **conductTests** ()=0
- virtual void **reportTestResults** () const =0
- **UnitTestInterface** (const UnitTestInterface &)=delete
- **UnitTestInterface** (UnitTestInterface &&)=delete
- UnitTestInterface & **operator=** (const UnitTestInterface &)=delete
- UnitTestInterface & **operator=** (UnitTestInterface &&)=delete

5.79.1 Detailed Description

The UnitTestInterface struct encapsulate a basic unit test interface.

Author

Thanos Theo, 2018

Version

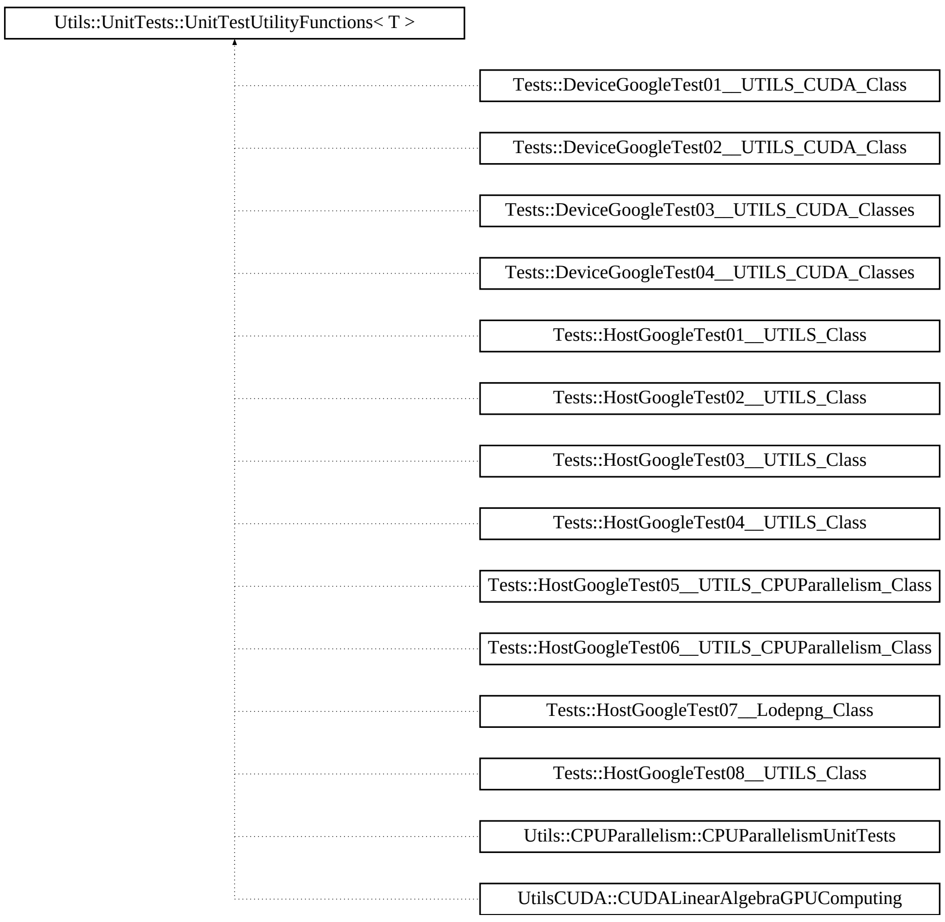
14.0.0.0

5.80 Utils::UnitTests::UnitTestUtilityFunctions< T > Class Template Reference

The UnitTestUtilityFunctions class adds unit testing utility function support through private inheritance.

```
#include <UnitTests.h>
```

Inheritance diagram for Utils::UnitTests::UnitTestUtilityFunctions< T >:


Utils::UnitTests::UnitTestUtilityFunctions< T >
Tests::DeviceGoogleTest01__UTILS_CUDA_Class
Tests::DeviceGoogleTest02__UTILS_CUDA_Class
Tests::DeviceGoogleTest03__UTILS_CUDA_Classes
Tests::DeviceGoogleTest04__UTILS_CUDA_Classes
Tests::HostGoogleTest01__UTILS_Class
Tests::HostGoogleTest02__UTILS_Class
Tests::HostGoogleTest03__UTILS_Class
Tests::HostGoogleTest04__UTILS_Class
Tests::HostGoogleTest05__UTILS_CPUParallelism_Class
Tests::HostGoogleTest06__UTILS_CPUParallelism_Class
Tests::HostGoogleTest07__Lodepng_Class
Tests::HostGoogleTest08__UTILS_Class
Utils::CPUParallelism::CPUParallelismUnitTests
UtilsCUDA::CUDALinearAlgebraGPUComputing

Public Member Functions

- **UnitTestUtilityFunctions** (const UnitTestUtilityFunctions &)=delete
- **UnitTestUtilityFunctions** (UnitTestUtilityFunctions &&)=delete
- UnitTestUtilityFunctions & **operator=** (const UnitTestUtilityFunctions &)=delete
- UnitTestUtilityFunctions & **operator=** (UnitTestUtilityFunctions &&)=delete

Static Public Member Functions

- static T **delta** (T a, T b)
- static bool **checkAbsoluteError** (T a, T b, T epsilon=getDefaultEpsilon())
- static bool **checkRelativeError** (T a, T b, T epsilon=getDefaultEpsilon())
- static bool **checkComplexAbsoluteError** (std::complex< T > a, std::complex< T > b, T epsilon=get↩ DefaultEpsilon())
- static bool **checkComplexRelativeError** (std::complex< T > a, std::complex< T > b, T epsilon=getDefault↩ Epsilon())

- template<typename I , typename W >
  static bool checkComplexRootMeanSquareError (const I *__restrict arrayA, const W *__restrict arrayB, std↩
  ::size_t arraySize, T epsilon=getDefaultEpsilon(), typename std::enable_if<!std::is_floating_point< I >::value &&!std::is_floating_point< W >::value >::type *=nullptr)

  Note: Template types I & W should not be decimals but only complex numbers of types std::complex, fftw & cufft.
- template<typename I , typename W >
  static bool checkComplexTwoNormError (const I *__restrict arrayA, const W *__restrict arrayB, std::size↩
  _t arraySize, T epsilon=getDefaultEpsilon(), typename std::enable_if<!std::is_floating_point< I >::value &&!std::is_floating_point< W >::value >::type *=nullptr)

  Note: Template types I & W should not be decimals but only complex numbers of types std::complex, fftw & cufft.
- static std::tuple< bool, std::string > **checkSeriesError** (const T *__restrict arrayA, const T *__restrict arrayB, std::size_t arraySize, bool frequencyData=false, T epsilonTime=getDefaultEpsilonTime(), T epsilon↩
  Frequency=getDefaultEpsilonFrequency())
- template<typename I , typename W >
  static std::tuple< bool, std::string > checkSeriesError (const I *__restrict arrayA, const W *__restrict arrayB, std::size_t arraySize, bool frequencyData=false, T epsilonTime=getDefaultEpsilonTime(), T epsilon↩
  Frequency=getDefaultEpsilonFrequency(), typename std::enable_if<!std::is_floating_point< I >::value &&!std::is_floating_point< W >::value >::type *=nullptr)

  Note: Template types I & W should not be decimals but only complex numbers of types std::complex, fftw & cufft.
- template<typename I , typename W >
  static std::tuple< bool, std::string > verifyComplexArraysAbsoluteError (const std::string &arrayAName, const I *__restrict arrayA, std::size_t arrayASize, const std::string &arrayBName, const W *__restrict arrayB, std↩
  ::size_t arrayBSize, T epsilon=getDefaultEpsilon(), typename std::enable_if<!std::is_floating_point< I >↩
  ::value &&!std::is_floating_point< W >::value >::type *=nullptr)

  Note: Template types I & W should not be decimals but only complex numbers of types std::complex, fftw & cufft.
- template<typename I , typename W >
  static std::tuple< bool, std::string > verifyComplexArraysRelativeError (const std::string &arrayAName, const I *__restrict arrayA, std::size_t arrayASize, const std::string &arrayBName, const W *__restrict arrayB, std↩
  ::size_t arrayBSize, T epsilon=getDefaultEpsilon(), typename std::enable_if<!std::is_floating_point< I >↩
  ::value &&!std::is_floating_point< W >::value >::type *=nullptr)

  Note: Template types I & W should not be decimals but only complex numbers of types std::complex, fftw & cufft.
- static void **parseComplexArrayFromTextRowMajor** (const std::list< std::string > &dataLines, std↩
  ::complex< T > *__restrict complexArray, std::uint32_t dataSize)
- static void **parseComplexArrayFromTextColumnMajor** (const std::list< std::string > &dataLines, std↩
  ::complex< T > *__restrict complexArray)

Static Protected Member Functions

- static T **getDefaultEpsilon** ()
- static T **getDefaultEpsilonTime** ()
- static T **getDefaultEpsilonFrequency** ()

5.80.1 Detailed Description

template<typename T>
class Utils::UnitTests::UnitTestUtilityFunctions< T >

The UnitTestUtilityFunctions class adds unit testing utility function support through private inheritance.

Author

Thanos Theo, 2018

Version

14.0.0.0

5.80.2 Member Function Documentation

5.80.2.1 checkComplexRootMeanSquareError()

```
template<typename T >
template<typename I , typename W >
static bool Utils::UnitTests::UnitTestUtilityFunctions< T >::checkComplexRootMeanSquareError (
              const I *__restrict arrayA,
              const W *__restrict arrayB,
              std::size_t arraySize,
              T epsilon = getDefaultEpsilon(),
              typename std::enable_if<!std::is_floating_point< I >::value &&!std::is_floating↩
_point< W >::value >::type *  = nullptr )  [inline], [static]
```

Note: Template types I & W should not be decimals but only complex numbers of types std::complex, fftw & cufft.

These template I & W types are not checked with template metaprogramming (besides if not decimal), so as to avoid dependencies to non-std complex numbers structs (fftw & cufft) in the Utils component.

5.80.2.2 checkComplexTwoNormError()

```
template<typename T >
template<typename I , typename W >
static bool Utils::UnitTests::UnitTestUtilityFunctions< T >::checkComplexTwoNormError (
              const I *__restrict arrayA,
              const W *__restrict arrayB,
              std::size_t arraySize,
              T epsilon = getDefaultEpsilon(),
              typename std::enable_if<!std::is_floating_point< I >::value &&!std::is_floating↩
_point< W >::value >::type *  = nullptr )  [inline], [static]
```

Note: Template types I & W should not be decimals but only complex numbers of types std::complex, fftw & cufft.

These template I & W types are not checked with template metaprogramming (besides if not decimal), so as to avoid dependencies to non-std complex numbers structs (fftw & cufft) in the Utils component.

5.80.2.3 checkSeriesError()

```
template<typename T >
template<typename I , typename W >
static std::tuple<bool, std::string> Utils::UnitTests::UnitTestUtilityFunctions< T >::check↩
SeriesError (
              const I *__restrict arrayA,
              const W *__restrict arrayB,
              std::size_t arraySize,
              bool frequencyData = false,
              T epsilonTime = getDefaultEpsilonTime(),
              T epsilonFrequency = getDefaultEpsilonFrequency(),
              typename std::enable_if<!std::is_floating_point< I >::value &&!std::is_floating↩
_point< W >::value >::type *  = nullptr )  [inline], [static]
```

Note: Template types I & W should not be decimals but only complex numbers of types std::complex, fftw & cufft.

These template I & W types are not checked with template metaprogramming (besides if not decimal), so as to avoid dependencies to non-std complex numbers structs (fftw & cufft) in the Utils component.

5.80.2.4 verifyComplexArraysAbsoluteError()

```
template<typename T >
template<typename I , typename W >
static std::tuple<bool, std::string> Utils::UnitTests::UnitTestUtilityFunctions< T >::verify↩
ComplexArraysAbsoluteError (
             const std::string & arrayAName,
             const I *__restrict arrayA,
             std::size_t arrayASize,
             const std::string & arrayBName,
             const W *__restrict arrayB,
             std::size_t arrayBSize,
             T epsilon = getDefaultEpsilon(),
             typename std::enable_if<!std::is_floating_point< I >::value &&!std::is_floating↩
_point< W >::value >::type * = nullptr ) [inline], [static]
```

Note: Template types I & W should not be decimals but only complex numbers of types std::complex, fftw & cufft.

These template I & W types are not checked with template metaprogramming (besides if not decimal), so as to avoid dependencies to non-std complex numbers structs (fftw & cufft) in the Utils component.

5.80.2.5 verifyComplexArraysRelativeError()

```
template<typename T >
template<typename I , typename W >
static std::tuple<bool, std::string> Utils::UnitTests::UnitTestUtilityFunctions< T >::verify↩
ComplexArraysRelativeError (
             const std::string & arrayAName,
             const I *__restrict arrayA,
             std::size_t arrayASize,
             const std::string & arrayBName,
             const W *__restrict arrayB,
             std::size_t arrayBSize,
             T epsilon = getDefaultEpsilon(),
             typename std::enable_if<!std::is_floating_point< I >::value &&!std::is_floating↩
_point< W >::value >::type * = nullptr ) [inline], [static]
```

Note: Template types I & W should not be decimals but only complex numbers of types std::complex, fftw & cufft.

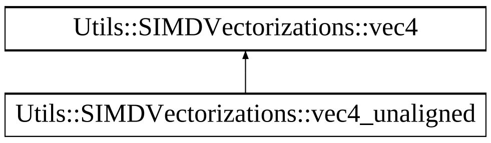
These template I & W types are not checked with template metaprogramming (besides if not decimal), so as to avoid dependencies to non-std complex numbers structs (fftw & cufft) in the Utils component.

5.81 Utils::SIMDVectorizations::vec4 Class Reference

The vec4 class is the main SIMD float4 class using the GLSL nomenclature.

```
#include <SIMDVectorizations.h>
```

Inheritance diagram for Utils::SIMDVectorizations::vec4:


Utils::SIMDVectorizations::vec4
Utils::SIMDVectorizations::vec4_unaligned

Public Member Functions

- **vec4** (__m128 value)
- **vec4** (const float ∗__restrict src)
- **vec4** (float x)
- **vec4** (const vec4 &)=default
- vec4 & **operator=** (const vec4 &)=default
- vec4 **operator+** (const vec4 &rhs) const
- vec4 **operator-** (const vec4 &rhs) const
- vec4 **operator∗** (const vec4 &rhs) const
- vec4 **operator/** (const vec4 &rhs) const
- vec4 **operator &** (const vec4 &rhs) const
- vec4 **operator|** (const vec4 &rhs) const
- vec4 **operator^** (const vec4 &rhs) const
- vec4 **operator==** (const vec4 &rhs) const
- vec4 **operator!=** (const vec4 &rhs) const
- vec4 **operator<** (const vec4 &rhs) const
- vec4 **operator<=** (const vec4 &rhs) const
- vec4 **operator>** (const vec4 &rhs) const
- vec4 **operator>=** (const vec4 &rhs) const
- float & **operator[ ]** (int index)
- float **operator[ ]** (int index) const
- not_vec4 **operator∼** () const
- bool **if_any_not_true** () const
- __m128 **get** () const
- float ∗ **store** (float ∗__restrict ptr) const
- float ∗ **store_unaligned** (float ∗__restrict ptr) const
- vec4 **if_then_else** (const vec4 &then, const vec4 &else_part) const

Private Attributes

- __m128 **_v**

Friends

- std::ostream & **operator<<** (std::ostream &o, const vec4 &y)
- vec4 **operator &** (const vec4 &lhs, const not_vec4 &rhs)
- vec4 **operator &** (const not_vec4 &lhs, const vec4 &rhs)

5.81.1 Detailed Description

The vec4 class is the main SIMD float4 class using the GLSL nomenclature.

Author

Thanos Theo, 2009-2018

Version

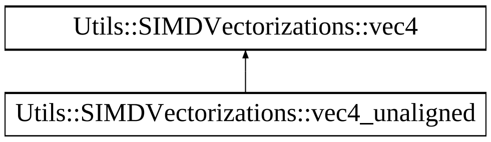
14.0.0.0

5.82 Utils::SIMDVectorizations::vec4_unaligned Class Reference

The vec4_unaligned class is the main unaligned SIMD float4 class using the GLSL nomenclature.

```
#include <SIMDVectorizations.h>
```

Inheritance diagram for Utils::SIMDVectorizations::vec4_unaligned:


Utils::SIMDVectorizations::vec4
Utils::SIMDVectorizations::vec4_unaligned

Public Member Functions

- **vec4_unaligned** (const float *__restrict src)

5.82.1 Detailed Description

The vec4_unaligned class is the main unaligned SIMD float4 class using the GLSL nomenclature.

Author

Thanos Theo, 2009-2018

Version

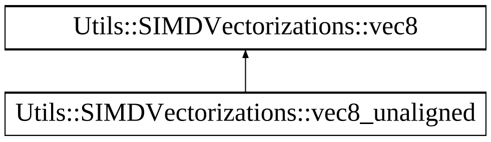
14.0.0.0

5.83 Utils::SIMDVectorizations::vec8 Class Reference

The vec8 class is the main SIMD float8 class using the GLSL nomenclature.

```
#include <SIMDVectorizations.h>
```

Inheritance diagram for Utils::SIMDVectorizations::vec8:


Utils::SIMDVectorizations::vec8
Utils::SIMDVectorizations::vec8_unaligned

Public Member Functions

- **vec8** (__m256 value)
- **vec8** (const float ∗__restrict src)
- **vec8** (float x)
- **vec8** (const vec8 &)=default
- vec8 & **operator=** (const vec8 &)=default
- vec8 **operator+** (const vec8 &rhs) const
- vec8 **operator-** (const vec8 &rhs) const
- vec8 **operator∗** (const vec8 &rhs) const
- vec8 **operator/** (const vec8 &rhs) const
- vec8 **operator &** (const vec8 &rhs) const
- vec8 **operator|** (const vec8 &rhs) const
- vec8 **operator^** (const vec8 &rhs) const
- vec8 **operator==** (const vec8 &rhs) const
- vec8 **operator!=** (const vec8 &rhs) const
- vec8 **operator<** (const vec8 &rhs) const
- vec8 **operator<=** (const vec8 &rhs) const
- vec8 **operator>** (const vec8 &rhs) const
- vec8 **operator>=** (const vec8 &rhs) const
- float & **operator[ ]** (int index)
- float **operator[ ]** (int index) const
- not_vec8 **operator∼** () const
- bool **if_any_not_true** () const
- __m256 **get** () const
- float ∗ **store** (float ∗__restrict ptr) const
- float ∗ **store_unaligned** (float ∗__restrict ptr) const
- vec8 **if_then_else** (const vec8 &then, const vec8 &else_part) const

Private Attributes

- __m256 **_v**

Friends

- std::ostream & **operator<<** (std::ostream &o, const vec8 &y)
- vec8 **operator &** (const vec8 &lhs, const not_vec8 &rhs)
- vec8 **operator &** (const not_vec8 &lhs, const vec8 &rhs)

5.83.1 Detailed Description

The vec8 class is the main SIMD float8 class using the GLSL nomenclature.

Author

Thanos Theo, 2009-2018

Version

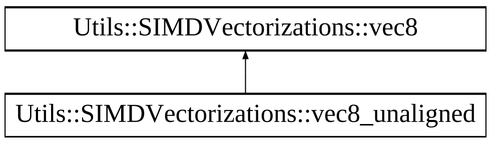
14.0.0.0

5.84 Utils::SIMDVectorizations::vec8_unaligned Class Reference

The vec8_unaligned class is the main unaligned SIMD float8 class using the GLSL nomenclature.

```
#include <SIMDVectorizations.h>
```

Inheritance diagram for Utils::SIMDVectorizations::vec8_unaligned:


Utils::SIMDVectorizations::vec8
Utils::SIMDVectorizations::vec8_unaligned

Public Member Functions

- **vec8_unaligned** (const float ∗__restrict src)

5.84.1 Detailed Description

The vec8_unaligned class is the main unaligned SIMD float8 class using the GLSL nomenclature.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Index