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 OpenGLAssimpModelLoader

  This class encapsulates usage of an OpenGL Model Loader using the Assimp library.
- 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 OpenGLDriverInfo 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 OpenGLILTexture

  This class encapsulates usage of an OpenGL texture using the DevIL library for image handling.
- class OpenGLLight

  This class encapsulates usage of an OpenGL light (directional or positional).
- struct OpenGLLightInterface

  This abstract class encapsulates usage of an OpenGL light.
- class OpenGLLookAtCamera

  This class encapsulates usage of an OpenGL LookAt camera.
- class OpenGLMaterial

This class encapsulates usage of OpenGL materials.

- class OpenGLModelAmbientLight

  This class encapsulates usage of an OpenGL model ambient light.
- class OpenGLPerlinNoise

  This class provides Perlin Noise functionality for GLSL Shaders usage through a 3D OpenGL texture.
- 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 OpenGLShaderImpostorModels

  This class loads & encapsulates usage of the GLSL shader point sphere models.
- 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 OpenGLShaderSurfaceLightingLODModels

  This class loads & encapsulates usage of the 5 GLSL shader lighting models with LOD (real-time adaptive tessellation with GL 4.0+): Phong, Blinn-Phong, Gaussian, Toon & Gooch.
- class OpenGLShaderSurfaceLightingModels

  This class loads & encapsulates usage of the 5 GLSL shader lighting models: Phong, Blinn-Phong, Gaussian, Toon & Gooch.
- struct OpenGLSimplexNoise

  This class includes the improved Simplex Perlin Noise static methods (original author Prof.
- class OpenGLUniqueColorsGenerator

  This class encapsulates usage of creating unique colors based on a prime number generator.
- 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 CommonFunctions_CommonFunctions header file.

Author

Thanos Theo, 2009-2018

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

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled FXAA_Antialias_ApplyFXAA_Antialias header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled GBufferEffects_ApplyFullScreenQuad header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled GBufferEffects_ApplyGBufferEffects header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled GBufferEffects_ApplyGBufferEffectsBlurXY header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled GBufferEffects_ApplyGBufferEffectsHSSAO header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled IlluminatedLineModels_ApplyIlluminatedHaloModel header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled IlluminatedLineModels_ApplyIlluminatedLineModel header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled OrderIndependentTransparency_ApplyClearABuffer3D header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled OrderIndependentTransparency_ApplyClearABuffer3DLinkedList header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled OrderIndependentTransparency_ApplyDisplayABuffer3D header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled OrderIndependentTransparency_ApplyDisplayABuffer3DLinkedList header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled OrderIndependentTransparency_ApplyFullScreenQuad header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled OrderIndependentTransparency_RenderABuffer3D header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled OrderIndependentTransparency_RenderABuffer3DLinkedList header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled PointCylinderModels_ApplyCylinderModel header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled PointSphereModels_ApplyPointSphereModel header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled PointSphereModels_ApplyQuadSphereModel header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled PointSphereTessLODModels_ApplyPointSphereTessLODModel header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled PointSphereTessQuadModels_ApplyPointSphereTessQuadModel header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled SurfaceLightingLODModels_ApplySurfaceLightingLODModels header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled SurfaceLightingModels_ApplySurfaceLightingModels header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled SurfaceLightingModels_ApplySurfaceSimplePhongLightingModel header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled SurfaceLightingModels_SurfaceLightingModels header file.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

Automatically generated by the ShaderFilesGenerator.

Defines the scrambled SurfaceLightingModels_SurfaceSimplePhongLightingModel 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 ModelLoaderDepthOfFieldTest

  ModelLoaderDepthOfFieldTest is the 6th set of OpenGL rendering tests.
- class ModelLoaderOITTest

  ModelLoaderOITTest is the 7th set of OpenGL rendering tests.
- class ModelLoaderTest

  ModelLoaderTest is the 3rd set of OpenGL rendering tests.
- class ParticleRenderingModelLoaderTest

  ParticleRenderingModelLoaderTest is the 5th set of OpenGL rendering tests.
- class ParticleRenderingSuperQuadricsTest

  ParticleRenderingSuperQuadricsTest is the 4th set of OpenGL rendering tests.
- class SuperQuadricsTest

  SuperQuadricsTest is the 2nd 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 SuperQuadrics Namespace Reference

Namespace SuperQuadrics for the Super Quadric algorithmic shapes.

Classes

- struct ModelSettings

  ModelSettings class which holds information for the settings of a SuperQuadrics model.
- struct SuperQuadricSettings

  SuperQuadricSettings is the class that acts as a placeholder of the SuperQuadricShape variables needed for the parametric polar coordinate equations.
- class SuperQuadricShape

  A SuperQuadricShape is the class that creates SuperQuadric-based shapes using parametric polar coordinate equations.
- struct SuperQuadricShapesProducer

  Many examples using the SuperQuadricShapesProducer for producing various SuperQuadric-based shapes.

4.4.1 Detailed Description

Namespace SuperQuadrics for the Super Quadric algorithmic shapes.

Author

Thanos Theo, 2018

Version

14.0.0.0

4.5 Tests Namespace Reference

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

Namespaces

- DeviceGoogleTest01__UTILS_CUDA_Classes

  Device Google Test 01 for the UtilsCUDA::CUDADriverInfo class functionality.
- DeviceGoogleTest02__UTILS_CUDA_Classes

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

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

  Device Google Test 04 for the UtilsCUDA::CUDAMemoryHandler set of classes functionality.
- DeviceGoogleTest05__UTILS_CUDA_Classes

  Device Google Test 05 for the UtilsCUDA::CUDAMemoryHandler (RawDeviceMemory, Span) set of classes functionality.
- DeviceGoogleTest06__UTILS_CUDA_Classes

  Device Google Test 06 for the UtilsCUDA::CUDAUtilityFunctions functionality.
- DeviceGoogleTest07__UTILS_CUDA_Classes

  Device Google Test 07 for the UtilsCUDA::CUDADeviceUtilityFunctions functionality.
- DeviceGoogleTest08__UTILS_CUDA_Classes

  Device Google Test 08 for the UtilsCUDA::CUDAKernelLauncher functionality.
- DeviceGoogleTest09__UTILS_CUDA_Classes

  Device Google Test 09 for the UtilsCUDA::CUDAMemoryPool functionality.
- DeviceGoogleTest10__UTILS_CUDA_Classes

  Device Google Test 10 for the UtilsCUDA::CUDAUtilityFunctions fast memset functionality.
- DeviceGoogleTest11__UTILS_CUDA_Classes

  Device Google Test 11 for the UtilsCUDA::CUDAUtilityFunctions powf(float) & pow(double) functionality.
- DeviceGoogleTest12__UTILS_CUDA_Classes

  Device Google Test 12 for the UtilsCUDA::CUDAQueue functionality.
- HostDeviceStressGoogleTest01

  Device Stress Google Test 01 for the UtilsCUDA::CUDALinearAlgebraGPUComputingStressTest class functionality.
- HostGoogleTest01__UTILS_Classes

  Host Google Test 01 for the Utils::AccurateTimers::AccurateCPUTimer class functionality.
- HostGoogleTest02__UTILS_Classes

  Host Google Test 02 for the Utils::Randomizers::RandomRNGWELL512 class functionality.
- HostGoogleTest03__UTILS_Classes

  Host Google Test 03 for the Utils::SIMDVectorizations classes functionality.
- HostGoogleTest04__UTILS_Classes

  Host Google Test 04 for the Utils::UtilityFunctions::BitManipulationFunctions class functionality.
- HostGoogleTest05__UTILS_CPUParallelism_Classes

  Host Google Test 05 for the Utils::CPUParallelism parallelFor() functionality.
- HostGoogleTest06__UTILS_CPUParallelism_Classes

  Host Google Test 06 for the Utils::CPUParallelism::CPUParallelismTest class for the parallelFor() functionality.
- HostGoogleTest07__Lodepng_Classes

  Host Google Test 07 for the lodepng class for png encoding/decoding functionality.
- HostGoogleTest08__ASTAR_Classes

  Host Google Test 08 for the Utils::Pathfinding::Astar class for the A∗ algorithm functionality.
- HostGoogleTest09__UTILS_Classes

  Host Google Test 09 for the Utils::UtilityFunctions::MathFunctions class functionality.
- HostGoogleTest10__UTILS_CPUParallelism_Classes

  Host Google Test 10 for the for the parallelFor() thread local & ThreadPool functionality.
- HostStressGoogleTest01

  Host Stress Google Test 01 for the Utils::LinearAlgebraCPUComputingStressTest class functionality.

4.5.1 Detailed Description

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

Author

Thanos Theo, 2018

Version

14.0.0.0

Author

Thanos Theo, 2019

Version

14.0.0.0

4.6 Tests::DeviceGoogleTest01__UTILS_CUDA_Classes Namespace Reference

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

Functions

- void **executeTest** ()

4.6.1 Detailed Description

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

Author

Thanos Theo, 2018

Version

14.0.0.0

4.7 Tests::DeviceGoogleTest02__UTILS_CUDA_Classes Namespace Reference

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

Functions

- void **executeTest** ()

4.7.1 Detailed Description

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

Author

Thanos Theo, 2018

Version

14.0.0.0

4.8 Tests::DeviceGoogleTest03__UTILS_CUDA_Classes Namespace Reference

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

Functions

- void **executeTest** ()

4.8.1 Detailed Description

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

Author

Thanos Theo, 2018

Version

14.0.0.0

4.9 Tests::DeviceGoogleTest04__UTILS_CUDA_Classes Namespace Reference

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

Functions

- void **executeTest** ()

4.9.1 Detailed Description

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

Author

Thanos Theo, 2018

Version

14.0.0.0

4.10 Tests::DeviceGoogleTest05__UTILS_CUDA_Classes Namespace Reference

Device Google Test 05 for the UtilsCUDA::CUDAMemoryHandler (RawDeviceMemory, Span) set of classes functionality.

Functions

- void **executeTest** ()

4.10.1 Detailed Description

Device Google Test 05 for the UtilsCUDA::CUDAMemoryHandler (RawDeviceMemory, Span) set of classes functionality.

Author

David Lenz, 2019

Version

14.0.0.0

4.11 Tests::DeviceGoogleTest06__UTILS_CUDA_Classes Namespace Reference

Device Google Test 06 for the UtilsCUDA::CUDAUtilityFunctions functionality.

Functions

- void **executeTest** ()

4.11.1 Detailed Description

Device Google Test 06 for the UtilsCUDA::CUDAUtilityFunctions functionality.

Author

Thanos Theo, 2019

Version

14.0.0.0

4.12 Tests::DeviceGoogleTest07__UTILS_CUDA_Classes Namespace Reference

Device Google Test 07 for the UtilsCUDA::CUDADeviceUtilityFunctions functionality.

Functions

- void **executeTest** ()

4.12.1 Detailed Description

Device Google Test 07 for the UtilsCUDA::CUDADeviceUtilityFunctions functionality.

Author

Thanos Theo, 2019

Version

14.0.0.0

4.13 Tests::DeviceGoogleTest08__UTILS_CUDA_Classes Namespace Reference

Device Google Test 08 for the UtilsCUDA::CUDAKernelLauncher functionality.

Functions

- void **executeTest** ()

4.13.1 Detailed Description

Device Google Test 08 for the UtilsCUDA::CUDAKernelLauncher functionality.

Author

David Lenz, 2019

Version

14.0.0.0

4.14 Tests::DeviceGoogleTest09__UTILS_CUDA_Classes Namespace Reference

Device Google Test 09 for the UtilsCUDA::CUDAMemoryPool functionality.

Functions

- void **executeTest** ()

4.14.1 Detailed Description

Device Google Test 09 for the UtilsCUDA::CUDAMemoryPool functionality.

Author

Thanos Theo, 2019

Version

14.0.0.0

4.15 Tests::DeviceGoogleTest10__UTILS_CUDA_Classes Namespace Reference

Device Google Test 10 for the UtilsCUDA::CUDAUtilityFunctions fast memset functionality.

Functions

- void **executeTest** ()

4.15.1 Detailed Description

Device Google Test 10 for the UtilsCUDA::CUDAUtilityFunctions fast memset functionality.

Author

Leonid Volnin, 2019

Version

14.0.0.0

4.16 Tests::DeviceGoogleTest11__UTILS_CUDA_Classes Namespace Reference

Device Google Test 11 for the UtilsCUDA::CUDAUtilityFunctions powf(float) & pow(double) functionality.

Functions

- void **executeTest** ()

4.16.1 Detailed Description

Device Google Test 11 for the UtilsCUDA::CUDAUtilityFunctions powf(float) & pow(double) functionality.

Author

Thanos Theo, 2019

Version

14.0.0.0

4.17 Tests::DeviceGoogleTest12__UTILS_CUDA_Classes Namespace Reference

Device Google Test 12 for the UtilsCUDA::CUDAQueue functionality.

Functions

- void **executeTest** ()

4.17.1 Detailed Description

Device Google Test 12 for the UtilsCUDA::CUDAQueue functionality.

Author

Leonid Volnin, 2019

Version

14.0.0.0

4.18 Tests::HostDeviceStressGoogleTest01 Namespace Reference

Device Stress Google Test 01 for the UtilsCUDA::CUDALinearAlgebraGPUComputingStressTest class functionality.

Functions

- void **executeTest** ()

4.18.1 Detailed Description

Device Stress Google Test 01 for the UtilsCUDA::CUDALinearAlgebraGPUComputingStressTest class functionality.

Author

Thanos Theo, 2019

Version

14.0.0.0

4.19 Tests::HostGoogleTest01__UTILS_Classes Namespace Reference

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

Functions

- void **executeTest** ()

4.19.1 Detailed Description

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

Author

Thanos Theo, 2018

Version

14.0.0.0

4.20 Tests::HostGoogleTest02__UTILS_Classes Namespace Reference

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

Functions

- void **executeTest** ()

4.20.1 Detailed Description

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

Author

Thanos Theo, 2018

Version

14.0.0.0

4.21 Tests::HostGoogleTest03__UTILS_Classes Namespace Reference

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

Functions

- void **executeTest** ()

4.21.1 Detailed Description

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

Author

Thanos Theo, 2018

Version

14.0.0.0

4.22 Tests::HostGoogleTest04__UTILS_Classes Namespace Reference

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

Functions

- void **executeTest** ()

4.22.1 Detailed Description

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

Author

Thanos Theo, 2018

Version

14.0.0.0

4.23 Tests::HostGoogleTest05__UTILS_CPUParallelism_Classes Namespace Reference

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

Functions

- void **executeTest** ()

4.23.1 Detailed Description

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

Author

Thanos Theo, 2018

Version

14.0.0.0

4.24 Tests::HostGoogleTest06__UTILS_CPUParallelism_Classes Namespace Reference

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

Functions

- void **executeTest** ()

4.24.1 Detailed Description

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

Author

Thanos Theo, 2018

Version

14.0.0.0

4.25 Tests::HostGoogleTest07__Lodepng_Classes Namespace Reference

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

Functions

- void **executeTest** ()

4.25.1 Detailed Description

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

Author

Thanos Theo, 2018

Version

14.0.0.0

4.26 Tests::HostGoogleTest08__ASTAR_Classes Namespace Reference

Host Google Test 08 for the Utils::Pathfinding::Astar class for the A∗ algorithm functionality.

Functions

- void **executeTest** ()

4.26.1 Detailed Description

Host Google Test 08 for the Utils::Pathfinding::Astar class for the A∗ algorithm functionality.

Author

Thanos Theo, 2018

Version

14.0.0.0

4.27 Tests::HostGoogleTest09__UTILS_Classes Namespace Reference

Host Google Test 09 for the Utils::UtilityFunctions::MathFunctions class functionality.

Functions

- void **executeTest** ()

4.27.1 Detailed Description

Host Google Test 09 for the Utils::UtilityFunctions::MathFunctions class functionality.

Author

Thanos Theo, 2018

Version

14.0.0.0

4.28 Tests::HostGoogleTest10__UTILS_CPUParallelism_Classes Namespace Reference

Host Google Test 10 for the for the parallelFor() thread local & ThreadPool functionality.

Functions

- void **executeTest** ()

4.28.1 Detailed Description

Host Google Test 10 for the for the parallelFor() thread local & ThreadPool functionality.

Author

Thanos Theo, 2018

Version

14.0.0.0

4.29 Tests::HostStressGoogleTest01 Namespace Reference

Host Stress Google Test 01 for the Utils::LinearAlgebraCPUComputingStressTest class functionality.

Functions

- void **executeTest** ()

4.29.1 Detailed Description

Host Stress Google Test 01 for the Utils::LinearAlgebraCPUComputingStressTest class functionality.

Author

Thanos Theo, 2019

Version

14.0.0.0

4.30 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.
- Pathfinding

  Namespace Pathfinding contains path finding algorithms like A∗.
- 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 LinearAlgebraCPUComputingStressTest

  This class contains a basic Linear Algebra CPU Computing stress test case in the host.
- 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)

Variables

- template<typename T >
  constexpr T PI = T(3.14159265358979323846)

  MathConstants.h:

  This header-only file contains preprocessor math constants defines to be used throughout the GPU framework.
- constexpr float **PI_FLT** = PI<float>
- constexpr double **PI_DBL** = PI<double>

4.30.1 Detailed Description

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

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.30.2 Variable Documentation

4.30.2.1 PI

```
template<typename T >
constexpr T Utils::PI = T(3.14159265358979323846)
```

MathConstants.h:

This header-only file contains preprocessor math constants defines to be used throughout the GPU framework.

Author

Thanos Theo, 2018

Version

14.0.0.0

4.31 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.
- class AccurateTimerInterface

  The AccurateTimerInterface struct encapsulates a basic interface for a generic high resolution timer using the Curiously Recurring Template Pattern (CRTP).
- struct AccurateTimerLog

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

  ProfileCPUTimer profiling helper class using RAII.

4.31.1 Detailed Description

Namespace AccurateTimers contains utility classes for accurate timer logging.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.32 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 CPUParallelismTest

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

  This class encapsulates all the CPUParalellism related utility functions.
- 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.

Enumerations

- enum ThreadAffinityMask : std::uint64_t {
  AFFINITY_MASK_NONE = 0, **AFFINITY_MASK_1_CPU_CORE** = 0x1, **AFFINITY_MASK_2_CPU_CORES** = 0x3, **AFFINITY_MASK_4_CPU_CORES** = 0xf,
  AFFINITY_MASK_8_CPU_CORES = 0xff, **AFFINITY_MASK_16_CPU_CORES** = 0xffff, **AFFINITY_MASK_32_CPU_CORES** = 0xffffffff, **AFFINITY_MASK_ALL** = 0xffffffffffffffff }

  Thread affinity mask constants.
- enum ThreadPriorities : std::size_t { **PRIORITY_NONE** = 0, **PRIORITY_MIN** = 1, **PRIORITY_NORMAL** = 50, **PRIORITY_MAX** = 99 }

  Thread priority constants.

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(), std::uint64_t affinityMask=AFFINITY_MASK_ALL, std::size_t priority=PRIORITY_NONE)

  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(), std::uint64_t affinityMask=AFFINITY_MASK_ALL, std::size_t priority=PRIORITY_NONE)
- UTILS_MODULE_API void **parallelFor** (std::size_t indexEnd, const FunctionView< void(std::size_t)> &kernelFunction, ThreadPool &threadPool)
- UTILS_MODULE_API void **parallelFor** (std::size_t indexStart, std::size_t indexEnd, const FunctionView< void(std::size_t)> &kernelFunction, ThreadPool &threadPool)
- UTILS_MODULE_API void parallelForThreadLocal (std::size_t indexEnd, const FunctionView< void(std::size_t, std::size_t)> &kernelFunction, std::size_t numberOfThreads=numberOfHardwareThreads(), std::uint64_t affinityMask=AFFINITY_MASK_ALL, std::size_t priority=PRIORITY_NONE)

  parallelFor() versions with both the index & threadIdx 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=numberOfHardwareThreads(), std::uint64_t affinityMask=AFFINITY_MASK_ALL, std::size_t priority=PRIORITY_NONE)
- UTILS_MODULE_API void **parallelForThreadLocal** (std::size_t indexEnd, const FunctionView< void(std::size_t, std::size_t)> &kernelFunction, ThreadPool &threadPool)
- UTILS_MODULE_API void **parallelForThreadLocal** (std::size_t indexStart, std::size_t indexEnd, const FunctionView< void(std::size_t, std::size_t)> &kernelFunction, ThreadPool &threadPool)
- UTILS_MODULE_API void **setAffinity** (std::thread &threadHandle, std::size_t threadIdx, std::uint64_t affinityMask)
- UTILS_MODULE_API bool **getAffinity** (std::thread &threadHandle, std::size_t threadIdx)
- UTILS_MODULE_API void **setPriority** (std::thread &threadHandle, std::size_t threadIdx, std::size_t priority)
- UTILS_MODULE_API std::size_t **getPriority** (std::thread &threadHandle, std::size_t threadIdx)

4.32.1 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

4.32.2 Enumeration Type Documentation

4.32.2.1 ThreadAffinityMask

```
enum Utils::CPUParallelism::ThreadAffinityMask : std::uint64_t
```

Thread affinity mask constants.

Note: On non-Windows platforms, the pthread_setaffinity_np() call is being used for CPU affinity.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.32.2.2 ThreadPriorities

```
enum Utils::CPUParallelism::ThreadPriorities : std::size_t
```

Thread priority constants.

Note: On non-Windows platforms, the pthread_setschedprio() call is being used for thread priority.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.33 Utils::Pathfinding Namespace Reference

Namespace Pathfinding contains path finding algorithms like A∗.

Classes

- class Astar

  This class encapsulates usage of the A∗ algorithm.

4.33.1 Detailed Description

Namespace Pathfinding contains path finding algorithms like A∗.

Author

Thanos Theo, 2018

Version

14.0.0.0

4.34 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.

4.34.1 Detailed Description

Namespace Randomizers contains random number generator classes.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.35 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 isSupportedSSE41 ()

  Function to test for SSE41 support (x86 architecture).
- bool isSupportedSSE42 ()

  Function to test for SSE42 support (x86 architecture).
- bool isSupportedAVX ()

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

  Function to test for AVX2 support (x86 architecture).
- bool isSupportedAVX512F ()

  Function to test for AVX512F 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)

4.35.1 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

4.35.2 Function Documentation

4.35.2.1 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

4.35.2.2 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

4.35.2.3 isSupportedAVX()

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

Function to test for AVX support (x86 architecture).

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.35.2.4 isSupportedAVX2()

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

Function to test for AVX2 support (x86 architecture).

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.35.2.5 isSupportedAVX512F()

```
bool Utils::SIMDVectorizations::isSupportedAVX512F ( )  [inline]
```

Function to test for AVX512F support (x86 architecture).

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.35.2.6 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

4.35.2.7 isSupportedSSE3()

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

Function to test for SSE3 support (x86 architecture).

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.35.2.8 isSupportedSSE41()

```
bool Utils::SIMDVectorizations::isSupportedSSE41 ( )  [inline]
```

Function to test for SSE41 support (x86 architecture).

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.35.2.9 isSupportedSSE42()

```
bool Utils::SIMDVectorizations::isSupportedSSE42 ( )  [inline]
```

Function to test for SSE42 support (x86 architecture).

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.36 Utils::UnitTests Namespace Reference

Namespace UnitTests contains classes used for unit testing.

Classes

- class IUnitTests

  The IUnitTests struct encapsulate a basic unit test interface using the Curiously Recurring Template Pattern (CRTP).
- 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 >

4.36.1 Detailed Description

Namespace UnitTests contains classes used for unit testing.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.37 Utils::UtilityFunctions Namespace Reference

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

Classes

- struct ArrayIndicingFunctions

  The ArrayIndicingFunctions struct provides array indexing functionality.
- struct Base64CompressorScrambler

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

  The BitManipulationFunctions struct provides bit manipulation functionality.
- class DebugConsole

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

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

  The StdAuxiliaryFunctions struct 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.
- struct TrigonometricFunctions

  The TrigonometricFunctions struct covers essential operations involving angles on the trigonometric circle.

4.37.1 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

4.38 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.

4.38.1 Detailed Description

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

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

4.39 UtilsCUDA Namespace Reference

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

Namespaces

- CUDAParallelFor

  namespace CUDAParallelFor for encapsulating a CUDA related parallelFor() construct using a combination of C99 variadic macros & C++11 variadic templates.

Classes

- struct CUDADeleter

  Custom deleter class for device memory.
- 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 CUDALinearAlgebraGPUComputingStressTest

  This class contains a basic Linear Algebra GPU Computing stress test case in host & device.
- class CUDALinearAlgebraGPUComputingTest

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

  This class encapsulates CUDA Memory Pool functionality for both host & device with reporting.
- class CUDAMemoryRegistry

  This class encapsulates CUDA Memory Registry functionality for pre-allocated host memory with reporting.
- class CUDAProcessMemoryPool

  This class encapsulates CUDA Process Memory Pool functionality for both host & device with reporting.
- class CUDAQueue

  CUDAQueue class for GPU memory.
- class CUDAQueueView
- 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.
- class CUDAUtilityDeviceFunctions

  This class encapsulates all the CUDA related device only utility functions.
- struct CUDAUtilityFunctions

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

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

  Helper class to differentiate whether something is fixed size at compile time or of dynamic size.
- class DynamicOrCompileTimeSize< DYNAMIC_SIZE >

  Template specialization for dynamically sized objects.
- class HostDeviceMemory

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

  This class encapsulates usage of a collection of host handling techniques (host side only) & the RAII C++ idiom.
- class KernelLauncher

  CUDA helper class to perform a more readable kernel launch in code with the fluent builder pattern.
- class MemoryHandlersAbstraction

  The MemoryHandlersAbstraction class encapsulates a basic abstraction layer for the memory handlers using the Curiously Recurring Template Pattern (CRTP).
- 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

Custom deleter class for (pinned) host memory.

- class ProfileGPUTimer

  ProfileGPUTimer profiling helper class using RAII.

- class RawDeviceMemory

  Helper class for the Span class below.

- class Span

  Helper class implementing the "span" paradigm from the CppCoreGuidelines applied to device arrays.

- class SpanStorage

  Helper class to store a pointer and an associated size with it.

Typedefs

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

Functions

- template<typename T >
  PinnedUniquePtr< T > **make_unique_pinned** (std::size_t numberOfElements, unsigned int flags=cuda↩HostRegisterDefault) noexcept
- template<typename T >
  DeviceUniquePtr< T > **make_unique_device** (std::size_t numberOfElements, int device=0, bool use↩UnifiedMemory=false) noexcept
- template<typename T >
  bool isValidHostDevicePointer (const T *ptr)

  Check the validity of the given host/device pointer.

- template<typename T >
  bool isValidDevicePointerWithCurrentDevice (const T *ptr)

  Check the validity of the given device pointer which the memory was allocated or registered versus the current device for the calling host thread.

Variables

- constexpr std::size_t **DYNAMIC_SIZE** = std::numeric_limits<std::size_t>::max()

4.39.1 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

4.39.2 Function Documentation

4.39.2.1 isValidDevicePointerWithCurrentDevice()

```
template<typename T >
bool UtilsCUDA::isValidDevicePointerWithCurrentDevice (
            const T * ptr )  [inline]
```

Check the validity of the given device pointer which the memory was allocated or registered versus the current device for the calling host thread.

Author

Thanos Theo, 2019

Version

14.0.0.0

4.39.2.2 isValidHostDevicePointer()

```
template<typename T >
bool UtilsCUDA::isValidHostDevicePointer (
            const T * ptr )  [inline]
```

Check the validity of the given host/device pointer.

Author

Thanos Theo, 2019

Version

14.0.0.0

4.40 UtilsCUDA::CUDAParallelFor Namespace Reference

namespace CUDAParallelFor for encapsulating a CUDA related parallelFor() construct using a combination of C99 variadic macros & C++11 variadic templates.

Functions

- template<typename FunctionType , typename... Args>
  __global__ void **kernel** (std::size_t arraySize, const FunctionType &lambdaFunction, Args... args)
- template<typename FunctionType , typename... Args>
  static void **launchCUDAParallelForWithDimensions** (std::size_t arraySize, const dim3 &blocks, const dim3 &threads, const FunctionType &lambdaFunction, Args &&... args)
- template<typename FunctionType , typename... Args>
  static void **launchCUDAParallelFor** (std::size_t arraySize, const FunctionType &lambdaFunction, Args &&... args)
- template<typename FunctionType , typename... Args>
  static void **launchCUDAParallelForInStreamWithDimensions** (std::size_t arraySize, const dim3 &blocks, const dim3 &threads, std::size_t sharedMemoryBytes, const cudaStream_t &stream, const FunctionType &lambdaFunction, Args &&... args)
- template<typename FunctionType , typename... Args>
  static void **launchCUDAParallelForInStream** (std::size_t arraySize, std::size_t sharedMemoryBytes, const cudaStream_t &stream, const FunctionType &lambdaFunction, Args &&... args)

4.40.1 Detailed Description

namespace CUDAParallelFor for encapsulating a CUDA related parallelFor() construct using a combination of C99 variadic macros & C++11 variadic templates.

Author

Thanos Theo, Amir Shahvarani, 2019

Version

14.0.0.0

Chapter 5

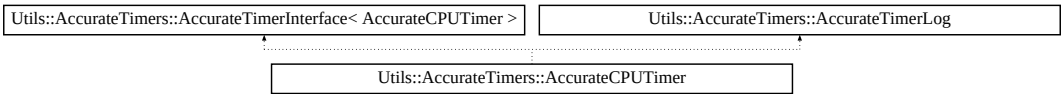
Class Documentation

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:



Public Member Functions

- void **startTimer** ()
- void **stopTimer** ()
- double **getElapsedTimeInNanoSecs** ()
- double **getElapsedTimeInMicroSecs** ()
- double **getElapsedTimeInMilliSecs** ()
- double **getElapsedTimeInSecs** ()
- double **getMeanTimeInNanoSecs** ()
- double **getMeanTimeInMicroSecs** ()
- double **getMeanTimeInMilliSecs** ()
- double **getMeanTimeInSecs** ()
- double **getDecimalElapsedTimeInMicroSecs** ()
- double **getDecimalElapsedTimeInMilliSecs** ()
- double **getDecimalElapsedTimeInSecs** ()
- double **getDecimalMeanTimeInMicroSecs** ()
- double **getDecimalMeanTimeInMilliSecs** ()
- double **getDecimalMeanTimeInSecs** ()
- **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. Using the Curiously Recurring Template Pattern (CRTP).

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.2 Utils::AccurateTimers::AccurateTimerInterface< Derived > Class Template Reference

The AccurateTimerInterface struct encapsulates a basic interface for a generic high resolution timer using the Curiously Recurring Template Pattern (CRTP).

`#include <AccurateTimers.h>`

Public Member Functions

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

Private Member Functions

- Derived * **asDerived** ()
- const Derived * **asDerived** () const

5.2.1 Detailed Description

template<typename Derived>
class Utils::AccurateTimers::AccurateTimerInterface< Derived >

The AccurateTimerInterface struct encapsulates a basic interface for a generic high resolution timer using the Curiously Recurring Template Pattern (CRTP).

Author

Thanos Theo, 2009-2018

Version

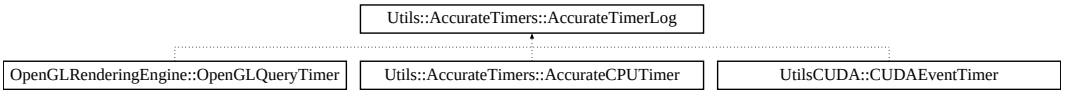
14.0.0.0

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:



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 struct 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< std::size_t, std::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< std::size_t, std::size_t, std::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 struct provides array indexing functionality.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.6 Utils::Pathfinding::Astar Class Reference

This class encapsulates usage of the A∗ algorithm.

`#include <Astar.h>`

Public Types

- enum DiagonalMode : std::size_t { **NO_DIADONALS** = 0, **WITH_DIADONALS** = 1 }

  Diagonal mode enumeration.

Public Member Functions

- void setStartAndGoal (std::size_t start, std::size_t goal)

  A function to set the A∗ start and goal positions.
- void setStrategy (const FunctionView< float(std::size_t)> &strategy) const

  A function for setting the state.
- bool execute () const

  An implementation of the A∗ path finding algorithm (not taking diagonals into account).
- void unravelAndMarkOptimalPath (std::uint8_t ∗__restrict optimalPath, std::uint8_t markValue) const

  A function for unraveling and marking the optimal path after after the execution of the A∗ algorithm.
- Astar (std::size_t width, std::size_t height, DiagonalMode diagonalMode=DiagonalMode::NO_DIADONALS) noexcept

  The non-default explicit constructor.
- **Astar** (const Astar &)=delete
- **Astar** (Astar &&)=delete
- Astar & **operator=** (const Astar &)=delete
- Astar & **operator=** (Astar &&)=delete

Private Member Functions

- float calculateCostH (std::size_t index) const

  A function for calculating the H cost.
- void resetState () const

  A function for resetting the state.

Private Attributes

- std::size_t **width_** = 0
- std::size_t **height_** = 0
- std::size_t **arraySize_** = 0
- std::size_t **start_** = 0
- std::size_t **goal_** = 0
- DiagonalMode **diagonalMode_** = DiagonalMode::NO_DIADONALS
- std::unique_ptr< float[ ]> **weights_** = nullptr
- std::unique_ptr< float[ ]> **costsH_** = nullptr
- std::unique_ptr< float[ ]> **totalCosts_** = nullptr
- std::unique_ptr< std::size_t[ ]> **paths_** = nullptr

5.6.1 Detailed Description

This class encapsulates usage of the A∗ algorithm.

This class encapsulates usage of the A∗ algorithm.
Path finding algorithms conceptual ideas were based on: http://qiao.github.io/PathFinding.↩
js/visual/.
Heuristics for grid maps conceptual ideas were based on: http://theory.stanford.edu/∼amitp/↩
GameProgramming/Heuristics.html.
A∗ prototype C/C++ implementation was based on: https://github.com/hjweide/a-star/.

Author

Thanos Theo, 2018

Version

14.0.0.0

5.6.2 Member Function Documentation

5.6.2.1 calculateCostH()

```
float Astar::calculateCostH (
            std::size_t index ) const  [private]
```

A function for calculating the H cost.

Internally it uses a Manhattan distance metric.

5.7 Utils::UtilityFunctions::Base64CompressorScrambler Struct Reference

The Base64CompressorScrambler struct 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)

5.7.1 Detailed Description

The Base64CompressorScrambler struct provides encoding/decoding functionality to strings.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.8 Utils::UtilityFunctions::BitManipulationFunctions Struct Reference

The BitManipulationFunctions struct 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

- template<typename T >
  static bool isPowerOfTwo (T value, std::enable_if_t< std::is_integral< T >::value > *=nullptr)

  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 getPrevPowerOfTwo (unsigned int value)

  Gets the previous power-of-two of a given 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).

5.8.1 Detailed Description

The BitManipulationFunctions struct provides bit manipulation functionality.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.8.2 Member Function Documentation

5.8.2.1 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

5.8.2.2 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

5.8.2.3 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

5.8.2.4 getPrevPowerOfTwo()

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

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

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

5.8.2.5 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.

5.8.2.6 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.

5.8.2.7 isPowerOfTwo()

```
template<typename T >
static bool Utils::UtilityFunctions::BitManipulationFunctions::isPowerOfTwo (
            T value,
            std::enable_if_t< std::is_integral< T >::value > *  = nullptr )  [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

5.9 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_**

5.9.1 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

5.10 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
- std::string **getTextureFileName** () const
- std::string **getModelFileName** () const
- bool **getUseModelLoaderDescriptor** () const
- std::string **getModelLoaderDescriptorFileName** () 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_** = 3
- std::string **textureFileName_** = "DotRedLogo.png"
- std::string **modelFileName_** = "dragon.obj"
- bool **useModelLoaderDescriptor_** = false
- std::string **modelLoaderDescriptorFileName_** = "ModelLoaderDescriptor.tsr"

5.10.1 Detailed Description

This class encapsulates config file handling.

Author

Thanos Theo, 2009-2018

Version

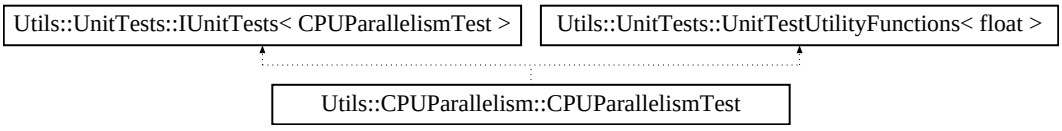
14.0.0.0

5.11 Utils::CPUParallelism::CPUParallelismTest Class Reference

This class encapsulates unit testing of CPUParallelism libraries.

`#include <CPUParallelismTest.h>`

Inheritance diagram for Utils::CPUParallelism::CPUParallelismTest:



Public Member Functions

- void **resetTests** ()
- bool **conductTests** ()
- void **reportTestResults** ()
- **CPUParallelismTest** (std::size_t dimensions=512, std::size_t numberOfThreads=numberOfHardware↩ Threads(), bool useRandomness=false) noexcept
- **CPUParallelismTest** (std::tuple< std::size_t, std::size_t, std::size_t > dimensionsXYZ, std::size_t number↩ OfThreads=numberOfHardwareThreads(), bool useRandomness=false) noexcept
- **CPUParallelismTest** (const CPUParallelismTest &)=delete
- **CPUParallelismTest** (CPUParallelismTest &&)=delete
- CPUParallelismTest & **operator=** (const CPUParallelismTest &)=delete
- CPUParallelismTest & **operator=** (CPUParallelismTest &&)=delete

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

5.11.1 Detailed Description

This class encapsulates unit testing of CPUParallelism libraries.

CPUParallelismTest.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

5.12 Utils::CPUParallelism::CPUParallelismUtilityFunctions Class Reference

This class encapsulates all the CPUParalellism related utility functions.

`#include <CPUParallelismUtilityFunctions.h>`

Public Member Functions

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

Static Public Member Functions

- template<typename F , typename... Ts>
  static 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 >
  static T atomicAdd (std::atomic< T > &value, T newValue, std::enable_if_t< std::is_floating_point< T >↩
  ::value > *=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).
- template<typename T >
  static T atomicMultiply (std::atomic< T > &value, T newValue, std::enable_if_t< std::is_floating_point< T >::value > *=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).
- template<typename T >
  static T atomicMin (std::atomic< T > &value, T newValue, std::enable_if_t< std::is_arithmetic< T >::value > *=nullptr)

  Perform an atomic min to the T (arithmetic type only allowed for T).
- template<typename T >
  static T atomicMax (std::atomic< T > &value, T newValue, std::enable_if_t< std::is_arithmetic< T >::value > *=nullptr)

  Perform an atomic max to the T (arithmetic type only allowed for T).

Static Private Member Functions

- template<typename T , typename ArithmeticOp >
  static T atomicArithmeticOp (std::atomic< T > &value, T newValue, ArithmeticOp arithmeticOp)

  Perform an atomic arithmetic operation to the T via spin-locking on compare_exchange_weak(), the Compare-and-Swap (CAS) algorithm.
- template<typename T , typename ComparisonOp >
  static T atomicComparisonOp (std::atomic< T > &value, T newValue, ComparisonOp comparisonOp)

  Perform an atomic comparison operation to the T via spin-locking on compare_exchange_weak(), the Compare-and-Swap (CAS) algorithm.

5.12.1 Detailed Description

This class encapsulates all the CPUParalellism related utility functions.

Author

Thanos Theo, 2018

Version

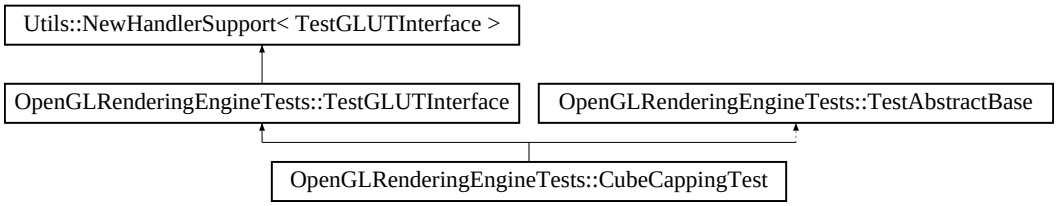
14.0.0.0

5.13 OpenGLRenderingEngineTests::CubeCappingTest Class Reference

CubeCappingTest is the 1st set of OpenGL rendering tests.

`#include <CubeCappingTest.h>`

Inheritance diagram for 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

5.13.1 Detailed Description

CubeCappingTest is the 1st set of OpenGL rendering tests.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.14 UtilsCUDA::CUDADeleter< T > Struct Template Reference

Custom deleter class for device memory.

`#include <CUDAMemoryHandlers.h>`

Public Member Functions

- **CUDADeleter** (bool useDeleter=true) noexcept
- void **operator()** (T *ptr) noexcept

Public Attributes

- bool **useDeleter_** = false

5.14.1 Detailed Description

template<typename T>
struct UtilsCUDA::CUDADeleter< T >

Custom deleter class for device memory.

Author

David Lenz, Thanos Theo, 2018

Version

14.0.0.0

5.15 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 (int device) const
  Device is a Fermi-based GPU.
- bool getIsKepler (int device) const
  Device is a Kepler-based GPU.
- bool getIsMaxwell (int device) const
  Device is a Maxwell-based GPU.
- bool getIsPascal (int device) const
  Device is a Pascal-based GPU.
- bool getIsVolta (int device) const
  Device is a Volta-based GPU.
- bool getIsTuring (int device) const
  Device is a Turing-based GPU.
- bool getIsAtLeastFermi (int device) const
  Device is at least a Fermi-based GPU.
- bool getIsAtLeastKepler (int device) const
  Device is at least a Kepler-based GPU.
- bool getIsAtLeastMaxwell (int device) const
  Device is at least a Maxwell-based GPU.
- bool getIsAtLeastPascal (int device) const
  Device is at least a Pascal-based GPU.
- bool getIsAtLeastVolta (int device) const
  Device is at least a Volta-based GPU.
- bool getIsAtLeastTuring (int device) const
  Device is at least a Turing-based GPU.
- bool getHasDynamicParallelism (int device) const
  Device support for Dynamic Parallelism.
- bool getHasUnifiedMemory (int device) const
  Device support for Unified Memory.
- std::string getName (int device) const
  ASCII string identifying device.
- std::size_t getTotalGlobalMemory (int device) const
  Global memory available on device in bytes.
- std::size_t getSharedMemoryPerBlock (int device) const
  Shared memory available per block in bytes.
- int getRegistersPerBlock (int device) const
  32-bit registers available per block
- int getWarpSize (int device) const
  Warp size in threads.
- std::size_t getMemoryPitch (int device) const
  Maximum pitch in bytes allowed by memory copies.
- int getMaxThreadsPerBlock (int device) const
  Maximum number of threads per block.
- const int * getMaxThreadsDimension (int device) const

Maximum size of each dimension of a block.

- const int * getMaxGridSize (int device) const

  Maximum size of each dimension of a grid.

- int getClockRate (int device) const

  Clock frequency in kilohertz.

- std::size_t getTotalConstMemory (int device) const

  Constant memory available on device in bytes.

- int getMajorVersion (int device) const

  Major compute capability.

- int getMinorVersion (int device) const

  Minor compute capability.

- std::size_t getTextureAlignment (int device) const

  Alignment requirement for textures.

- std::size_t getTexturePitchAlignment (int device) const

  Pitch alignment requirement for texture references bound to pitched memory.

- int getDeviceOverlap (int device) const

  Device can concurrently copy memory and execute a kernel. Deprecated. Use instead asyncEngineCount.

- int getMultiProcessorCount (int device) const

  Number of multiprocessors on device.

- int getKernelExecTimeoutEnabled (int device) const

  Specified whether there is a run time limit on kernels.

- int getIntegrated (int device) const

  Device is integrated as opposed to discrete.

- int getCanMapHostMemory (int device) const

  Device can map host memory with cudaHostAlloc/cudaHostGetDevicePointer.

- int getComputeMode (int device) const

  Compute mode (See ::cudaComputeMode)

- int getMaxTexture1D (int device) const

  Maximum 1D texture size.

- int getMaxTexture1DMipmap (int device) const

  Maximum 1D mipmapped texture size.

- int getMaxTexture1DLinear (int device) const

  Maximum size for 1D textures bound to linear memory.

- const int * getMaxTexture2D (int device) const

  Maximum 2D texture dimensions.

- const int * getMaxTexture2DMipmap (int device) const

  Maximum 2D mipmapped texture dimensions.

- const int * getMaxTexture2DLinear (int device) const

  Maximum dimensions (width, height, pitch) for 2D textures bound to pitched memory.

- const int * getMaxTexture2DGather (int device) const

  Maximum 2D texture dimensions if texture gather operations have to be performed.

- const int * getMaxTexture3D (int device) const

  Maximum 3D texture dimensions.

- const int * getMaxTexture3DAlt (int device) const

  Maximum alternate 3D texture dimensions.

- int getMaxTextureCubemap (int device) const

  Maximum Cubemap texture dimensions.

- const int * getMaxTexture1DLayered (int device) const

  Maximum 1D layered texture dimensions.

- const int * getMaxTexture2DLayered (int device) const

  Maximum 2D layered texture dimensions.

- const int * getMaxTextureCubemapLayered (int device) const

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

  Maximum 1D surface size.
- const int * getMaxSurface2D (int device) const

  Maximum 2D surface dimensions.
- const int * getMaxSurface3D (int device) const

  Maximum 3D surface dimensions.
- const int * getMaxSurface1DLayered (int device) const

  Maximum 1D layered surface dimensions.
- const int * getMaxSurface2DLayered (int device) const

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

  Maximum Cubemap surface dimensions.
- const int * getMaxSurfaceCubemapLayered (int device) const

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

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

  Device can possibly execute multiple kernels concurrently.
- int getConcurrentManagedAccess (int device) const

  Device can coherently access managed memory concurrently with the CPU.
- int getHostNativeAtomicSupported (int device) const

  Link between the device and the host supports native atomic operations.
- int getECCEnabled (int device) const

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

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

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

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

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

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

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

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

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

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

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

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

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

Device supports caching locals in L1.

- std::size_t getSharedMemoryPerMultiprocessor (int device) const

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

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

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

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

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

  Device count.
- void reportCUDADriverInfo () const

  Reports the full CUDA Driver Info.
- **CUDADriverInfo** (unsigned int deviceFlags=cudaDeviceScheduleAuto, bool enableProfiling=false) noexcept
- **CUDADriverInfo** (const CUDADriverInfo &)=delete
- **CUDADriverInfo** (CUDADriverInfo &&)=delete
- CUDADriverInfo & **operator=** (const CUDADriverInfo &)=delete
- CUDADriverInfo & **operator=** (CUDADriverInfo &&)=delete

Private Member Functions

- std::string **getGPUArchitecture** (int device) const
- void **reportCUDAPlatformVersions** () const
- void **reportCUDADeviceCapabilities** (int device) const

Private Attributes

- bool **enableProfiling_** = false
- int **cudaDriverVersion_** = 0
- int **cudaRuntimeVersion_** = 0
- int **cudaDeviceCount_** = 0
- std::unique_ptr< cudaDeviceProp[ ]> **allCudaDevicesProperties_** = nullptr

5.15.1 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

Version

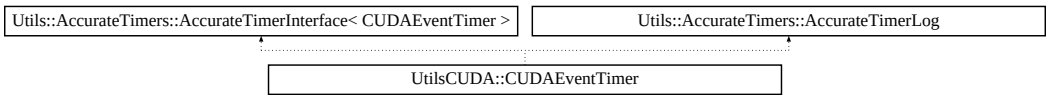
14.0.0.0

5.16 UtilsCUDA::CUDAEventTimer Class Reference

This class contains an AccurateTimers encapsulation of CUDA event timers.

`#include <CUDAEventTimer.h>`

Inheritance diagram for UtilsCUDA::CUDAEventTimer:



Public Member Functions

- void **startTimer** ()
- void **stopTimer** ()
- double **getElapsedTimeInNanoSecs** ()
- double **getElapsedTimeInMicroSecs** ()
- double **getElapsedTimeInMilliSecs** ()
- double **getElapsedTimeInSecs** ()
- double **getMeanTimeInNanoSecs** ()
- double **getMeanTimeInMicroSecs** ()
- double **getMeanTimeInMilliSecs** ()
- double **getMeanTimeInSecs** ()
- double **getDecimalElapsedTimeInMicroSecs** ()
- double **getDecimalElapsedTimeInMilliSecs** ()
- double **getDecimalElapsedTimeInSecs** ()
- double **getDecimalMeanTimeInMicroSecs** ()
- double **getDecimalMeanTimeInMilliSecs** ()
- double **getDecimalMeanTimeInSecs** ()
- **CUDAEventTimer** (int device=0, const cudaStream_t &cudaStream=nullptr) noexcept
- **CUDAEventTimer** (const CUDAEventTimer &)=delete
- **CUDAEventTimer** (CUDAEventTimer &&)=delete
- CUDAEventTimer & **operator=** (const CUDAEventTimer &)=delete
- CUDAEventTimer & **operator=** (CUDAEventTimer &&)=delete

Private Member Functions

- float **getElapsedTime** ()

Private Attributes

- const cudaStream_t **cudaStream_** {}
- cudaEvent_t **start_** {}
- cudaEvent_t **stop_** {}

Additional Inherited Members

5.16.1 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, no up-casting should ever be used. Using the Curiously Recurring Template Pattern (CRTP).

Author

Thanos Theo, 2018

Version

14.0.0.0

5.17 UtilsCUDA::CUDAGPUComputingAbstraction< Derived > Class Template Reference

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

`#include <CUDAGPUComputingAbstraction.h>`

Public Member Functions

- void initializeGPUMemory ()
  Initializes GPU memory.
- void performGPUComputing ()
  Performs the GPU Computing calculations.
- void retrieveGPUResults ()
  Retrieves the results from the GPU.
- bool verifyComputingResults ()
  Verifies the computing results between the CPU and the GPU.
- void releaseGPUComputingResources ()
  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

Private Member Functions

- Derived * **asDerived** ()
- const Derived * **asDerived** () const

5.17.1 Detailed Description

template<typename Derived>
class UtilsCUDA::CUDAGPUComputingAbstraction< Derived >

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

Using the Curiously Recurring Template Pattern (CRTP).

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. Using the Curiously Recurring Template Pattern (CRTP).

Author

Thanos Theo, 2018

Version

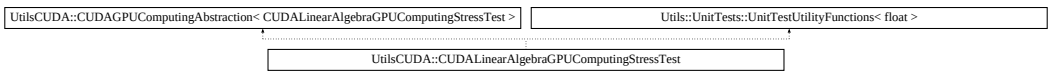
14.0.0.0

5.18 UtilsCUDA::CUDALinearAlgebraGPUComputingStressTest Class Reference

This class contains a basic Linear Algebra GPU Computing stress test case in host & device.

`#include <CUDALinearAlgebraGPUComputingStressTest.h>`

Inheritance diagram for UtilsCUDA::CUDALinearAlgebraGPUComputingStressTest:



Public Types

- enum **RunTypes** : std::size_t { **RUN_CPU** = 0, **RUN_GPU** = 1, **RUN_BOTH** = 2 }

Public Member Functions

- void initializeGPUMemory ()

  Initializes GPU memory.
- void performGPUComputing ()

  Performs the GPU Computing calculations.
- void retrieveGPUResults ()

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

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

  Releases the GPU Computing resources.
- **CUDALinearAlgebraGPUComputingStressTest** (const CUDADriverInfo &cudaDriverInfo, CUDAProcessMemoryPool &cudaProcessMemoryPool, int device=0, std::size_t arraySize=8192, const RunTypes &runType=RunTypes::RUN_BOTH, std::size_t numberOfCPUTheads=1, std::size_t numberOfCPUKernelIterations=160, bool useUnifiedMemory=false, bool useCounter=false) noexcept
- **CUDALinearAlgebraGPUComputingStressTest** (const CUDALinearAlgebraGPUComputingStressTest &)=delete
- **CUDALinearAlgebraGPUComputingStressTest** (CUDALinearAlgebraGPUComputingStressTest &&)=delete
- CUDALinearAlgebraGPUComputingStressTest & **operator=** (const CUDALinearAlgebraGPUComputingStressTest &)=delete
- CUDALinearAlgebraGPUComputingStressTest & **operator=** (CUDALinearAlgebraGPUComputingStressTest &&)=delete

Private Attributes

- std::size_t **arraySize_** = 8192 * 8192
- RunTypes **runType_** = RunTypes::RUN_BOTH
- std::size_t **numberOfCPUThreads_** = 0
- std::size_t **numberOfCPUKernelIterations_** = 0
- bool **useUnifiedMemory_** = false
- bool **useCounter_** = false
- DeviceMemory< std::int32_t > **arrayA_**
- DeviceMemory< std::int32_t > **arrayB_**
- DeviceMemory< std::int32_t > **arrayC_**
- DeviceMemory< std::int32_t > **kernelExecutionCounterUVA_**
- DeviceMemory< std::int32_t > **globalSynchronizationUVA_**
- DeviceMemory< std::int32_t > **globalBarrierUVA_**
- DeviceMemory< std::int32_t > **gpuStopFlagUVA_**
- DeviceMemory< std::int32_t > **cpuStopFlagUVA_**
- HostDeviceMemory< std::int32_t > **hostDeviceArrayA_**
- HostDeviceMemory< std::int32_t > **hostDeviceArrayB_**
- HostDeviceMemory< std::int32_t > **hostDeviceArrayC_**
- HostDeviceMemory< std::int32_t > **kernelExecutionCounter_**
- HostDeviceMemory< std::int32_t > **globalSynchronization_**
- HostDeviceMemory< std::int32_t > **globalBarrier_**
- HostDeviceMemory< std::int32_t > **gpuStopFlag_**
- HostDeviceMemory< std::int32_t > **cpuStopFlag_**
- CUDAProcessMemoryPool & **cudaProcessMemoryPool_**
- const CUDAStreamsHandler **cudaStreamsHandler_**
- CUDAEventTimer **gpuTimer_**
- std::unique_ptr< std::int32_t[]> **cpuArrayC_** = nullptr

Additional Inherited Members

5.18.1 Detailed Description

This class contains a basic Linear Algebra GPU Computing stress test case in host & device.

Using the Curiously Recurring Template Pattern (CRTP).

CUDALinearAlgebraGPUComputingStressTest.h:

This class contains a basic Linear Algebra GPU Computing stress test case in host & device. Using the Curiously Recurring Template Pattern (CRTP).

Author

Thanos Theo, Amir Shahvaran, 2019

Version

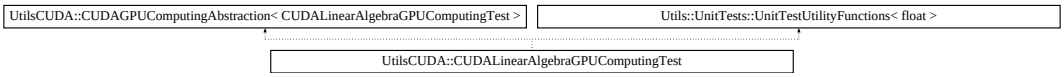
14.0.0.0

5.19 UtilsCUDA::CUDALinearAlgebraGPUComputingTest Class Reference

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

`#include <CUDALinearAlgebraGPUComputingTest.h>`

Inheritance diagram for UtilsCUDA::CUDALinearAlgebraGPUComputingTest:



Public Member Functions

- void initializeGPUMemory ()
  Initializes GPU memory.
- void performGPUComputing ()
  Performs the GPU Computing calculations.
- void retrieveGPUResults ()
  Retrieves the results from the GPU.
- bool verifyComputingResults ()
  Verifies the computing results between the CPU and the GPU.
- void releaseGPUComputingResources ()
  Releases the GPU Computing resources.
- **CUDALinearAlgebraGPUComputingTest** (const CUDADriverInfo &cudaDriverInfo, int device=0, bool use↩ UnifiedMemory=false, std::size_t arraySize=8192) noexcept
- **CUDALinearAlgebraGPUComputingTest** (const CUDALinearAlgebraGPUComputingTest &)=delete
- **CUDALinearAlgebraGPUComputingTest** (CUDALinearAlgebraGPUComputingTest &&)=delete
- CUDALinearAlgebraGPUComputingTest & **operator=** (const CUDALinearAlgebraGPUComputingTest &)=delete
- CUDALinearAlgebraGPUComputingTest & **operator=** (CUDALinearAlgebraGPUComputingTest &&)=delete

Private Attributes

- std::size_t **arraySize_** = 8192 ∗ 8192
- bool **useUnifiedMemory_** = false
- DeviceMemory< std::int32_t > **arrayA_**
- DeviceMemory< std::int32_t > **arrayB_**
- DeviceMemory< std::int32_t > **arrayC_**
- HostDeviceMemory< std::int32_t > **hostDeviceArrayA_**
- HostDeviceMemory< std::int32_t > **hostDeviceArrayB_**
- HostDeviceMemory< std::int32_t > **hostDeviceArrayC_**
- CUDAProcessMemoryPool **cudaProcessMemoryPool_**
- const CUDAStreamsHandler **cudaStreamsHandler_**
- CUDAEventTimer **gpuTimer_**

Additional Inherited Members

5.19.1 Detailed Description

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

Using the Curiously Recurring Template Pattern (CRTP).

CUDALinearAlgebraGPUComputingTest.h:

This class contains a basic Linear Algebra GPU Computing test case in CUDA. Using the Curiously Recurring Template Pattern (CRTP).

Author

Thanos Theo, 2018

Version

14.0.0.0

5.20 UtilsCUDA::CUDAMemoryPool Class Reference

This class encapsulates CUDA Memory Pool functionality for both host & device with reporting.

```
#include <CUDAMemoryPool.h>
```

Classes

- struct MemoryPoolData

  struct for Memory Pool Data

Public Types

- enum MemoryPoolTypes : std::size_t { **HOST_MEMORY** = 0, **DEVICE_MEMORY** = 1 }

  enum for Memory Pool Types

Public Member Functions

- template<typename T >
  bool addToHostMemoryPool (HostMemory< T > &hostHandler, std::size_t numberOfElements)

  Adds to the Host Memory Pool (wrapping a non-template function) via a HostMemory handle.
- template<typename T >
  bool addToDeviceMemoryPool (DeviceMemory< T > &deviceHandler, std::size_t numberOfElements, int device=0)

  Adds to the Device Memory Pool (wrapping a non-template function) via a memory handle.
- template<typename T >
  bool addToHostDeviceMemoryPool (HostDeviceMemory< T > &hostDeviceHandler, std::size_t numberOf↩Elements, int device=0)

  Adds to the Host & Device Memory Pools (wrapping a non-template function) via a memory handle.
- void allocateHostMemoryPool (const std::string &name=std::string(), unsigned int flags=cudaHostRegister↩Default)

  Registers host memory in the Host Memory Pool.
- void allocateDeviceMemoryPool (const std::string &name=std::string(), const std::bitset< MAX_DEVICES > &unifiedMemoryFlags=std::bitset< MAX_DEVICES >())

  Allocates GPU-side memory in the Device Memory Pool.
- void allocateHostDeviceMemoryPool (const std::string &name=std::string(), const std::bitset< MAX_DE↩VICES > &unifiedMemoryFlags=std::bitset< MAX_DEVICES >(), unsigned int flags=cudaHostRegister↩Default)

  Allocates CPU-side & GPU-side memory in the Host/Device Memory Pool.
- void freeHostMemoryPool ()

  Frees (de-allocates) CPU-side memory & clears all the Host Memory Pool internal data structures.
- void freeDeviceMemoryPool ()

  Frees (de-allocates) GPU-side memory & clears all the Device Memory Pool internal data structures.
- void freeHostDeviceMemoryPool ()

  Frees (de-allocates) CPU-side & GPU-side memory & clears all the Host/Device Memory Pool internal data structures.
- std::size_t getHostMemoryPoolSize () const

  Gets the Host Memory Pool size.
- std::size_t getDeviceMemoryPoolSize (int device=0) const

  Gets the Device Memory Pool size.
- std::size_t getHostMemoryPoolTotalBytes () const

  Gets the Host Memory Pool total bytes.
- std::size_t getDeviceMemoryPoolTotalBytes (int device=0) const

  Gets the Device Memory Pool total bytes.
- **CUDAMemoryPool** (const CUDADriverInfo &cudaDriverInfo, bool useSeparateAllocations=bool(GPU_FR↩AMEWORK_CUDA_MEMORY_POOL_USE_SEPARATE_ALLOCATIONS)) noexcept
- **CUDAMemoryPool** (const CUDAMemoryPool &)=delete
- **CUDAMemoryPool** (CUDAMemoryPool &&)=delete
- CUDAMemoryPool & **operator=** (const CUDAMemoryPool &)=delete
- CUDAMemoryPool & **operator=** (CUDAMemoryPool &&)=delete

Static Public Attributes

- static constexpr std::size_t **MAX_DEVICES** = 64

Private Member Functions

- bool addMemoryPoolData (std::size_t numberOfElements, std::size_t sizeOfElement, int device, const MemoryPoolTypes &type, const std::function< void(std::uint8_t *ptr, bool)> &memoryHandlerSetFunction)

  Adds state to the CUDA Memory Pool.

Private Attributes

- bool useSeparateAllocations_ = false

  Use separate memory allocations (Note: to be enabled for debugging purposes only)
- bool isHostAllocated_ = false

  The Host Memory Pool host allocated check.
- std::uint8_t * hostMemoryPoolPtr_ = nullptr

  The Host Memory Pool pointer.
- std::size_t hostBytesToAllocate_ = 0

  The total Host Memory Pool bytes consumed.
- std::vector< MemoryPoolData > hostMemoryPool_

  The Host Memory Pool is stored in a vector.
- bool isDeviceAllocated_ = false

  The Device Memory Pool device allocated check.
- std::size_t deviceCount_ = 1

  The number of available devices (default is 1)
- std::unique_ptr< std::size_t[]> textureAlignmentPerDevice_ = nullptr

  The texture alignment per device (1 per available GPU device)
- std::unique_ptr< std::uint8_t *[]> deviceMemoryPoolPtrPerDevice_ = nullptr

  The Device Memory Pool pointers (1 per available GPU device)
- std::unique_ptr< std::size_t[]> deviceBytesToAllocatePerDevice_ = nullptr

  The total Device Memory Pool bytes consumed (1 per available GPU device)
- std::vector< MemoryPoolData > deviceMemoryPool_

  The Device Memory Pool is stored in a vector.

5.20.1 Detailed Description

This class encapsulates CUDA Memory Pool functionality for both host & device with reporting.

CUDAMemoryPool.h:

This class encapsulates CUDA Memory Pool functionality for both host & device with reporting.

The work pattern to use the CUDA Memory Pool is as follows:

1. Use the addToHostMemoryPool(), addToDeviceMemoryPool() & addToHostDeviceMemoryPool() to add host/device T* data respectively to the memory pool (or together).
2. Use the allocateHostMemoryPool(), allocateDeviceMemoryPool() & allocateHostDeviceMemoryPool() for host/device batch allocations.
3. Use the freeHostMemoryPool(), freeDeviceMemoryPool() & freeHostDeviceMemoryPool() to delete all host/device data explicitly from the memory pool.

Note: 1. The destructor will also freeHostMemoryPool()/freeDeviceMemoryPool()/freeHostDeviceMemoryPool() in RAII fashion.

1. The allocateDeviceMemoryPool() supports Unified Memory allocation per device (default is off).
2. After using the allocateHostMemoryPool()/allocateDeviceMemoryPool()/allocateHostDeviceMemoryPool(), the addToHostMemoryPool()/addToDeviceMemoryPool()/addToHostDeviceMemoryPool() call is invalid (nothing added, false returned).

Author

Thanos Theo, 2019

Version

14.0.0.0

5.21 UtilsCUDA::CUDAMemoryRegistry Class Reference

This class encapsulates CUDA Memory Registry functionality for pre-allocated host memory with reporting.

`#include <CUDAMemoryRegistry.h>`

Classes

- struct MemoryRegistryData
  struct for Memory Registry Data

Public Member Functions

- void registerMemoryRegistry (const std::string &name=std::string(), unsigned int flags=cudaHostRegisterDefault)
  Registers host memory in the Memory Registry.
- void unregisterMemoryRegistry ()
  Unregisters host memory from the Memory Registry.
- template<typename T >
  bool addToMemoryRegistry (const std::string &name, T *ptr, std::size_t numberOfElements)
  Adds to the Memory Registry (wrapping a non-template function) a MemoryRegistryData.
- template<typename T >
  MemoryRegistryData getPtrTupleFromMemoryRegistry (const std::string &name) const
  Gets from the Memory Registry (wrapping a non-template function) a MemoryRegistryData.
- template<typename T >
  T * getPtrFromMemoryRegistry (const std::string &name) const
  Gets from the Memory Registry (wrapping a non-template function) a MemoryRegistryData.
- std::size_t getMemoryRegistrySize () const
  Gets the Memory Registry size.
- **CUDAMemoryRegistry** (const CUDAMemoryRegistry &)=delete
- **CUDAMemoryRegistry** (CUDAMemoryRegistry &&)=delete
- CUDAMemoryRegistry & **operator=** (const CUDAMemoryRegistry &)=delete
- CUDAMemoryRegistry & **operator=** (CUDAMemoryRegistry &&)=delete

Private Member Functions

- bool addToMemoryRegistryPtr (const std::string &name, std::uint8_t *ptr, std::size_t numberOfElements, std::size_t sizeOfElement)

  Adds to the Memory Registry a MemoryRegistryData.
- MemoryRegistryData getFromMemoryRegistryPtr (const std::string &name) const

  Gets from the Memory Registry a MemoryRegistryData.
- void reportMemoryRegistryInformation (const std::string &name=std::string()) const

  Reports information from the Memory Registry.

Private Attributes

- bool isRegistered_ = false

  The Memory Registry registered check.
- std::vector< std::string > memoryRegistryNames_

  The Memory Registry names is stored in a vector.
- std::unordered_map< std::string, MemoryRegistryData > memoryRegistry_

  The Memory Registry is stored in an unordered map.

5.21.1 Detailed Description

This class encapsulates CUDA Memory Registry functionality for pre-allocated host memory with reporting.

CUDAMemoryRegistry.h:

This class encapsulates CUDA Memory Registry functionality for pre-allocated host memory with reporting.

The work pattern to use the CUDA Memory Registry is as follows:

1. Use the addToMemoryRegistry() to register host T* data.
2. Use the registerMemoryRegistry() for host batch register (per 'name' functions also available).
3. Use the getPtrFromMemoryRegistry() to get T* the data.
4. Use the unregisterMemoryRegistry() to delete all host data (per 'name' functions also available).

Note: 1. The destructor will also unregisterMemoryRegistry() in RAII fashion.

1. Before using the registerMemoryRegistry(), using getPtrFromMemoryRegistry() call is invalid (nullptr returned).
2. After using the registerMemoryRegistry(), the addToMemoryRegistry() call is invalid (nothing added, false returned).
3. The returned getPtrFromMemoryRegistry() is invalid after a unregisterMemoryRegistry() call.

Author

Thanos Theo, 2019

Version

14.0.0.0

5.22 UtilsCUDA::CUDAProcessMemoryPool Class Reference

This class encapsulates CUDA Process Memory Pool functionality for both host & device with reporting.

`#include <CUDAProcessMemoryPool.h>`

Classes

- struct MemoryPoolData

  struct for Memory Pool Data

Public Types

- enum MemoryPoolTypes : std::size_t { **HOST_MEMORY** = 0, **DEVICE_MEMORY** = 1 }

  enum for Memory Pool Types

Public Member Functions

- void allocateHostMemoryPool (std::size_t hostBytesToAllocate=0, unsigned int flags=cudaHostRegisterDefault)

  Registers host memory in the Host Memory Pool.
- void allocateDeviceMemoryPool (const std::array< std::size_t, MAX_DEVICES > &deviceBytesToAllocatePerDevice=std::array< std::size_t, MAX_DEVICES >(), const std::bitset< MAX_DEVICES > &unifiedMemoryFlags=std::bitset< MAX_DEVICES >())

  Allocates GPU-side memory in the Device Memory Pool.
- void allocateHostDeviceMemoryPool (std::size_t hostBytesToAllocate=0, const std::array< std::size_t, MAX_DEVICES > &deviceBytesToAllocatePerDevice=std::array< std::size_t, MAX_DEVICES >(), const std::bitset< MAX_DEVICES > &unifiedMemoryFlags=std::bitset< MAX_DEVICES >(), unsigned int flags=cudaHostRegisterDefault)

  Allocates CPU-side & GPU-side memory in the Host/Device Memory Pool.
- template<typename T >
  bool reserve (HostMemory< T > &hostHandler, std::size_t numberOfElements)

  Reserves in the Host Memory Pool (wrapping a non-template function) via a HostMemory handle.
- template<typename T >
  bool reserve (DeviceMemory< T > &deviceHandler, std::size_t numberOfElements, int device=0)

  Reserves in the Device Memory Pool (wrapping a non-template function) via a DeviceMemory handle.
- template<typename T >
  bool reserve (HostDeviceMemory< T > &hostDeviceHandler, std::size_t numberOfElements, int device=0)

  Reserves in to the Host & Device Memory Pool (wrapping a non-template function) via a HostDeviceMemory handle.
- void freeHostMemoryPool ()

  Frees (de-allocates) CPU-side memory & clears all the Host Memory Pool internal data structures.
- void freeDeviceMemoryPool ()

  Frees (de-allocates) GPU-side memory & clears all the Device Memory Pool internal data structures.
- void freeHostDeviceMemoryPool ()

  Frees (de-allocates) CPU-side & GPU-side memory & clears all the Host/Device Memory Pool internal data structures.
- std::size_t getHostMemoryPoolSize () const

  Gets the Host Memory Pool size.
- std::size_t getDeviceMemoryPoolSize (int device=0) const

  Gets the Device Memory Pool size.
- std::size_t getHostMemoryPoolTotalBytes () const

Gets the Host Memory Pool total bytes.

- std::size_t getDeviceMemoryPoolTotalBytes (int device=0) const

  Gets the Device Memory Pool total bytes.
- void reportHostMemoryPoolInformation (const std::string &name=std::string()) const

  Reports information from the Host Memory Pool.
- void reportDeviceMemoryPoolInformation (const std::string &name=std::string()) const

  Reports information from the Device Memory Pool.
- void reportHostDeviceMemoryPoolInformation (const std::string &name=std::string()) const

  Reports information from the Host/Device Memory Pool.
- **CUDAProcessMemoryPool** (const CUDADriverInfo &cudaDriverInfo, bool useDefaultAllocations=true, bool useSeparateAllocations=bool(GPU_FRAMEWORK_CUDA_PROCESS_MEMORY_POOL_USE_SEPAR↩ ATE_ALLOCATIONS)) noexcept
- **CUDAProcessMemoryPool** (const CUDAProcessMemoryPool &)=delete
- **CUDAProcessMemoryPool** (CUDAProcessMemoryPool &&)=delete
- CUDAProcessMemoryPool & **operator=** (const CUDAProcessMemoryPool &)=delete
- CUDAProcessMemoryPool & **operator=** (CUDAProcessMemoryPool &&)=delete

Static Public Attributes

- static constexpr std::size_t **MAX_DEVICES** = 64

Private Member Functions

- bool reserveMemoryPoolData (std::size_t numberOfElements, std::size_t sizeOfElement, int device, const MemoryPoolTypes &type)

  Reserves state to the CUDA Memory Pool.

Private Attributes

- bool useDefaultAllocations_ = false

  Use default memory allocations for all devices.
- bool useSeparateAllocations_ = false

  Use separate memory allocations (Note: to be enabled for debugging purposes only)
- const CUDADriverInfo & cudaDriverInfo_

  The CUDADriverInfo reference.
- bool isHostAllocated_ = false

  The Host Memory Pool host allocated check.
- std::uint8_t * hostMemoryPoolPtr_ = nullptr

  The Host Memory Pool pointer.
- std::size_t hostMemoryPoolOffset_ = 0

  The Host Memory Pool offset.
- std::size_t hostBytesToAllocate_ = 0

  The total Host Memory Pool bytes consumed.
- std::vector< MemoryPoolData > hostMemoryPool_

  The Host Memory Pool is stored in a vector.
- unsigned int flags_ = cudaHostRegisterDefault

  The Host Memory Pool flags.
- bool isDeviceAllocated_ = false

  The Device Memory Pool device allocated check.

- std::size_t deviceCount_ = 1

  The number of available devices (default is 1)
- std::unique_ptr< std::size_t[ ]> textureAlignmentPerDevice_ = nullptr

  The texture alignment per device (1 per available GPU device)
- std::unique_ptr< std::uint8_t *[ ]> deviceMemoryPoolPtrPerDevice_ = nullptr

  The Device Memory Pool pointers (1 per available GPU device)
- std::unique_ptr< std::size_t[ ]> deviceMemoryPoolOffsetPerDevice_ = nullptr

  The Device Memory Pool offsets (1 per available GPU device)
- std::unique_ptr< std::size_t[ ]> deviceBytesToAllocatePerDevice_ = nullptr

  The total Device Memory Pool bytes consumed (1 per available GPU device)
- std::vector< MemoryPoolData > deviceMemoryPool_

  The Device Memory Pool is stored in a vector.
- std::bitset< MAX_DEVICES > unifiedMemoryFlags_

  The Device Memory Pool Unified Memory (UVA) flags.

5.22.1 Detailed Description

This class encapsulates CUDA Process Memory Pool functionality for both host & device with reporting.

CUDAProcessMemoryPool.h:

This class encapsulates CUDA Process Memory Pool functionality for both host & device with reporting.

The work pattern to use the CUDA Process Memory Pool is as follows:

1. Use the allocateHostMemoryPool(), allocateDeviceMemoryPool() & allocateHostDeviceMemoryPool() for host/device batch allocations.
2. Use the reserve() calls to reserve host/device T* data from the memory pool.
3. Use the freeHostMemoryPool(), freeDeviceMemoryPool() & freeHostDeviceMemoryPool() to delete all host/device data explicitly from the memory pool.

Note: 1. The destructor will also freeHostMemoryPool()/freeDeviceMemoryPool()/freeHostDeviceMemoryPool() in RAII fashion.

1. The allocateDeviceMemoryPool() supports Unified Memory allocation per device (default is off).
2. Before using the allocateHostMemoryPool()/allocateDeviceMemoryPool()/allocateHostDeviceMemoryPool(), all the reserve() calls are invalid (nothing added, false returned).

Author

Thanos Theo, 2019

Version

14.0.0.0

5.23 UtilsCUDA::CUDAQueue< T > Class Template Reference

CUDAQueue class for GPU memory.

`#include <CUDAQueue.h>`

Public Member Functions

- **CUDAQueue** (DeviceMemory< T > &buffer, std::size_t bufferSize)
- CUDAQueue (std::size_t bufferSize, int device=0, bool useUnifiedMemory=false)

  Second constructor for the CUDAQueue: with an owned storage buffer allocated directly, optionally on a given device & with Unified Memory enabled.
- void push_back (const DeviceMemory< T > &elements, std::size_t length)

  Copies.
- std::size_t front (DeviceMemory< T > &dst, std::size_t length) const

  Copies elements from begin of the CUDAQueue to.
- std::size_t pop_front (std::size_t length)

  Removes no more than.
- std::size_t size () const

  Number of elements in the CUDAQueue.
- bool empty () const

  Check if the CUDAQueue is empty.
- std::size_t reserved () const

  Maximum size of the CUDAQueue.
- CUDAQueueView< T > view () const

  A CUDAQueueView to access CUDAQueue elements in sequential order.

Private Attributes

- DeviceMemory< T > **ownedStorage_**
- DeviceMemory< T > & **buffer_**
- std::size_t **bufferSize_** = 0
- std::size_t **actualSize_** = 0
- std::size_t **offset_** = 0

5.23.1 Detailed Description

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

CUDAQueue class for GPU memory.

CUDAQueueView to linearly access elements in the CUDAQueue class.

It uses a pre-allocated memory buffer to avoid expensive memory allocations and de-allocations. Elements are stored as a ring buffer (also known as a circular buffer).

Template Parameters

T	- should be a Plain Old Data (POD) type

Author

Leonid Volnin, 2019

Version

14.0.0.0

Template Parameters

T	see

Author

Leonid Volnin, 2019

Version

14.0.0.0

5.23.2 Constructor & Destructor Documentation

5.23.2.1 CUDAQueue()

```
template<typename T >
UtilsCUDA::CUDAQueue< T >::CUDAQueue (
            std::size_t bufferSize,
            int device = 0,
            bool useUnifiedMemory = false )  [inline], [explicit]
```

Second constructor for the CUDAQueue: with an owned storage buffer allocated directly, optionally on a given device & with Unified Memory enabled.

Parameters

bufferSize	size of the storage buffer. It is the user's responsibility to allocate enough memory to store all data.
device	optional parameter to use a given device.
useUnifiedMemory	optional parameter to enable Unified Memory.

5.23.3 Member Function Documentation

5.23.3.1 front()

```
template<typename T >
std::size_t UtilsCUDA::CUDAQueue< T >::front (
            DeviceMemory< T > & dst,
            std::size_t length ) const  [inline]
```

Copies elements from begin of the CUDAQueue to.

Parameters

dst	buffer. If the CUDAQueue has less than
length	elements, copies only what the CUDAQueue has.
dst	destination buffer
length	number of elements to copy

Returns

number of elements that were copied

5.23.3.2 pop_front()

```
template<typename T >
std::size_t UtilsCUDA::CUDAQueue< T >::pop_front (
            std::size_t length )  [inline]
```

Removes no more than.

Parameters

length	elements from begin of CUDAQueue.
length	

Returns

number of actual elements removed

5.23.3.3 push_back()

```
template<typename T >
void UtilsCUDA::CUDAQueue< T >::push_back (
            const DeviceMemory< T > & elements,
            std::size_t length )  [inline]
```

Copies.

Parameters

elements	to the end of the CUDAQueue.
elements	
length	number of elements to copy

5.24 CUDAQueue Class Reference

First constructor for the CUDAQueue: with a given buffer.

5.24.1 Detailed Description

First constructor for the CUDAQueue: with a given buffer.

Note: Due to two step initialization of device memory allocation via the CUDA memory pool (see dotred_utils_↩ cuda::CUDAMemoryPool), it is necessary to pass the buffer size because memory may not be allocated yet. This should be updated once the memory pool allocation strategy changes.

Parameters

buffer	storage for CUDAQueue elements. Stored as a reference, so it should have longer lifetime than
bufferSize	size of the storage buffer. It is the user's responsibility to allocate enough memory to store all data.

5.25 UtilsCUDA::CUDAQueueView< T > Class Template Reference

Public Member Functions

- **CUDAQueueView** (const T *ptr, std::size_t offset, std::size_t length)
- __forceinline__ __host__ __device__ T & **operator[ ]** (std::size_t index)
- __forceinline__ __host__ __device__ const T & **operator[ ]** (std::size_t index) const

Private Attributes

- const T * **devicePtr_** = nullptr
- std::size_t **offset_** = 0
- std::size_t **length_** = 0

5.26 UtilsCUDA::CUDASpinLock< T > Class Template Reference

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

`#include <CUDASpinLock.h>`

Public Member Functions

- __forceinline__ __device__ void **acquireBlock** ()
- __forceinline__ __device__ void **releaseBlock** ()
- __forceinline__ __device__ void **acquireGrid** ()
- __forceinline__ __device__ void **releaseGrid** ()
- __forceinline__ __device__ **CUDASpinLock** (T *sequencer, T *barrier=nullptr) noexcept
- **CUDASpinLock** (const CUDASpinLock &)=delete
- **CUDASpinLock** (CUDASpinLock &&)=delete
- CUDASpinLock & **operator=** (const CUDASpinLock &)=delete
- CUDASpinLock & **operator=** (CUDASpinLock &&)=delete

Private Attributes

- T * **sequencer_** = nullptr
- T * **barrier_** = nullptr

5.26.1 Detailed Description

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

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 CUDA threads 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.

Note 1: the CUDASpinLock::acquireBlock() function below is prone to deadlock if used for INTRABLOCK synchronization. Expected usage is for one thread in each block to attempt to acquireBlock() 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.

Note 2: the CUDASpinLock::acquireGrid() function below is prone to deadlock if used for INTRAGRID synchronization. Expected usage is for one thread in each block to attempt to acquireGrid() the CUDASpinLock, otherwise the divergent code execution tends to deadlock. Additionally, it also NEEDS a persistent kernel to use this way for grid synchronization, ie all blocks to be executing for this to work. See the CUDA Stress Test for a workable use case. The modern sanctioned way is to use CUDA 9.0 Cooperative Groups, ie <cooperative_groups.h>.

Example code for INTRABLOCK synchronization usage: **forceinline device** void sumDoubles(double* pSum, int* spinlock, const double* in, size_t N, int* acquireCount) { SharedMemory<double> shared; CUDASpinLock<int> 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 = reduceBlock<double, double>(); __syncthreads();

if (threadIdx.x == 0) { globalSpinlock.acquireBlock(); *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.↩ releaseBlock(); } } }

Author

Thanos Theo, Amir Shahvaran, 2019

Version

14.0.0.0

5.27 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
- void **addCallback** (std::size_t index, const cudaStreamCallback_t &callback, void *data) const 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** (int device) noexcept
- void **uninitialize** () const noexcept

Private Attributes

- std::size_t **numberOfStreams_** = 0
- std::unique_ptr< cudaStream_t[ ]> **cudaStreams_** = nullptr
- bool **useStreamPriorities_** = false
- int **priorityType_** = cudaStreamNonBlocking
- int **priorityHighest_** = 0
- int **priorityLowest_** = 0

5.27.1 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

5.28 UtilsCUDA::CUDAUtilityDeviceFunctions Class Reference

This class encapsulates all the CUDA related device only utility functions.

`#include <CUDAUtilityDeviceFunctions.h>`

Public Member Functions

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

Static Public Member Functions

- static __forceinline__ __device__ std::uint32_t globalThreadCount ()

  globalThreadCount() calculates the total number of threads in the running grid.
- static __forceinline__ __device__ uint3 globalIndex ()

  globalIdx() calculates the global index of a given thread.
- static __forceinline__ __device__ std::uint32_t linearIndex (const uint3 &index, const dim3 &dimension)

  globalLinearIdx() calculates the linear order of a given thread based on given dimension.
- static __forceinline__ __device__ std::uint32_t globalLinearIndex ()

  globalLinearIdx() calculates the global linear order of a given thread.
- static __forceinline__ __device__ float atomicMin (float *address, float value)

  Device atomic min for floats.
- static __forceinline__ __device__ float atomicMax (float *address, float value)

  Device atomic max for floats.
- static __forceinline__ __device__ double atomicMin (double *address, double value)

  Device atomic min for doubles.
- static __forceinline__ __device__ double atomicMax (double *address, double value)

  Device atomic max for doubles.

Static Private Member Functions

- template<typename ComparisonOp >
  static __forceinline__ __device__ float atomicArithmeticOpFloat (float *address, float value, ComparisonOp comparisonOp)

  atomicArithmeticOpFloat() for floats
- template<typename ComparisonOp >
  static __forceinline__ __device__ double atomicArithmeticOpDouble (double *address, double value, ComparisonOp comparisonOp)

  atomicArithmeticOpDouble() for doubles

5.28.1 Detailed Description

This class encapsulates all the CUDA related device only utility functions.

NOTE: *** These do NOT work for shared-memory atomics at this time! *** More information: https://github.com/treecode/Bonsai/blob/master/runtime/profiling/derived_atomic_functions.h

Author

Thanos Theo, 2019

Version

14.0.0.0

5.28.2 Member Function Documentation

5.28.2.1 atomicArithmeticOpDouble()

```
template<typename ComparisonOp >
static __forceinline__ __device__ double UtilsCUDA::CUDAUtilityDeviceFunctions::atomicArithmetic↩
OpDouble (
            double * address,
            double value,
            ComparisonOp comparisonOp )  [inline], [static], [private]
```

atomicArithmeticOpDouble() for doubles

For all float & double atomics below: Must do the compare with integers, not floating point, since NaN is never equal to any other NaN

NOTE: *** These do NOT work for shared-memory atomics at this time! *** More information: https↩://github.com/treecode/Bonsai/blob/master/runtime/profiling/derived_atomic↩_functions.h

Author

Thanos Theo, 2019

5.28.2.2 atomicArithmeticOpFloat()

```
template<typename ComparisonOp >
static __forceinline__ __device__ float UtilsCUDA::CUDAUtilityDeviceFunctions::atomicArithmetic↩
OpFloat (
            float * address,
            float value,
            ComparisonOp comparisonOp )  [inline], [static], [private]
```

atomicArithmeticOpFloat() for floats

For all float & double atomics below: Must do the compare with integers, not floating point, since NaN is never equal to any other NaN

NOTE: *** These do NOT work for shared-memory atomics at this time! *** More information: https↩://github.com/treecode/Bonsai/blob/master/runtime/profiling/derived_atomic↩_functions.h

Author

Thanos Theo, 2019

5.28.2.3 globalIndex()

```
static __forceinline__ __device__ uint3 UtilsCUDA::CUDAUtilityDeviceFunctions::globalIndex ( )
[inline], [static]
```

globalIdx() calculates the global index of a given thread.

Author

Amir Shahvarani, 2019

5.28.2.4 globalLinearIndex()

```
static __forceinline__ __device__ std::uint32_t UtilsCUDA::CUDAUtilityDeviceFunctions::global↩
LinearIndex ( )  [inline], [static]
```

globalLinearIdx() calculates the global linear order of a given thread.

Author

Amir Shahvarani, 2019

5.28.2.5 globalThreadCount()

```
static __forceinline__ __device__ std::uint32_t UtilsCUDA::CUDAUtilityDeviceFunctions::global↩
ThreadCount ( )  [inline], [static]
```

globalThreadCount() calculates the total number of threads in the running grid.

Author

Amir Shahvarani, 2019

5.28.2.6 linearIndex()

```
static __forceinline__ __device__ std::uint32_t UtilsCUDA::CUDAUtilityDeviceFunctions::linear↩
Index (
            const uint3 & index,
            const dim3 & dimension )  [inline], [static]
```

globalLinearIdx() calculates the linear order of a given thread based on given dimension.

NOTE: index must be within the boundaries of dimension.

Author

Amir Shahvarani, 2019

5.29 UtilsCUDA::CUDAUtilityFunctions Struct Reference

This struct 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 __forceinline__ __host__ __device__ void **printfCUDAImpl** (const char *format, Args... args)
- template<typename T >
  static __forceinline__ __host__ __device__ bool equal (const T left, const T right, std::enable_if_t< std::is_floating_point< T >::value > *=nullptr)

  GLSL-style equal function.
- template<typename T >
  static __forceinline__ __host__ __device__ T **sign** (const T x, std::enable_if_t< std::is_floating_point< T >::value > *=nullptr)
- template<typename T >
  static __forceinline__ __host__ __device__ T fract (T x, std::enable_if_t< std::is_same< T, float >::value > *=nullptr)

  GLSL-style fract function (float version).
- template<typename T >
  static __forceinline__ __host__ __device__ T fract (T x, std::enable_if_t< std::is_same< T, double >::value > *=nullptr)

  GLSL-style fract function (double version).
- template<typename T >
  static __forceinline__ __host__ __device__ 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 __forceinline__ __host__ __device__ T toDegrees (const T radians, std::enable_if_t< std::is_floating_point< T >::value > *=nullptr)

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

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

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

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

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

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

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

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

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

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

  This function returns uniformly distributed double4 values in the range [0, 1] (double version).
- template<std::uint32_t N = 16>
  static __forceinline__ __host__ __device__ std::uint32_t seedGenerator (std::uint32_t value0, std::uint32_t value1)

  Seed generator for the Linear Congruential Generator (LGC).
- static __forceinline__ __host__ __device__ 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 __forceinline__ __host__ __device__ float rand1f (std::uint32_t &seed)

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

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

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

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

  This function is the GPU version checkAbsoluteError to be used with CUDA.
- template<typename T >
  static bool checkRelativeError (T a, T b, T relativeError)

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

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

  Get the uint32 bit representation to a float32.
- static __forceinline__ __host__ __device__ 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 __forceinline__ __host__ __device__ 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 __forceinline__ __host__ __device__ std::uint64_t asUint64 (double value)

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

  Get the uint64 bit representation to a float64.
- static __forceinline__ __host__ __device__ 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 __forceinline__ __host__ __device__ 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.
- template<std::size_t EXPONENT, typename T >
  static __forceinline__ __host__ __device__ T pow (T value, std::enable_if_t< std::is_arithmetic< T >::value > *=nullptr)

  Templatized pow<EXPONENT>(T) function with 1 base case needed: pow<0> where T is an arithmetic primitive type.
- static __forceinline__ __host__ __device__ void * memset (void *__restrict ptr, std::uint8_t value, std::size_t length)

  Fast memset implementation with alignment considered.
- static std::uint8_t **getCUDAExitCodeOffset** ()
- static void **checkCUDAErrorImpl** (const cudaError_t &errnum, const char *file, const char *function, int line, bool abort=true)
- static void **checkCUDAErrorImpl** (const CUresult &errnum, const char *file, const char *function, int line, bool abort=true)
- static void **checkCUDAErrorImpl** (const curandStatus_t &errnum, const char *file, const char *function, int line, bool abort=true)
- static std::uint32_t **getWarpSize** ()
- static dim3 **getDefaultThreads1DDimensions** ()
- static dim3 **getDefaultThreads2DDimensions** ()
- static std::tuple< dim3, dim3 > calculateCUDA1DKernelDimensions (std::size_t arraySize, const dim3 &threads1D=getDefaultThreads1DDimensions())

The calculateCUDA2DKernelDimensions() function efficiently calculates the dimensions for a CUDA 1D kernel.

- static std::tuple< dim3, dim3 > calculateCUDA2DKernelDimensions (std::size_t arraySize, const dim3 &threads2D=getDefaultThreads2DDimensions())

  The calculateCUDA2DKernelDimensions() function efficiently calculates the (non power-of-two) square dimensions for a CUDA 2D kernel.
- static std::tuple< dim3, dim3 > calculateCUDA2DKernelDimensionsXY (std::size_t arraySizeX, std::size_t arraySizeY, const dim3 &threads2D=getDefaultThreads2DDimensions())

  The calculateCUDA2DKernelDimensionsXY() function efficiently calculates the XY dimensions for a CUDA 2D kernel.
- static std::tuple< dim3, dim3 > calculateCUDAPersistentKernel (const CUDADriverInfo &cudaDriverInfo, int device, uint32_t threadsPerBlock, uint32_t sharedMemoryPerBlock=0)

  The calculateCUDAPersistentKernel() function efficiently calculates the dimensions of persistent kernel to run on current device.
- static std::string checkAndReportCUDAMemory (int device, bool useUVA=false)

  The checkAndReportCUDAMemory() function checks & reports CUDA memory per given device.

5.29.1 Detailed Description

This struct encapsulates all the CUDA related utility functions.

Author

Thanos Theo, 2018

Version

14.0.0.0

5.29.2 Member Function Documentation

5.29.2.1 asFloat32()

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

Get the uint32 bit representation to a float32.

Author

Thanos Theo, 2018

5.29.2.2 asFloat64()

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

Get the uint64 bit representation to a float64.

Author

Thanos Theo, 2018

5.29.2.3 asUint32()

```
static __forceinline__ __host__ __device__ std::uint32_t UtilsCUDA::CUDAUtilityFunctions::as↩
Uint32 (
            float value )  [inline], [static]
```

Get the float32 bit representation to a uint32.

Author

Thanos Theo, 2018

5.29.2.4 asUint64()

```
static __forceinline__ __host__ __device__ std::uint64_t UtilsCUDA::CUDAUtilityFunctions::as↩
Uint64 (
            double value )  [inline], [static]
```

Get the float64 bit representation to a uint64.

Author

Thanos Theo, 2018

5.29.2.5 calculateCUDA1DKernelDimensions()

```
tuple< dim3, dim3 > CUDAUtilityFunctions::calculateCUDA1DKernelDimensions (
            std::size_t arraySize,
            const dim3 & threads1D = getDefaultThreads1DDimensions() )  [static]
```

The calculateCUDA2DKernelDimensions() function efficiently calculates the dimensions for a CUDA 1D kernel.

Author

Amir Shahvarani, 2019

Version

14.0.0.0

5.29.2.6 calculateCUDA2DKernelDimensions()

```
tuple< dim3, dim3 > CUDAUtilityFunctions::calculateCUDA2DKernelDimensions (
            std::size_t arraySize,
            const dim3 & threads2D = getDefaultThreads2DDimensions() )  [static]
```

The calculateCUDA2DKernelDimensions() function efficiently calculates the (non power-of-two) square dimensions for a CUDA 2D kernel.

Author

Thanos Theo, 2019

Version

14.0.0.0

5.29.2.7 calculateCUDA2DKernelDimensionsXY()

```
tuple< dim3, dim3 > CUDAUtilityFunctions::calculateCUDA2DKernelDimensionsXY (
            std::size_t arraySizeX,
            std::size_t arraySizeY,
            const dim3 & threads2D = getDefaultThreads2DDimensions() )  [static]
```

The calculateCUDA2DKernelDimensionsXY() function efficiently calculates the XY dimensions for a CUDA 2D kernel.

Author

Amir Shahvarani, 2019

Version

14.0.0.0

5.29.2.8 calculateCUDAPersistentKernel()

```
tuple< dim3, dim3 > CUDAUtilityFunctions::calculateCUDAPersistentKernel (
            const CUDADriverInfo & cudaDriverInfo,
            int device,
            uint32_t threadsPerBlock,
            uint32_t sharedMemoryPerBlock = 0 )  [static]
```

The calculateCUDAPersistentKernel() function efficiently calculates the dimensions of persistent kernel to run on current device.

Author

Amir Shahvarani, 2019

Version

14.0.0.0

5.29.2.9 checkAbsoluteError()

```
template<typename T >
static __forceinline__ __host__ __device__ bool UtilsCUDA::CUDAUtilityFunctions::checkAbsolute↩
Error (
            T a,
            T b,
            T error )  [inline], [static]
```

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

Author

Thanos Theo, 2018

5.29.2.10 checkAndReportCUDAMemory()

```
string CUDAUtilityFunctions::checkAndReportCUDAMemory (
            int device,
            bool useUVA = false )  [static]
```

The checkAndReportCUDAMemory() function checks & reports CUDA memory per given device.

Author

Thanos Theo, 2019

Version

14.0.0.0

5.29.2.11 checkRelativeError()

```
template<typename T >
static bool UtilsCUDA::CUDAUtilityFunctions::checkRelativeError (
            T a,
            T b,
            T relativeError )  [inline], [static]
```

This function is the GPU version checkRelativeError to be used with CUDA.

Author

Thanos Theo, 2018

5.29.2.12 float32Flip()

```
static __forceinline__ __host__ __device__ std::uint32_t 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

5.29.2.13 float32Unflip()

```
static __forceinline__ __host__ __device__ float UtilsCUDA::CUDAUtilityFunctions::float32↩
Unflip (
            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

5.29.2.14 float64Flip()

```
static __forceinline__ __host__ __device__ std::uint64_t 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

5.29.2.15 float64Unflip()

```
static __forceinline__ __host__ __device__ double UtilsCUDA::CUDAUtilityFunctions::float64↩
Unflip (
            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

5.29.2.16 memset()

```
static __forceinline__ __host__ __device__ void* UtilsCUDA::CUDAUtilityFunctions::memset (
            void *__restrict ptr,
            std::uint8_t value,
            std::size_t length )  [inline], [static]
```

Fast memset implementation with alignment considered.

For use in both host & device code.

Note: This memset variant is meant to be used within device code and not to replace the cudaMemset() kernel call. For the CPU side, we assume that std::memset() call is already optimal and probably vectorized. We provide below the **host** variant for verification purposes.

Author

Leonid Volnin, 2019

Version

14.0.0.0

5.29.2.17 pow()

```
template<std::size_t EXPONENT, typename T >
static __forceinline__ __host__ __device__ T UtilsCUDA::CUDAUtilityFunctions::pow (
            T value,
            std::enable_if_t< std::is_arithmetic< T >::value > *  = nullptr )  [inline],
[static]
```

Templatized pow<EXPONENT>(T) function with 1 base case needed: pow<0> where T is an arithmetic primitive type.

Author

Ben Dart, Amir Shahvarani, Inaki Pujol, Thanos Theo, 2019

5.29.2.18 rand1() [1/2]

```
static __forceinline__ __host__ __device__ float 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

5.29.2.19 rand1() [2/2]

```
static __forceinline__ __host__ __device__ double 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

5.29.2.20 rand1f()

```
static __forceinline__ __host__ __device__ float 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

5.29.2.21 rand1u()

```
static __forceinline__ __host__ __device__ std::uint32_t 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

5.29.2.22 rand2() [1/2]

```
static __forceinline__ __host__ __device__ float2 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

5.29.2.23 rand2() [2/2]

```
static __forceinline__ __host__ __device__ double2 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

5.29.2.24 rand2f()

```
static __forceinline__ __host__ __device__ float2 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

5.29.2.25 rand3() [1/2]

```
static __forceinline__ __host__ __device__ float3 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

5.29.2.26 rand3() [2/2]

```
static __forceinline__ __host__ __device__ double3 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

5.29.2.27 rand3f()

```
static __forceinline__ __host__ __device__ float3 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

5.29.2.28 rand4() [1/2]

```
static __forceinline__ __host__ __device__ float4 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

5.29.2.29 rand4() [2/2]

```
static __forceinline__ __host__ __device__ double4 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

5.29.2.30 rand4f()

```
static __forceinline__ __host__ __device__ float4 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

5.29.2.31 seedGenerator()

```
template<std::uint32_t N = 16>
static __forceinline__ __host__ __device__ std::uint32_t UtilsCUDA::CUDAUtilityFunctions↩
::seedGenerator (
            std::uint32_t value0,
            std::uint32_t value1 )  [inline], [static]
```

Seed generator for the Linear Congruential Generator (LGC).

Note: default loop (for unrolling) with value of 16.

Author

Thanos Theo, 2018

5.30 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::recursive_mutex & **getPrintMutex** ()
- static std::string **getLogFileName** ()
- static bool **getUseLogFile** ()

5.30.1 Detailed Description

The DebugConsole class provides debugging & logging functionality.

Author

Thanos Theo, 2009-2018

Version

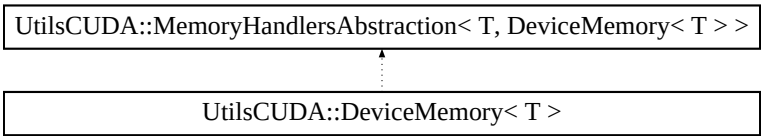
14.0.0.0

5.31 UtilsCUDA::DeviceMemory< T > Class Template Reference

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

`#include <CUDAMemoryHandlers.h>`

Inheritance diagram for UtilsCUDA::DeviceMemory< T >:



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 **reset** () noexcept
- std::future< void > **resetAsync** () noexcept
- void **swap** (DeviceMemory< T > &other) noexcept
- T * **get** ()
- const T * **get** () const
- T * **device** ()
- const T * **device** () const
- T & **operator[ ]** (std::size_t index)
- const T & **operator[ ]** (std::size_t index) const
- **operator bool** () const noexcept
- std::size_t **getNumberOfElements** () const
- bool **isMemoryPoolMode** () const
- **DeviceMemory** (std::size_t numberOfElements, int device=0, bool useUnifiedMemory=false)
- **DeviceMemory** (const DeviceMemory &)=delete
- **DeviceMemory** (DeviceMemory &&)=delete
- DeviceMemory & **operator=** (const DeviceMemory &)=delete
- DeviceMemory & **operator=** (DeviceMemory &&)=delete

- void memset (int value) noexcept

  Function overloads use the class member numberOfElements and copy all.
- void **memsetAsync** (int value, const cudaStream_t &stream) noexcept
- void **copyHostToDevice** (const void *hostPtr) noexcept
- void **copyHostToDeviceAsync** (const void *hostPtr, const cudaStream_t &stream) noexcept
- void **copyDeviceToDevice** (const void *devicePtr) noexcept
- void **copyDeviceToDeviceAsync** (const void *devicePtr, const cudaStream_t &stream) noexcept
- void **copyDeviceFromDevice** (void *devicePtr) const noexcept
- void **copyDeviceFromDeviceAsync** (void *devicePtr, const cudaStream_t &stream) const noexcept
- void **copyDeviceToHost** (void *hostPtr) const noexcept
- void copyDeviceToHost (void *hostPtr, const cudaStream_t &stream) const noexcept

  Note: asynchronous memcpy on the given stream, enforce further synchronization with device -> host copy for that stream.
- void **copyDeviceToHostAsync** (void *hostPtr, const cudaStream_t &stream) const noexcept
- void **memPrefetch** (int dstDevice) const noexcept
- void **memPrefetchAsync** (int dstDevice, const cudaStream_t &stream) const noexcept
- void **memPrefetchWithAdvise** (int dstDevice) const noexcept
- void **memPrefetchWithAdviseAsync** (int dstDevice, const cudaStream_t &stream) const noexcept

- void memset (int value, std::size_t numberOfElements) noexcept

  Function overloads with a given numberOfElements as function parameter.
- void **memsetAsync** (int value, std::size_t numberOfElements, const cudaStream_t &stream) noexcept
- void **copyHostToDevice** (const void *hostPtr, std::size_t numberOfElements) noexcept
- void **copyHostToDeviceAsync** (const void *hostPtr, std::size_t numberOfElements, const cudaStream_t &stream) noexcept
- void **copyDeviceToDevice** (const void *devicePtr, std::size_t numberOfElements) noexcept
- void **copyDeviceToDeviceAsync** (const void *devicePtr, std::size_t numberOfElements, const cudaStream_t &stream) noexcept
- void **copyDeviceFromDevice** (void *devicePtr, std::size_t numberOfElements) const noexcept
- void **copyDeviceFromDeviceAsync** (void *devicePtr, std::size_t numberOfElements, const cudaStream_t &stream) const noexcept
- void **copyDeviceToHost** (void *hostPtr, std::size_t numberOfElements) const noexcept
- void copyDeviceToHost (void *hostPtr, std::size_t numberOfElements, const cudaStream_t &stream) const noexcept

  Note: asynchronous memcpy on the given stream, enforce further synchronization with device -> host copy for that stream.
- void **copyDeviceToHostAsync** (void *hostPtr, std::size_t numberOfElements, const cudaStream_t &stream) const noexcept
- void **memPrefetch** (std::size_t numberOfElements, int dstDevice) const noexcept
- void **memPrefetchAsync** (std::size_t numberOfElements, int dstDevice, const cudaStream_t &stream) const noexcept
- void **memPrefetchWithAdvise** (std::size_t numberOfElements, int dstDevice) const noexcept
- void **memPrefetchWithAdviseAsync** (std::size_t numberOfElements, int dstDevice, const cudaStream_t &stream) const noexcept

- void memsetAsBytes (int value, std::size_t numberOfBytes) noexcept

  below are function overloads with a given number of bytes as function parameter
- void **memsetAsBytesAsync** (int value, std::size_t numberOfBytes, const cudaStream_t &stream) noexcept
- void **copyHostToDeviceAsBytes** (const void *hostPtr, std::size_t numberOfBytes) noexcept
- void **copyHostToDeviceAsBytesAsync** (const void *hostPtr, std::size_t numberOfBytes, const cudaStream_t &stream) noexcept
- void **copyDeviceToDeviceAsBytes** (const void *devicePtr, std::size_t numberOfBytes) noexcept
- void **copyDeviceToDeviceAsBytesAsync** (const void *devicePtr, std::size_t numberOfBytes, const cudaStream_t &stream) noexcept

- void **copyDeviceFromDeviceAsBytes** (void *devicePtr, std::size_t numberOfBytes) const noexcept
- void **copyDeviceFromDeviceAsBytesAsync** (void *devicePtr, std::size_t numberOfBytes, const cuda↩ Stream_t &stream) const noexcept
- void **copyDeviceToHostAsBytes** (void *hostPtr, std::size_t numberOfBytes) const noexcept
- void copyDeviceToHostAsBytes (void *hostPtr, std::size_t numberOfBytes, const cudaStream_t &stream) const noexcept

  Note: asynchronous memcpy on the given stream, enforce further synchronization with device -> host copy for that stream.
- void **copyDeviceToHostAsBytesAsync** (void *hostPtr, std::size_t numberOfBytes, const cudaStream_↩ t &stream) const noexcept
- void **memPrefetchAsBytes** (std::size_t numberOfBytes, int dstDevice) const noexcept
- void **memPrefetchAsBytesAsync** (std::size_t numberOfBytes, int dstDevice, const cudaStream_t &stream) const noexcept
- void **memPrefetchWithAdviseAsBytes** (std::size_t numberOfBytes, int dstDevice) const noexcept
- void **memPrefetchWithAdviseAsBytesAsync** (std::size_t numberOfBytes, int dstDevice, const cuda↩ Stream_t &stream) const noexcept

- operator RawDeviceMemory< T > ()

  Convenience functions to pass pointers to kernels.
- **operator RawDeviceMemory< const T >** () const

- Span< T > deviceSpan ()

  Explicit conversions to spans.
- Span< const T > **deviceSpan** () const
- Span< T > **deviceSpan** (std::size_t count)
- Span< const T > **deviceSpan** (std::size_t count) const
- template<std::size_t SIZE>
  Span< T, SIZE > **deviceSpan** ()
- template<std::size_t SIZE>
  Span< const T, SIZE > **deviceSpan** () const

Private Member Functions

- void setDevicePtr (std::uint8_t *devicePtr, bool useUnifiedMemory=false) noexcept

  Note that the device ptr will NOT be automatically deleted (deleter set to false)

Private Attributes

- DeviceUniquePtr< T > **devicePtr_** = nullptr
- std::size_t **memoryPoolDeviceIndex_** = 0
- bool **useUnifiedMemory_** = false

Friends

- class **CUDAMemoryPool**
- class **CUDAProcessMemoryPool**

5.31.1 Detailed Description

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

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

Using the Curiously Recurring Template Pattern (CRTP).

Author

David Lenz, Thanos Theo, 2019

Version

14.0.0.0

5.32 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

5.32.1 Detailed Description

The double2 class provides double2 functionality.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.33 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

5.33.1 Detailed Description

The double3 class provides double3 functionality.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.34 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

5.34.1 Detailed Description

The double4 class provides double4 functionality.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.35 UtilsCUDA::DynamicOrCompileTimeSize< SIZE > Class Template Reference

Helper class to differentiate whether something is fixed size at compile time or of dynamic size.

`#include <CUDAMemoryWrappers.h>`

Public Member Functions

- __forceinline__ __host__ __device__ **DynamicOrCompileTimeSize** (std::size_t size)
- void **resize** (std::size_t newSize)
- __forceinline__ __host__ __device__ std::size_t **size** () const

5.35.1 Detailed Description

template<std::size_t SIZE = DYNAMIC_SIZE>
class UtilsCUDA::DynamicOrCompileTimeSize< SIZE >

Helper class to differentiate whether something is fixed size at compile time or of dynamic size.

Used for zero-cost abstraction of Span. In case of a fixed size, the size does not need to be stored

Template Parameters

SIZE	indicates whether the size of the underlying array could be dynamic (not known at compile time) or static

Author

David Lenz, 2019

Version

14.0.0.0

5.36 UtilsCUDA::DynamicOrCompileTimeSize< DYNAMIC_SIZE > Class Template Reference

Template specialization for dynamically sized objects.

`#include <CUDAMemoryWrappers.h>`

Public Member Functions

- __forceinline__ __host__ __device__ **DynamicOrCompileTimeSize** (std::size_t size)
- __forceinline__ __host__ __device__ std::size_t **size** () const
- void resize (std::size_t newSize)

  Allows to change the stored size.

Private Attributes

- std::size_t **size_** = 0

5.36.1 Detailed Description

template<>
class UtilsCUDA::DynamicOrCompileTimeSize< DYNAMIC_SIZE >

Template specialization for dynamically sized objects.

It needs to store the size in a size_t.

Author

David Lenz, 2019

Version

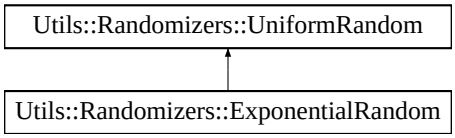
14.0.0.0

5.37 Utils::Randomizers::ExponentialRandom Class Reference

The ExponentialRandom class provides a exponential random number generator.

`#include <Randomizers.h>`

Inheritance diagram for 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

5.37.1 Detailed Description

The ExponentialRandom class provides a exponential random number generator.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.38 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

5.38.1 Detailed Description

The float2 class provides float2 functionality.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.39 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

5.39.1 Detailed Description

The float3 class provides float3 functionality.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.40 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

5.40.1 Detailed Description

The float4 class provides float4 functionality.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.41 Utils::FunctionView< Fn > Class Template Reference

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

`#include <FunctionView.h>`

5.41.1 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

5.42 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

5.43 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.

5.43.1 Detailed Description

This class contains only static CG & OpenGL related methods.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.43.2 Member Function Documentation

5.43.2.1 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.

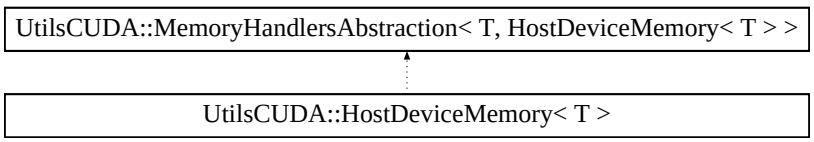
- The first parameter is a GLenum.
- The second parameter (optional) is a string that can be used to indicate the location where the error check occurs.
- The third parameter (optional) determines the destination of the error message. It defaults to cout, but could also be a file.

5.44 UtilsCUDA::HostDeviceMemory< T > Class Template Reference

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

```
#include <CUDAMemoryHandlers.h>
```

Inheritance diagram for UtilsCUDA::HostDeviceMemory< T >:



Public Member Functions

- void **allocate** (std::size_t numberOfElements, int device=0, unsigned int flags=cudaHostRegisterDefault) noexcept
- std::future< void > **allocateAsync** (std::size_t numberOfElements, int device=0, unsigned int flags=cudaHostRegisterDefault) noexcept
- void **reset** () noexcept
- std::future< void > **resetAsync** () noexcept
- void **swap** (HostDeviceMemory< T > &other) noexcept
- void **memset** (int value, bool memsetHost=true) noexcept
- void **memsetAsync** (int value, const cudaStream_t &stream, bool memsetHost=true) noexcept
- void **copyHostToDevice** () noexcept
- void **copyHostToDeviceAsync** (const cudaStream_t &stream) noexcept
- void **copyDeviceToHost** () const noexcept
- void copyDeviceToHost (const cudaStream_t &stream) const noexcept

  Note: asynchronous memcpy on the given stream, enforce further synchronization with device -> host copy for that stream.
- void **copyDeviceToHostAsync** (const cudaStream_t &stream) const noexcept
- T * **get** ()
- const T * **get** () const
- T * **host** ()
- const T * **host** () const
- T * **device** ()
- const T * **device** () const
- T & **operator[ ]** (std::size_t index)
- const T & **operator[ ]** (std::size_t index) const
- **operator bool** () const noexcept
- std::size_t **getNumberOfElements** () const
- bool **isMemoryPoolMode** () const
- **HostDeviceMemory** (std::size_t numberOfElements, int device=0, unsigned int flags=cudaHostRegisterDefault)
- **HostDeviceMemory** (const HostDeviceMemory &)=delete
- **HostDeviceMemory** (HostDeviceMemory &&)=delete
- HostDeviceMemory & **operator=** (const HostDeviceMemory &)=delete
- HostDeviceMemory & **operator=** (HostDeviceMemory &&)=delete

- operator RawDeviceMemory< T > ()

  Convenience functions to pass pointers to kernels.
- **operator RawDeviceMemory< const T >** () const

- Span< T > hostSpan ()

  Explicit conversions to spans (for host)
- Span< const T > **hostSpan** () const
- Span< T > **hostSpan** (std::size_t count)
- Span< const T > **hostSpan** (std::size_t count) const
- template<std::size_t SIZE>
  Span< T, SIZE > **hostSpan** ()
- template<std::size_t SIZE>
  Span< const T, SIZE > **hostSpan** () const
- Span< T > deviceSpan ()

  Explicit conversions to spans (for device)
- Span< const T > **deviceSpan** () const
- Span< T > **deviceSpan** (std::size_t count)
- Span< const T > **deviceSpan** (std::size_t count) const
- template<std::size_t SIZE>
  Span< T, SIZE > **deviceSpan** ()
- template<std::size_t SIZE>
  Span< const T, SIZE > **deviceSpan** () const

Private Member Functions

- void setHostPtr (std::uint8_t *hostPtr) noexcept

  Note that the host ptr will NOT be automatically deleted (deleter set to false)
- void setDevicePtr (std::uint8_t *devicePtr, bool=false) noexcept

  Note that the device ptr will NOT be automatically deleted (deleter set to false)

Private Attributes

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

Friends

- class **CUDAMemoryPool**
- class **CUDAProcessMemoryPool**

5.44.1 Detailed Description

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

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

Using the Curiously Recurring Template Pattern (CRTP).

Author

David Lenz, Thanos Theo, 2019

Version

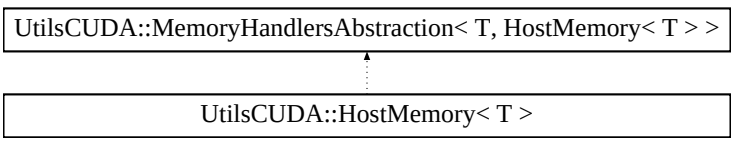
14.0.0.0

5.45 UtilsCUDA::HostMemory< T > Class Template Reference

This class encapsulates usage of a collection of host handling techniques (host side only) & the RAII C++ idiom.

`#include <CUDAMemoryHandlers.h>`

Inheritance diagram for UtilsCUDA::HostMemory< T >:



Public Member Functions

- void **allocate** (std::size_t numberOfElements, unsigned int flags=cudaHostRegisterDefault) noexcept
- std::future< void > **allocateAsync** (std::size_t numberOfElements, unsigned int flags=cudaHostRegister↩ Default) noexcept
- void **reset** () noexcept
- std::future< void > **resetAsync** () noexcept
- void **swap** (HostMemory< T > &other) noexcept
- void **memset** (int value) noexcept
- T * **get** ()
- const T * **get** () const
- T * **host** ()
- const T * **host** () const
- T & **operator[ ]** (std::size_t index)
- const T & **operator[ ]** (std::size_t index) const
- **operator bool** () const noexcept
- std::size_t **getNumberOfElements** () const
- bool **isMemoryPoolMode** () const
- Span< T > hostSpan ()

  Explicit conversions to spans (for host)
- Span< const T > **hostSpan** () const
- Span< T > **hostSpan** (std::size_t count)
- Span< const T > **hostSpan** (std::size_t count) const
- template<std::size_t SIZE>
  Span< T, SIZE > **hostSpan** ()
- template<std::size_t SIZE>
  Span< const T, SIZE > **hostSpan** () const
- **HostMemory** (std::size_t numberOfElements, unsigned int flags=cudaHostRegisterDefault)
- **HostMemory** (const HostMemory &)=delete
- **HostMemory** (HostMemory &&)=delete
- HostMemory & **operator=** (const HostMemory &)=delete
- HostMemory & **operator=** (HostMemory &&)=delete

Private Member Functions

- void setHostPtr (std::uint8_t *hostPtr) noexcept

  Note that the host ptr will NOT be automatically deleted (deleter set to false)

Private Attributes

- PinnedUniquePtr< T > **hostPtr_** = nullptr
- std::size_t **memoryPoolHostIndex_** = 0

Friends

- class **CUDAMemoryPool**
- class **CUDAProcessMemoryPool**

5.45.1 Detailed Description

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

This class encapsulates usage of a collection of host handling techniques (host side only) & the RAII C++ idiom.

Using the Curiously Recurring Template Pattern (CRTP).

Author

Thanos Theo, 2019

Version

14.0.0.0

5.46 Utils::UnitTests::IUnitTests< Derived > Class Template Reference

The IUnitTests struct encapsulate a basic unit test interface using the Curiously Recurring Template Pattern (CRTP).

`#include <UnitTests.h>`

Public Member Functions

- void **resetTests** ()
- bool **conductTests** ()
- void **reportTestResults** ()
- **IUnitTests** (const IUnitTests &)=delete
- **IUnitTests** (IUnitTests &&)=delete
- IUnitTests & **operator=** (const IUnitTests &)=delete
- IUnitTests & **operator=** (IUnitTests &&)=delete

Private Member Functions

- Derived * **asDerived** ()
- const Derived * **asDerived** () const

5.46.1 Detailed Description

template<typename Derived>
class Utils::UnitTests::IUnitTests< Derived >

The IUnitTests struct encapsulate a basic unit test interface using the Curiously Recurring Template Pattern (CRTP).

Author

Thanos Theo, 2018

Version

14.0.0.0

5.47 UtilsCUDA::KernelLauncher Class Reference

CUDA helper class to perform a more readable kernel launch in code with the fluent builder pattern.

`#include <CUDAKernelLauncher.h>`

Public Member Functions

- template<typename FunctionType , typename... Args>
  void runCUDAParallelFor (std::size_t arraySize, const FunctionType &lambdaFunction, Args &&... args)

  Launches the CUDAParallelFor kernel with its given lambda and arguments.
- template<typename FunctionType , typename... Args>
  void run (const FunctionType &kernelFunction, Args &&... args)

  Launches the generic CUDA kernel with its given arguments.
- KernelLauncher & setGrid (const dim3 &gridSize)

  Specifies the grid size of the problem (i.e.
- KernelLauncher & setBlock (const dim3 &blockSize)

  Specifies the block size of the problem (i.e.
- KernelLauncher & setGridAndBlock (const std::tuple< dim3, dim3 > &gridBlockSizes)

  Specifies the grid and block size of the problem.
- KernelLauncher & setSharedMemory (std::size_t sharedMemoryBytes)

  Preallocate shared memory of a specific size.
- KernelLauncher & setStream (const cudaStream_t &stream)

  Perform the kernel launch in a specific stream.
- KernelLauncher & synchronized ()

  Block kernel execution until the kernel finishes in the given thread.
- KernelLauncher & asynchronous ()

  Does not block kernel execute until the kernel finishes in the given thread.
- **KernelLauncher** (KernelLauncher &)=default
- **KernelLauncher** (KernelLauncher &&)=default
- KernelLauncher & **operator=** (KernelLauncher &)=default
- KernelLauncher & **operator=** (KernelLauncher &&)=default

Static Public Member Functions

- static KernelLauncher create ()

  Create a KernelLauncher.

Private Attributes

- dim3 **gridSize_** = 1
- dim3 **blockSize_** = 1
- std::size_t **sharedMemoryBytes_** = 0
- cudaStream_t **stream_** = nullptr
- bool **synchronize_** = false

5.47.1 Detailed Description

CUDA helper class to perform a more readable kernel launch in code with the fluent builder pattern.

The main purpose of this class to make kernel launches more readable and explicit without knowing the exact launch syntax of CUDA. It also improves readability in an IDE that does not properly parse CUDA <<<>>> syntax.

Example Usage: KernelLauncher::create().setGrid({1, 100}).setBlock({1}).setStream(stream).synchronized().run(kernel↩ Function, arg1, arg2, arg3); This is equivalent to: kernelFunction<<<{1, 100}, {1}, 0, stream>>>(arg1, arg2, arg3); CUDAError_checkCUDAErrorDebug(cudaPeekAtLastError()); CUDAError_checkCUDAError(cudaStream↩ Synchronize(stream));

Or:

KernelLauncher::create().setGrid({1, 100}).setBlock({1}).setStream(stream).synchronized().runCUDAParallel↩ For(arraySize, kernelLambda, arg1, arg2, arg3); This is equivalent to: kernelFunction<<<{1, 100}, {1}, 0, stream>>>(arraySize, kernelLambda, arg1, arg2, arg3); CUDAError_checkCUDAErrorDebug(cudaPeekAtLast↩ Error()); CUDAError_checkCUDAError(cudaStreamSynchronize(stream));

default values are: gridSize = {1} blockSize = {1} stream = 0 (default CUDA stream) sharedMemory = 0

Author

David Lenz, Amir Shahvarani, Thanos Theo, 2019

Version

14.0.0.0

5.47.2 Member Function Documentation

5.47.2.1 setBlock()

```
KernelLauncher& UtilsCUDA::KernelLauncher::setBlock (
            const dim3 & blockSize )  [inline]
```

Specifies the block size of the problem (i.e.

the number of threads).

5.47.2.2 setGrid()

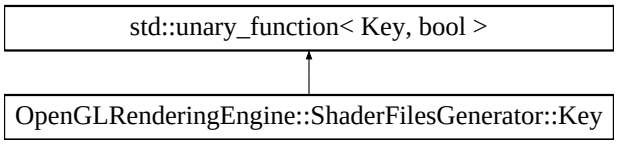
```
KernelLauncher& UtilsCUDA::KernelLauncher::setGrid (
            const dim3 & gridSize )  [inline]
```

Specifies the grid size of the problem (i.e.

the number of blocks).

5.48 OpenGLRenderingEngine::ShaderFilesGenerator::Key Class Reference

Inheritance diagram for 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

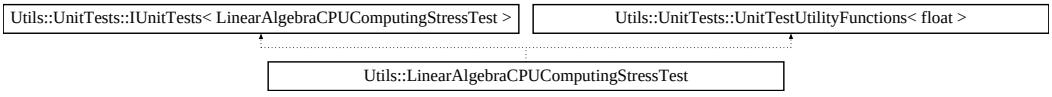
- std::string **first_** = ""
- BitsetType **second_** = 0

5.49 Utils::LinearAlgebraCPUComputingStressTest Class Reference

This class contains a basic Linear Algebra CPU Computing stress test case in the host.

`#include <LinearAlgebraCPUComputingStressTest.h>`

Inheritance diagram for Utils::LinearAlgebraCPUComputingStressTest:



Public Member Functions

- void **resetTests** ()
- bool **conductTests** ()
- void **reportTestResults** ()
- **LinearAlgebraCPUComputingStressTest** (std::size_t arraySize=16384, size_t numberOfCPUKernel↩ Iterations=160) noexcept
- **LinearAlgebraCPUComputingStressTest** (const LinearAlgebraCPUComputingStressTest &)=delete
- **LinearAlgebraCPUComputingStressTest** (LinearAlgebraCPUComputingStressTest &&)=delete
- LinearAlgebraCPUComputingStressTest & **operator=** (const LinearAlgebraCPUComputingStressTest &)=delete
- LinearAlgebraCPUComputingStressTest & **operator=** (LinearAlgebraCPUComputingStressTest &&)=delete

Private Attributes

- std::size_t **arraySize_** = 16384 ∗ 16384
- std::size_t **numberOfCPUKernelIterations_** = 0
- std::unique_ptr< std::int32_t[ ]> **arrayA_** = nullptr
- std::unique_ptr< std::int32_t[ ]> **arrayB_** = nullptr
- std::unique_ptr< std::int32_t[ ]> **arrayC_** = nullptr
- AccurateTimers::AccurateCPUTimer **cpuTimer_**
- double **totalTimeTakenInMs_** = 0.0

Additional Inherited Members

5.49.1 Detailed Description

This class contains a basic Linear Algebra CPU Computing stress test case in the host.

Using the Curiously Recurring Template Pattern (CRTP).

LinearAlgebraCPUComputingStressTest.h:

This class contains a basic Linear Algebra CPU Computing stress test case in the host. Using the Curiously Recurring Template Pattern (CRTP).

Author

Thanos Theo, 2019

Version

14.0.0.0

5.50 Utils::UtilityFunctions::MathFunctions Struct Reference

The MathFunctions struct 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)

  Strictly equal, using machine epsilon as tolerance.
- template<typename T >
  static bool scaledEpsilon (const T left, const T right, std::enable_if_t< std::is_arithmetic< T >::value > *=nullptr)

  Floating point error epsilon value scaled according to the range of the compared values.
- template<typename T >
  static bool relativelyEqual (const T left, const T right, std::enable_if_t< std::is_arithmetic< T >::value > *=nullptr)

  Relatively equal, using scaled machine epsilon as tolerance.
- template<typename T >
  static bool approximatelyEqual (const T left, const T right, const T tolerance, std::enable_if_t< std::is_↩ arithmetic< T >::value > *=nullptr)

  Approximately equal comparison for floating point numbers with explicitly specified tolerance.
- template<typename T >
  static bool marginallyLessThan (const T left, const T right, const T margin, std::enable_if_t< std::is_↩ arithmetic< T >::value > *=nullptr)

  Checks if left < right + margin.
- 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 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.

5.50.1 Detailed Description

The MathFunctions struct 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

5.50.2 Member Function Documentation

5.50.2.1 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

5.50.2.2 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

5.50.2.3 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

5.50.2.4 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

5.50.2.5 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

5.50.2.6 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

5.50.2.7 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

5.50.2.8 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

5.50.2.9 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

5.50.2.10 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

5.50.2.11 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

5.50.2.12 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

5.50.2.13 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

5.50.2.14 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

5.50.2.15 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

5.50.2.16 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

5.50.2.17 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

5.50.2.18 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

5.50.2.19 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

5.50.2.20 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

5.50.2.21 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

5.50.2.22 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

5.50.2.23 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.

5.51 UtilsCUDA::MemoryHandlersAbstraction< T, Derived > Class Template Reference

The MemoryHandlersAbstraction class encapsulates a basic abstraction layer for the memory handlers using the Curiously Recurring Template Pattern (CRTP).

`#include <CUDAMemoryHandlers.h>`

Public Member Functions

- void allocate (std::size_t numberOfElements, int device=0) noexcept

  Allocate functions (with future<void> variant)
- std::future< void > **allocateAsync** (std::size_t numberOfElements, int device=0) noexcept
- void reset () noexcept

  Reset functions (with future<void> variant)
- std::future< void > **resetAsync** () noexcept
- void swap (Derived &other) noexcept

  Swap function.
- void memset (int value) noexcept

  Memset function.
- T * get ()

  Convenience functions.
- const T * **get** () const
- T & **operator[ ]** (std::size_t index)
- const T & **operator[ ]** (std::size_t index) const
- **operator bool** () const noexcept
- std::size_t **getNumberOfElements** () const
- bool **isMemoryPoolMode** () const

Protected Member Functions

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

Protected Attributes

- std::size_t **numberOfElements_** = 0
- bool **memoryPoolMode_** = false

Private Member Functions

- Derived * **asDerived** ()
- const Derived * **asDerived** () const

5.51.1 Detailed Description

template<typename T, typename Derived>
class UtilsCUDA::MemoryHandlersAbstraction< T, Derived >

The MemoryHandlersAbstraction class encapsulates a basic abstraction layer for the memory handlers using the Curiously Recurring Template Pattern (CRTP).

Author

Thanos Theo, 2019

Version

14.0.0.0

5.52 UtilsCUDA::CUDAMemoryPool::MemoryPoolData Struct Reference

struct for Memory Pool Data

`#include <CUDAMemoryPool.h>`

Public Member Functions

- **MemoryPoolData** (std::uint8_t *ptr, std::size_t numberOfElements, std::size_t sizeOfElement, std::size_t offset, int device, const std::function< void(std::uint8_t *ptr, bool)> &memoryHandlerSetFunction) noexcept
- **MemoryPoolData** (const MemoryPoolData &)=delete
- **MemoryPoolData** (MemoryPoolData &&)=default
- MemoryPoolData & **operator=** (const MemoryPoolData &)=delete
- MemoryPoolData & **operator=** (MemoryPoolData &&)=delete

Public Attributes

- std::uint8_t * **ptr_** = nullptr
- std::size_t **numberOfElements_** = 0
- std::size_t **sizeOfElement_** = 0
- std::size_t **offset_** = 0
- int **device_** = 0
- const std::function< void(std::uint8_t *ptr, bool)> **memoryHandlerSetFunction** {nullptr}

5.52.1 Detailed Description

struct for Memory Pool Data

5.53 UtilsCUDA::CUDAProcessMemoryPool::MemoryPoolData Struct Reference

struct for Memory Pool Data

`#include <CUDAProcessMemoryPool.h>`

Public Member Functions

- **MemoryPoolData** (std::uint8_t *ptr, std::size_t numberOfElements, std::size_t sizeOfElement, std::size_t offset, int device) noexcept
- **MemoryPoolData** (const MemoryPoolData &)=delete
- **MemoryPoolData** (MemoryPoolData &&)=default
- MemoryPoolData & **operator=** (const MemoryPoolData &)=delete
- MemoryPoolData & **operator=** (MemoryPoolData &&)=delete

Public Attributes

- std::uint8_t * **ptr_** = nullptr
- std::size_t **numberOfElements_** = 0
- std::size_t **sizeOfElement_** = 0
- std::size_t **offset_** = 0
- int **device_** = 0

5.53.1 Detailed Description

struct for Memory Pool Data

5.54 UtilsCUDA::CUDAMemoryRegistry::MemoryRegistryData Struct Reference

struct for Memory Registry Data

`#include <CUDAMemoryRegistry.h>`

Public Member Functions

- **MemoryRegistryData** (std::uint8_t *ptr, std::size_t numberOfElements, std::size_t sizeOfElement) noexcept
- **MemoryRegistryData** (const MemoryRegistryData &)=default
- **MemoryRegistryData** (MemoryRegistryData &&)=default
- MemoryRegistryData & **operator=** (const MemoryRegistryData &)=default
- MemoryRegistryData & **operator=** (MemoryRegistryData &&)=default

Public Attributes

- std::uint8_t * **ptr_** = nullptr
- std::size_t **numberOfElements_** = 0
- std::size_t **sizeOfElement_** = 0

5.54.1 Detailed Description

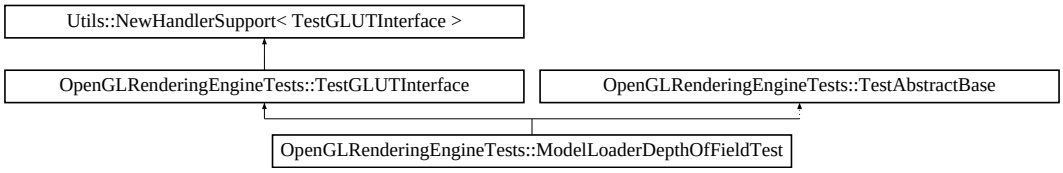
struct for Memory Registry Data

5.55 OpenGLRenderingEngineTests::ModelLoaderDepthOfFieldTest Class Reference

ModelLoaderDepthOfFieldTest is the 6th set of OpenGL rendering tests.

`#include <ModelLoaderDepthOfFieldTest.h>`

Inheritance diagram for OpenGLRenderingEngineTests::ModelLoaderDepthOfFieldTest:



Classes

- class OpenGLShaderFXAA_Antialias
- class OpenGLShaderGBufferEffects
- class OpenGLShaderGBufferEffectsBlurXY
- class OpenGLShaderGBufferEffectsHSSAO

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
- **ModelLoaderDepthOfFieldTest** (int screenWidth, int screenHeight, const std::string &textureFileName, const std::string &modelFileName, const std::string &modelLoaderDescriptorFileName, bool multisample) noexcept
- **ModelLoaderDepthOfFieldTest** (const ModelLoaderDepthOfFieldTest &)=delete
- **ModelLoaderDepthOfFieldTest** (ModelLoaderDepthOfFieldTest &&)=delete
- ModelLoaderDepthOfFieldTest & **operator=** (const ModelLoaderDepthOfFieldTest &)=delete
- ModelLoaderDepthOfFieldTest & **operator=** (ModelLoaderDepthOfFieldTest &&)=delete

Private Types

- enum **AllCachedRenderingTests** : std::size_t {
  USE_NO_CACHING_METHOD_1 = 0, **USE_NO_CACHING_METHOD_2** = 1, **USE_VERTEX_ARRAYS** = 2, **USE_DISPLAY_LISTS** = 3,
  USE_VBOS = 4 }
- enum **AllLightingModels** : std::size_t {
  FIXED_PIPELINE = 0, **PHONG** = 1, **BLINN_PHONG** = 2, **GAUSSIAN** = 3,
  TOON = 4, **GOOCH** = 5 }
- enum **GeometryShaderUsage** : std::size_t { **NO_USE** = 0, **PASS_THRU** = 1, **CREATE_NORMAL_GEO↩METRY** = 2 }
- enum **AllLODModels** : std::size_t { **PN_TRIANGLES** = 0, **PHONG_TESSELLATION** = 1 }
- enum **AllAdaptiveTesselationModes** : std::size_t {
  NO_ADAPTIVE_TESSELLATION = 0, **ADAPTIVE_TESSELLATION_TRIANGLE_POSITION_METRIC** = 1, **ADAPTIVE_TESSELLATION_LIGHT_METRIC** = 2, **ADAPTIVE_TESSELLATION_CONTOUR_METRIC** = 3,
  ADAPTIVE_TESSELLATION_ALL_METRICS = 4 }
- enum **GBUfferEffectTypes** : std::size_t {
  NO_GBUFFER_EFFECT = 0, **DEPTH_OF_FIELD** = 1, **EDGE_ENHANCEMENT** = 2, **HSSAO** = 3,
  HSSAO_PLUS = 4 }

Private Member Functions

- void **prepareFog** () const
- void **prepareLighting** ()
- void **updateLighting** ()
- void **prepareShaders** ()
- void **prepareLODShaders** ()
- void **prepareGBufferEffectsShaders** ()
- void **prepareFXAA_AntialiasShaders** ()
- void **prepareTexture** ()
- void **prepareScene** ()
- void **prepareGBufferEffectsFBOs** ()
- void **prepareHSSAOData** ()
- void **initGBufferEffectsFBOs** () const
- void **processGBufferEffectsFBOs** ()
- void **drawGBufferEffectsFBOs** ()
- void **prepareFXAA_AntialiasFBO** ()
- void **initFXAA_AntialiasFBO** () const
- void **drawFXAA_AntialiasFBO** ()
- void **prepareVBOs** ()
- void **deleteVBOs** ()
- void **renderNoCaching** (bool skipReInitModelDataForRendering)
- void **renderVertexArrays** ()
- void **renderDisplayLists** ()
- void **renderVBOs** ()
- void **clearScreen** (bool useGBuffer=false) const
- void **disableLights** (bool disableAllLightSources=true) const
- void **enableLights** (bool enableAllLightSources=true)
- void **renderModelScene** (bool useGBuffer=false)
- void **renderModelLoaderGeometry** (bool useGBuffer)
- bool **isUsingLOD** () const
- void **drawString** (const char ∗str, int x, int y, const float color[4], void ∗font) const
- void **drawString3D** (const char ∗str, float position[3], const float color[4], void ∗font) const
- void **showInfo** ()
- int **howMuchCurrentlyAvailableVRAMMemory** () const
- std::string **howMuchCurrentlyAvailableVRAMMemoryString** () const
- std::string **whichAdaptiveTessellationMetric** () const
- void **showFPS** ()

Private Attributes

- AllCachedRenderingTests **currentCachedRenderingTest** = USE_DISPLAY_LISTS
- AllLightingModels **currentLightingModel** = BLINN_PHONG
- GeometryShaderUsage **currentGeometryShader** = NO_USE
- AllLODModels **currentLODModel** = PN_TRIANGLES
- AllAdaptiveTesselationModes **currentAdaptiveTesselationMode** = ADAPTIVE_TESSELLATION_TRIANGLE_POSITION_METRIC
- GBUfferEffectTypes **currentGBUfferEffectType** = NO_GBUFFER_EFFECT
- OpenGLRenderingEngineTests::ModelLoaderDepthOfFieldTest::OpenGLShaderGBufferEffectsBlurXY * **openGLShaderGBufferEffectsBlurXY** = nullptr
- OpenGLRenderingEngineTests::ModelLoaderDepthOfFieldTest::OpenGLShaderGBufferEffectsHSSAO * **openGLShaderGBufferEffectsHSSAO** = nullptr
- OpenGLRenderingEngineTests::ModelLoaderDepthOfFieldTest::OpenGLShaderGBufferEffects * **openGLShaderGBufferEffects** = nullptr
- OpenGLRenderingEngineTests::ModelLoaderDepthOfFieldTest::OpenGLShaderFXAA_Antialias * **openGLShaderFXAA_Antialias** = nullptr
- OpenGLRenderingEngine::OpenGLModelAmbientLight * **openGLModelAmbientLight** = nullptr
- OpenGLRenderingEngine::OpenGLLight * **openGLLights** [NUMBER_OF_LIGHTS] = { nullptr }
- OpenGLRenderingEngine::OpenGLMaterial * **openGLMaterial** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingModels * **openGLShaderSurfaceLightingModels** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingModels * **openGLShaderSurfaceLightingModelsWithGeometry** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingModels * **openGLShaderSurfaceLightingModelsWithGeometryAndNormals** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingLODModels * **openGLShaderSurfaceLightingLODModels** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingLODModels * **openGLShaderSurfaceLightingLODModelsWithGeometry** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingLODModels * **openGLShaderSurfaceLightingLODModelsWithGeometryAndNormals** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectForGBufferPass1** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectRenderPass2** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectRenderPass3** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectForFXAA_Antialias** = nullptr
- OpenGLRenderingEngine::OpenGLILTexture * **openGLILTexture** = nullptr
- OpenGLRenderingEngine::OpenGLAssimpModelLoader * **openGLAssimpModelLoader** = nullptr
- bool **useLODModels** = false
- float **tessellationAlpha** = 1.0f
- float **tessellationLevel** = 32.0f
- bool **useAdaptiveTessellation** = true
- bool **useTrianglePositionAdaptiveTessellationMetric** = true
- bool **useLightAdaptiveTessellationMetric** = false
- bool **useContourAdaptiveTessellationMetric** = false
- int **depthOfFieldBlurXYNumberOfSamplesRange** = 8
- float **depthOfFieldRange** = 10.0f
- float **depthOfFieldZFocus** = 0.5f
- GLuint **rotationTextureID** = 0
- std::vector< float > **rotationTextureData**
- std::vector< float > **depthSamplesData**
- GLuint **allModelDataDisplayList**

- std::unordered_map< int, GLuint > **allModelStorageVBOColorsIDs**
- std::unordered_map< int, GLuint > **allModelStorageVBONormalsIDs**
- std::unordered_map< int, GLuint > **allModelStorageVBOVerticesIDs**
- std::unordered_map< int, GLuint > **allModelStorageVBOIndicesIDs**
- GLfloat **allLightsPositions** [NUMBER_OF_LIGHTS][3] = { { 0.0f } }
- int **useFog** = 0
- int **useTexturing** = 0
- int **useSphericalMapping** = 0
- int **usePositionalLights** = 0
- int **useOrenNayarDiffuseModel** = 0
- int **useFresnelFactorSchlickApproximationSpecularModel** = 0
- int **useColoredObjects** = 1
- int **useGouraudShading** = 1
- int **useBackFaceCulling** = 0
- int **previousAvailableMemory** = 0
- bool **reInitModelDataForRendering** = true
- int **printGLMemoryInfoOnSecondUpdate** = 2
- float **sinLightAngle** = 0.0f
- float **cosLightAngle** = 0.0f

Additional Inherited Members

5.55.1 Detailed Description

ModelLoaderDepthOfFieldTest is the 6th set of OpenGL rendering tests.

Author

Thanos Theo, 2009-2018

Version

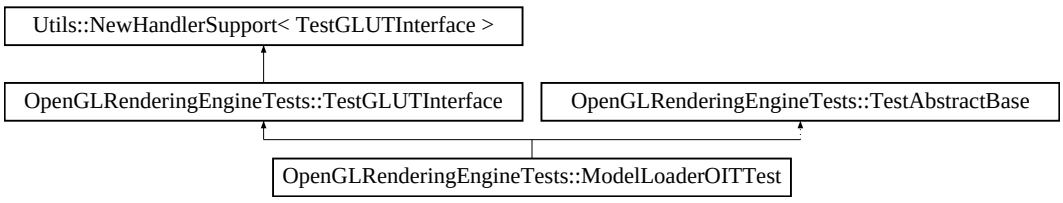
14.0.0.0

5.56 OpenGLRenderingEngineTests::ModelLoaderOITTest Class Reference

ModelLoaderOITTest is the 7th set of OpenGL rendering tests.

```
#include <ModelLoaderOITTest.h>
```

Inheritance diagram for OpenGLRenderingEngineTests::ModelLoaderOITTest:



Classes

- class OpenGLShaderClearABuffer3D
- class OpenGLShaderDisplayABuffer3D
- class OpenGLShaderFXAA_Antialias
- class OpenGLShaderGBufferEffects
- class OpenGLShaderGBufferEffectsBlurXY
- class OpenGLShaderGBufferEffectsHSSAO

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
- **ModelLoaderOITTest** (int screenWidth, int screenHeight, const std::string &textureFileName, const std::string &modelFileName, const std::string &modelLoaderDescriptorFileName, bool multisample) noexcept
- **ModelLoaderOITTest** (const ModelLoaderOITTest &)=delete
- **ModelLoaderOITTest** (ModelLoaderOITTest &&)=delete
- ModelLoaderOITTest & **operator=** (const ModelLoaderOITTest &)=delete
- ModelLoaderOITTest & **operator=** (ModelLoaderOITTest &&)=delete

Private Types

- enum **AllCachedRenderingTests** : size_t {
  USE_NO_CACHING_METHOD_1 = 0, **USE_NO_CACHING_METHOD_2** = 1, **USE_VERTEX_ARRAYS** = 2, **USE_DISPLAY_LISTS** = 3,
  USE_VBOS = 4 }
- enum **AllLightingModels** : size_t {
  FIXED_PIPELINE = 0, **PHONG** = 1, **BLINN_PHONG** = 2, **GAUSSIAN** = 3,
  TOON = 4, **GOOCH** = 5 }
- enum **GeometryShaderUsage** : size_t { **NO_USE** = 0, **PASS_THRU** = 1, **CREATE_NORMAL_GEOMETRY** = 2 }
- enum **AllLODModels** : size_t { **PN_TRIANGLES** = 0, **PHONG_TESSELLATION** = 1 }
- enum **AllAdaptiveTesselationModes** : std::size_t {
  NO_ADAPTIVE_TESSELLATION = 0, **ADAPTIVE_TESSELLATION_TRIANGLE_POSITION_METRIC** = 1, **ADAPTIVE_TESSELLATION_LIGHT_METRIC** = 2, **ADAPTIVE_TESSELLATION_CONTOUR_METRIC** = 3,
  ADAPTIVE_TESSELLATION_ALL_METRICS = 4 }
- enum **AllOITModes** : size_t {
  NO_OIT = 0, **A_BUFFER_3D_USE_TEXTURES** = 1, **A_BUFFER_3D_USE_BUFFERS** = 2, **A_BUFFER_3D_USE_LINKED_LIST_TEXTURES** = 3,
  A_BUFFER_3D_USE_LINKED_LIST_BUFFERS = 4 }
- enum **GBUfferEffectTypes** : size_t {
  NO_GBUFFER_EFFECT = 0, **DEPTH_OF_FIELD** = 1, **EDGE_ENHANCEMENT** = 2, **HSSAO** = 3,
  HSSAO_PLUS = 4 }

Private Member Functions

- void **prepareFog** () const
- void **prepareLighting** ()
- void **updateLighting** ()
- void **prepareShaders** ()
- void **prepareLODShaders** ()
- void **prepareABuffer3DShaders** ()
- void **prepareGBufferEffectsShaders** ()
- void **prepareFXAA_AntialiasShaders** ()
- void **prepareTexture** ()
- void **prepareScene** ()
- void **prepareGBufferEffectsFBOs** ()
- void **prepareHSSAOData** ()
- void **initGBufferEffectsFBOs** () const
- void **processGBufferEffectsFBOs** ()
- void **drawGBufferEffectsFBOs** ()
- void **prepareFXAA_AntialiasFBO** ()
- void **initFXAA_AntialiasFBO** () const
- void **drawFXAA_AntialiasFBO** ()
- void **initABuffer3D** ()
- bool **autoManageABuffer3D** ()
- void **deleteABuffer3D** ()
- void **writeABuffer3DBufferToFile** (GLbitfield barriers) const
- void **prepareVBOs** ()
- void **deleteVBOs** ()
- void **renderNoCaching** (bool skipReInitModelDataForRendering)
- void **renderVertexArrays** ()
- void **renderDisplayLists** ()
- void **renderVBOs** ()
- void **clearScreen** (bool useGBuffer=false) const
- void **disableLights** (bool disableAllLightSources=true) const
- void **enableLights** (bool enableAllLightSources=true)
- void **renderModelScene** (bool useGBuffer=false)
- void **renderModelLoaderGeometry** (bool useGBuffer)
- bool **isUsingLOD** () const
- void **drawString** (const char ∗str, int x, int y, const float color[4], void ∗font) const
- void **drawString3D** (const char ∗str, float position[3], const float color[4], void ∗font) const
- void **showInfo** ()
- int **howMuchCurrentlyAvailableVRAMMemory** () const
- std::string **howMuchCurrentlyAvailableVRAMMemoryString** () const
- std::string **whichAdaptiveTessellationMetric** () const
- void **showFPS** ()

Private Attributes

- AllCachedRenderingTests **currentCachedRenderingTest** = USE_DISPLAY_LISTS
- AllLightingModels **currentLightingModel** = BLINN_PHONG
- GeometryShaderUsage **currentGeometryShader** = NO_USE
- AllLODModels **currentLODModel** = PN_TRIANGLES
- AllAdaptiveTesselationModes **currentAdaptiveTesselationMode** = ADAPTIVE_TESSELLATION_TRIAN↩ GLE_POSITION_METRIC
- AllOITModes **currentOITMode** = A_BUFFER_3D_USE_BUFFERS
- GBUfferEffectTypes **currentGBUfferEffectType** = NO_GBUFFER_EFFECT

- OpenGLRenderingEngineTests::ModelLoaderOITTest::OpenGLShaderClearABuffer3D * **openGLShaderClearABuffer3D** = nullptr
- OpenGLRenderingEngineTests::ModelLoaderOITTest::OpenGLShaderDisplayABuffer3D * **openGLShaderDisplayABuffer3D** = nullptr
- OpenGLRenderingEngineTests::ModelLoaderOITTest::OpenGLShaderGBufferEffectsBlurXY * **openGLShaderGBufferEffectsBlurXY** = nullptr
- OpenGLRenderingEngineTests::ModelLoaderOITTest::OpenGLShaderGBufferEffectsHSSAO * **openGLShaderGBufferEffectsHSSAO** = nullptr
- OpenGLRenderingEngineTests::ModelLoaderOITTest::OpenGLShaderGBufferEffects * **openGLShaderGBufferEffects** = nullptr
- OpenGLRenderingEngineTests::ModelLoaderOITTest::OpenGLShaderFXAA_Antialias * **openGLShaderFXAA_Antialias** = nullptr
- OpenGLRenderingEngine::OpenGLModelAmbientLight * **openGLModelAmbientLight** = nullptr
- OpenGLRenderingEngine::OpenGLLight * **openGLLights** [NUMBER_OF_LIGHTS] = { nullptr }
- OpenGLRenderingEngine::OpenGLMaterial * **openGLMaterial** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingModels * **openGLShaderSurfaceLightingModels** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingModels * **openGLShaderSurfaceLightingModelsWithGeometry** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingModels * **openGLShaderSurfaceLightingModelsWithGeometryAndNormals** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingLODModels * **openGLShaderSurfaceLightingLODModels** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingLODModels * **openGLShaderSurfaceLightingLODModelsWithGeometry** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingLODModels * **openGLShaderSurfaceLightingLODModelsWithGeometryAndNormals** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectForGBufferPass1** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectRenderPass2** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectRenderPass3** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectForFXAA_Antialias** = nullptr
- OpenGLRenderingEngine::OpenGLILTexture * **openGLILTexture** = nullptr
- OpenGLRenderingEngine::OpenGLAssimpModelLoader * **openGLAssimpModelLoader** = nullptr
- bool **useLODModels** = false
- float **tessellationAlpha** = 1.0f
- float **tessellationLevel** = 32.0f
- bool **useAdaptiveTessellation** = true
- bool **useTrianglePositionAdaptiveTessellationMetric** = true
- bool **useLightAdaptiveTessellationMetric** = false
- bool **useContourAdaptiveTessellationMetric** = false
- int **depthOfFieldBlurXYNumberOfSamplesRange** = 8
- float **depthOfFieldRange** = 10.0f
- float **depthOfFieldZFocus** = 0.5f
- GLuint **rotationTextureID** = 0
- std::vector< float > **rotationTextureData**
- std::vector< float > **depthSamplesData**
- GLuint **allModelDataDisplayList** = 0
- std::unordered_map< int, GLuint > **allModelStorageVBOColorsIDs**
- std::unordered_map< int, GLuint > **allModelStorageVBONormalsIDs**
- std::unordered_map< int, GLuint > **allModelStorageVBOVerticesIDs**
- std::unordered_map< int, GLuint > **allModelStorageVBOIndicesIDs**
- GLsizei **aBuffer3DSize** = 16

- GLuint **aBuffer3DCounterTextureID** = 0
- GLuint **aBuffer3DTextureID** = 0
- GLuint **aBuffer3DCounterBufferID** = 0
- GLuint **aBuffer3DBufferID** = 0
- GLuint64EXT **aBuffer3DLinkedListSize** = 1 << 21
- GLsizei **aBuffer3DinkedListMaxSize** = 16
- GLuint **aBuffer3DLinkedListAtomicCounterBufferID** = 0
- GLuint **aBuffer3DLinkedListOffsetTextureID** = 0
- GLuint **aBuffer3DLinkedListTextureID** = 0
- GLuint **aBuffer3DLinkedListOffsetBufferID** = 0
- GLuint **aBuffer3DLinkedListBufferID** = 0
- bool **oitAutoManageABuffer3D** = 1
- bool **oitSortFragments** = true
- bool **oitResolveTranslucency** = true
- float **oitTranslucencyAbsorptionCoefficient** = 10.0f
- float **opacity** = 0.0f
- bool **oitUseDeferredShading** = false
- bool **oitCaptureABuffer3DBufferToFile** = false
- bool **reInitOITShaders** = false
- GLfloat **allLightsPositions** [NUMBER_OF_LIGHTS][3] = { { 0.0f } }
- int **useFog** = 0
- int **useTexturing** = 0
- int **useSphericalMapping** = 0
- int **usePositionalLights** = 0
- int **useOrenNayarDiffuseModel** = 0
- int **useFresnelFactorSchlickApproximationSpecularModel** = 0
- int **useColoredObjects** = 1
- int **useGouraudShading** = 1
- int **useBackFaceCulling** = 0
- int **previousAvailableMemory** = 0
- bool **reInitModelDataForRendering** = true
- int **printGLMemoryInfoOnSecondUpdate** = 2
- bool **isNavigating** = false
- float **sinLightAngle** = 0.0f
- float **cosLightAngle** = 0.0f
- GLuint **dummyVao** = 0

Additional Inherited Members

5.56.1 Detailed Description

ModelLoaderOITTest is the 7th set of OpenGL rendering tests.

Author

Thanos Theo, 2009-2018

Version

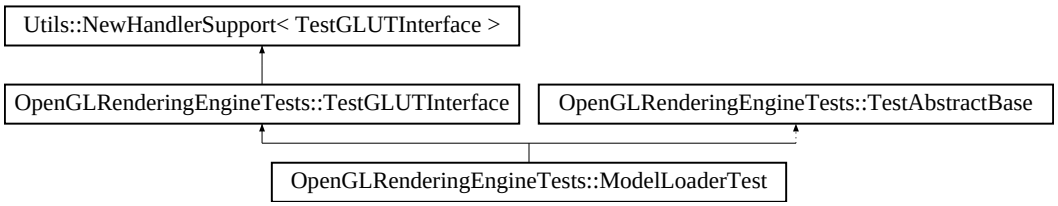
14.0.0.0

5.57 OpenGLRenderingEngineTests::ModelLoaderTest Class Reference

ModelLoaderTest is the 3rd set of OpenGL rendering tests.

`#include <ModelLoaderTest.h>`

Inheritance diagram for OpenGLRenderingEngineTests::ModelLoaderTest:



Classes

- class OpenGLShaderFXAA_Antialias
- class OpenGLShaderGBufferEffects
- class OpenGLShaderGBufferEffectsBlurXY
- class OpenGLShaderGBufferEffectsHSSAO

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
- **ModelLoaderTest** (int screenWidth, int screenHeight, const std::string &textureFileName, const std::string &modelFileName, const std::string &modelLoaderDescriptorFileName, bool multisample) noexcept
- **ModelLoaderTest** (const ModelLoaderTest &)=delete
- **ModelLoaderTest** (ModelLoaderTest &&)=delete
- ModelLoaderTest & **operator=** (const ModelLoaderTest &)=delete
- ModelLoaderTest & **operator=** (ModelLoaderTest &&)=delete

Private Types

- enum **AllCachedRenderingTests** : std::size_t {
  USE_NO_CACHING_METHOD_1 = 0, **USE_NO_CACHING_METHOD_2** = 1, **USE_VERTEX_ARRAYS** = 2, **USE_DISPLAY_LISTS** = 3,
  USE_VBOS = 4 }
- enum **AllLightingModels** : std::size_t {
  FIXED_PIPELINE = 0, **PHONG** = 1, **BLINN_PHONG** = 2, **GAUSSIAN** = 3,
  TOON = 4, **GOOCH** = 5 }
- enum **GeometryShaderUsage** : std::size_t { **NO_USE** = 0, **PASS_THRU** = 1, **CREATE_NORMAL_GEO↩METRY** = 2 }
- enum **AllLODModels** : std::size_t { **PN_TRIANGLES** = 0, **PHONG_TESSELLATION** = 1 }
- enum **AllAdaptiveTesselationModes** : std::size_t {
  NO_ADAPTIVE_TESSELLATION = 0, **ADAPTIVE_TESSELLATION_TRIANGLE_POSITION_METRIC** = 1, **ADAPTIVE_TESSELLATION_LIGHT_METRIC** = 2, **ADAPTIVE_TESSELLATION_CONTOUR_METRIC** = 3,
  ADAPTIVE_TESSELLATION_ALL_METRICS = 4 }
- enum **GBUfferEffectTypes** : std::size_t {
  NO_GBUFFER_EFFECT = 0, **DEPTH_OF_FIELD** = 1, **EDGE_ENHANCEMENT** = 2, **HSSAO** = 3,
  HSSAO_PLUS = 4 }

Private Member Functions

- void **prepareFog** () const
- void **prepareLighting** ()
- void **updateLighting** ()
- void **prepareShaders** ()
- void **prepareLODShaders** ()
- void **prepareGBufferEffectsShaders** ()
- void **prepareFXAA_AntialiasShaders** ()
- void **prepareTexture** ()
- void **prepareScene** ()
- void **prepareEnvironmentMappingFBO** ()
- void **initEnviromentMappingFBO** () const
- void **drawEnvironmentMapping** ()
- void **prepareDepthStencilFBO** ()
- void **initDepthStencilFBO** () const
- void **prepareGBufferEffectsFBOs** ()
- void **prepareHSSAOData** ()
- void **initGBufferEffectsFBOs** () const
- void **processGBufferEffectsFBOs** ()
- void **drawGBufferEffectsFBOs** ()
- void **prepareFXAA_AntialiasFBO** ()
- void **initFXAA_AntialiasFBO** () const
- void **drawFXAA_AntialiasFBO** ()
- void **prepareVBOs** ()
- void **deleteVBOs** ()
- void **renderNoCaching** (bool skipReInitModelDataForRendering)
- void **renderVertexArrays** ()
- void **renderDisplayLists** ()
- void **renderVBOs** ()
- void **clearScreen** (bool useGBuffer=false) const
- void **disableLights** (bool disableAllLightSources=true) const
- void **enableLights** (bool enableAllLightSources=true)
- void **renderModelScene** (bool environmentMappingRenderPass, bool useGBuffer=false)
- void **renderModelLoaderGeometry** (bool environmentMappingRenderPass, bool useGBuffer, bool enable↩ StencilBufferColor)
- bool **isUsingLOD** () const
- void **drawString** (const char ∗str, int x, int y, const float color[4], void ∗font) const
- void **drawString3D** (const char ∗str, float position[3], const float color[4], void ∗font) const
- void **showInfo** ()
- int **howMuchCurrentlyAvailableVRAMMemory** () const
- std::string **howMuchCurrentlyAvailableVRAMMemoryString** () const
- std::string **whichAdaptiveTessellationMetric** () const
- void **showFPS** ()

Private Attributes

- AllCachedRenderingTests **currentCachedRenderingTest** = USE_DISPLAY_LISTS
- AllLightingModels **currentLightingModel** = BLINN_PHONG
- GeometryShaderUsage **currentGeometryShader** = NO_USE
- AllLODModels **currentLODModel** = PN_TRIANGLES
- AllAdaptiveTesselationModes **currentAdaptiveTesselationMode** = ADAPTIVE_TESSELLATION_TRIAN↩ GLE_POSITION_METRIC
- GBUfferEffectTypes **currentGBUfferEffectType** = NO_GBUFFER_EFFECT

- OpenGLRenderingEngineTests::ModelLoaderTest::OpenGLShaderGBufferEffectsBlurXY * **openGLShaderGBufferEffectsBlurXY** = nullptr
- OpenGLRenderingEngineTests::ModelLoaderTest::OpenGLShaderGBufferEffectsHSSAO * **openGLShaderGBufferEffectsHSSAO** = nullptr
- OpenGLRenderingEngineTests::ModelLoaderTest::OpenGLShaderGBufferEffects * **openGLShaderGBufferEffects** = nullptr
- OpenGLRenderingEngineTests::ModelLoaderTest::OpenGLShaderFXAA_Antialias * **openGLShaderFXAA_Antialias** = nullptr
- OpenGLRenderingEngine::OpenGLModelAmbientLight * **openGLModelAmbientLight** = nullptr
- OpenGLRenderingEngine::OpenGLLight * **openGLLights** [NUMBER_OF_LIGHTS] = { nullptr }
- OpenGLRenderingEngine::OpenGLMaterial * **openGLMaterial** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingModels * **openGLShaderSurfaceLightingModels** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingModels * **openGLShaderSurfaceLightingModelsWithGeometry** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingModels * **openGLShaderSurfaceLightingModelsWithGeometryAndNormals** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingLODModels * **openGLShaderSurfaceLightingLODModels** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingLODModels * **openGLShaderSurfaceLightingLODModelsWithGeometry** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingLODModels * **openGLShaderSurfaceLightingLODModelsWithGeometryAndNormals** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectForEnvironmentMapping** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectForHighQualityOutline** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectForGBufferPass1** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectRenderPass2** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectRenderPass3** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectForFXAA_Antialias** = nullptr
- OpenGLRenderingEngine::OpenGLILTexture * **openGLILTexture** = nullptr
- OpenGLRenderingEngine::OpenGLAssimpModelLoader * **openGLAssimpModelLoader** = nullptr
- bool **useLODModels** = false
- float **tessellationAlpha** = 1.0f
- float **tessellationLevel** = 32.0f
- bool **useAdaptiveTessellation** = true
- bool **useTrianglePositionAdaptiveTessellationMetric** = true
- bool **useLightAdaptiveTessellationMetric** = false
- bool **useContourAdaptiveTessellationMetric** = false
- int **depthOfFieldBlurXYNumberOfSamplesRange** = 8
- float **depthOfFieldRange** = 10.0f
- float **depthOfFieldZFocus** = 0.5f
- GLuint **rotationTextureID** = 0
- std::vector< float > **rotationTextureData**
- std::vector< float > **depthSamplesData**
- GLuint **allModelDataDisplayList** = 0
- std::unordered_map< int, GLuint > **allModelStorageVBOColorsIDs**
- std::unordered_map< int, GLuint > **allModelStorageVBONormalsIDs**
- std::unordered_map< int, GLuint > **allModelStorageVBOVerticesIDs**
- std::unordered_map< int, GLuint > **allModelStorageVBOIndicesIDs**
- GLfloat **allLightsPositions** [NUMBER_OF_LIGHTS][3] = { { 0.0f } }

- int **useFog** = 0
- int **useTexturing** = 0
- int **useSphericalMapping** = 0
- int **useEnvironmentMapping** = 0
- int **useOutline** = 0
- int **useHighQualityOutline** = 0
- int **usePositionalLights** = 0
- int **useOrenNayarDiffuseModel** = 0
- int **useFresnelFactorSchlickApproximationSpecularModel** = 0
- int **useColoredObjects** = 1
- int **useGouraudShading** = 1
- int **useBackFaceCulling** = 0
- int **previousAvailableMemory** = 0
- bool **reInitModelDataForRendering** = true
- int **printGLMemoryInfoOnSecondUpdate** = 2
- float **sinLightAngle** = 0.0f
- float **cosLightAngle** = 0.0f

Additional Inherited Members

5.57.1 Detailed Description

ModelLoaderTest is the 3rd set of OpenGL rendering tests.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.58 SuperQuadrics::ModelSettings Struct Reference

ModelSettings class which holds information for the settings of a SuperQuadrics model.

`#include <ModelSettings.h>`

Public Member Functions

- **ModelSettings** (bool usingNormals, bool usingTexCoords) noexcept
- **ModelSettings** (bool usingCCW, bool usingNormals, bool usingTexCoords, bool usingManipulateGeometry, double shrinkGeometryFactor, bool usingGeometryExplosionFactor) noexcept
- **ModelSettings** (const ModelSettings &)=default
- **ModelSettings** (ModelSettings &&)=default
- ModelSettings & **operator=** (const ModelSettings &)=default
- ModelSettings & **operator=** (ModelSettings &&)=default

Public Attributes

- bool usingCCW = true

  ModelSettings usingCCW variable.
- bool usingNormals = false

  ModelSettings usingNormals variable.
- bool usingTexCoords = false

  ModelSettings usingTexCoords variable.
- bool usingManipulateGeometry = false

  ModelSettings usingManipulateGeometry variable.
- double shrinkGeometryFactor = 0.0

  ModelSettings shrinkGeometryFactor variable.
- bool usingGeometryExplosionFactor = false

  ModelSettings usingGeometryExplosionFactor variable.

5.58.1 Detailed Description

ModelSettings class which holds information for the settings of a SuperQuadrics model.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.59 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_**

5.60 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

5.60.1 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

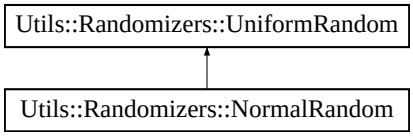
14.0.0.0

5.61 Utils::Randomizers::NormalRandom Class Reference

The NormalRandom class provides a normal random number generator.

`#include <Randomizers.h>`

Inheritance diagram for 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

5.61.1 Detailed Description

The NormalRandom class provides a normal random number generator.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.62 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_**

5.62.1 Detailed Description

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

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.63 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_**

5.63.1 Detailed Description

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

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.64 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 constexpr size_t **NUMBER_OF_TOTAL_SHADER_TYPES** = 6
- static constexpr CharacterEncodingMethods **CHARACTER_ENCODING_METHOD** = BASE64_FLIP_XO↩
  R_SWAP_BITS_COMPRESSION

5.64.1 Detailed Description

This class encapsulates usage of an OpenGL Asset Manager.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.65 OpenGLRenderingEngine::OpenGLAssimpModelLoader Class Reference

This class encapsulates usage of an OpenGL Model Loader using the Assimp library.

`#include <OpenGLAssimpModelLoader.h>`

Public Member Functions

- int **getTotalNumberOfFaces** () const
- int **getTotalNumberOfVertices** () const
- float **getSceneScaleFactor** () const
- float **getSceneCenterX** () const
- float **getSceneCenterY** () const
- float **getSceneCenterZ** () const
- std::unordered_map< int, std::vector< GLfloat > > & **getAllModelStorageColorsData** ()
- std::unordered_map< int, std::vector< GLfloat > > & **getAllModelStorageNormalsData** ()
- std::unordered_map< int, std::vector< GLfloat > > & **getAllModelStorageVerticesData** ()
- std::unordered_map< int, std::vector< GLuint > > & **getAllModelStorageIndicesData** ()
- void **load** (const std::string &modelDirectory, const std::string &modelFileName, const std::string &model↩ LoaderDescriptorFileName, bool triangulate=true, bool generateSmoothNormals=true, bool joinIdentical↩ Vertices=true)
- void **recursiveRenderRoot** () const
- **OpenGLAssimpModelLoader** (const OpenGLAssimpModelLoader &)=delete
- **OpenGLAssimpModelLoader** (OpenGLAssimpModelLoader &&)=delete
- OpenGLAssimpModelLoader & **operator=** (const OpenGLAssimpModelLoader &)=delete
- OpenGLAssimpModelLoader & **operator=** (OpenGLAssimpModelLoader &&)=delete

Private Member Functions

- void **resetSceneDataStructures** ()
- void **getBoundingBoxForNode** (const aiNode ∗node, aiMatrix4x4 ∗trafo, aiVector3D &minimum, aiVector3D &maximum) const
- void **getBoundingBox** (aiVector3D &minimum, aiVector3D &maximum) const
- void **recursiveStoreModelData** (const aiScene ∗scene, const aiNode ∗node)
- void **storeColorDataInMap** (int numberOfIndices, const aiColor4D ∗color)
- void **storeNormalDataInMap** (int numberOfIndices, const aiVector3D ∗normal)
- void **storeVertexDataInMap** (int numberOfIndices, const aiVector3D ∗vertex)
- void **recursiveStoreModelDataWithTriangulationAndIndexing** (const aiScene ∗scene, const aiNode ∗node)
- void **storeIndexDataInMap** (int numberOfIndices, const GLuint ∗index)
- float **getSceneScaleFactor** (aiVector3D &minimum, aiVector3D &maximum) const
- void **reportModelPrimitiveUsage** ()
- void **loadModel** (const std::string &modelDirectory, const std::string &modelFileName, bool triangulate, bool generateSmoothNormals, bool joinIdenticalVertices)
- bool **loadAsset** (const std::string &modelDirectory, const std::string &modelFileName, bool triangulate, bool generateSmoothNormals, bool joinIdenticalVertices)
- bool **writeToBinaryFilesForModelDataWithTriangulationAndIndexing** (const std::string &modelDirectory, const std::string &modelFileName)
- bool **loadFromBinaryFilesForModelDataWithTriangulationAndIndexing** (const std::string &model↩ Directory, const std::string &modelFileName)
- void **getBoundingBoxForModelDataWithTriangulationAndIndexing** ()

Private Attributes

- int **totalNumberOfFaces_** = 0
- int **totalNumberOfVertices_** = 0
- float **sceneScaleFactor_** = 1.0f
- std::unordered_map< int, std::vector< GLfloat > > **allModelStorageColorsData_**
- std::unordered_map< int, std::vector< GLfloat > > **allModelStorageNormalsData_**
- std::unordered_map< int, std::vector< GLfloat > > **allModelStorageVerticesData_**
- std::unordered_map< int, std::vector< GLuint > > **allModelStorageIndicesData_**
- const aiScene * **scene_** = nullptr
- aiVector3D **sceneMin_** {}
- aiVector3D **sceneMax_** {}
- aiVector3D **sceneCenter_** {}

5.65.1 Detailed Description

This class encapsulates usage of an OpenGL Model Loader using the Assimp library.

Author

Thanos Theo, 2009-2018

Version

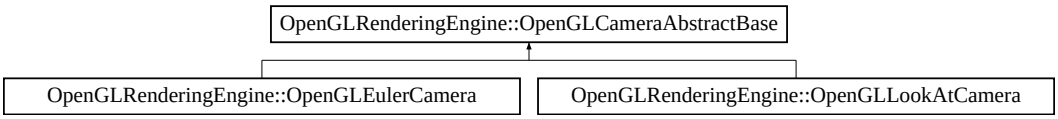
14.0.0.0

5.66 OpenGLRenderingEngine::OpenGLCameraAbstractBase Class Reference

This abstract class encapsulates usage of an OpenGL camera.

`#include <OpenGLCameraAbstractBase.h>`

Inheritance diagram for OpenGLRenderingEngine::OpenGLCameraAbstractBase:



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()

5.66.1 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

5.67 OpenGLRenderingEngine::OpenGLDriverInfo Class Reference

Gets GL vendor, version, supported extensions and other states using glGet* functions and store them in OpenGLDriverInfo 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 }

5.67.1 Detailed Description

Gets GL vendor, version, supported extensions and other states using glGet∗ functions and store them in OpenGLDriverInfo 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

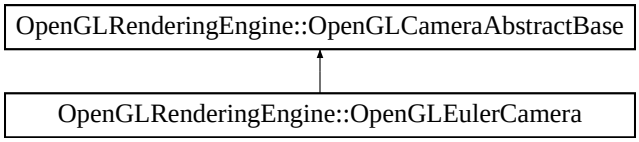
14.0.0.0

5.68 OpenGLRenderingEngine::OpenGLEulerCamera Class Reference

This class encapsulates usage of an OpenGL Euler camera.

`#include <OpenGLEulerCamera.h>`

Inheritance diagram for 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

5.68.1 Detailed Description

This class encapsulates usage of an OpenGL Euler camera.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.69 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 useDummyVAO=true) const
- void **disposeFrameBufferObjectResources** ()
- **OpenGLFrameBufferObject** (OpenGLDriverInfo *openGLDriverInfo, bool useTexture=true, bool depthRenderBuffer=false, bool useDepthTexture=false, bool depthStencilRenderBuffer=false, GLint textureFormat1=GL_RGBA8, GLenum textureFormat2=GL_RGBA, GLenum textureFormatType=GL_UNSIGNED_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

5.69.1 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

5.69.2 Member Function Documentation

5.69.2.1 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).

5.70 OpenGLRenderingEngine::OpenGLILTexture Class Reference

This class encapsulates usage of an OpenGL texture using the DevIL library for image handling.

```
#include <OpenGLILTexture.h>
```

Public Member Functions

- GLuint **getILTextureID** () const
- void **createILTexture** (const std::string &textureDirectory, const std::string &textureFileName, bool enforceImageFlip=true, GLenum textureUnit=0)
- void **deleteILTexture** ()
- void **enable** () const
- void **disable** () const
- void **bindTexture** (GLenum textureUnit=0) const
- void **unbind** (GLenum textureUnit) const
- **OpenGLILTexture** (bool generateMipmap=false, bool supportsMaxTextureMaxAnisotropy=false, GLint maxTextureMaxAnisotropy=0) noexcept
- **OpenGLILTexture** (const OpenGLILTexture &)=delete
- **OpenGLILTexture** (OpenGLILTexture &&)=delete
- OpenGLILTexture & **operator=** (const OpenGLILTexture &)=delete
- OpenGLILTexture & **operator=** (OpenGLILTexture &&)=delete

Static Public Member Functions

- static void **saveILImage** (const std::string &imageDirectory, const std::string &imageFileNamePrefix, const std::string &imageFileNameExtension, ILubyte imageFormat, ILuint imageWidth, ILuint imageHeight, ILubyte *imagePixels)

Private Attributes

- GLuint **ILtextureID_** = 0
- bool **generateMipmap_** = false
- bool **supportsMaxTextureMaxAnisotropy_** = false
- GLint **maxTextureMaxAnisotropy_** = 0

5.70.1 Detailed Description

This class encapsulates usage of an OpenGL texture using the DevIL library for image handling.

Author

Thanos Theo, 2009-2018

Version

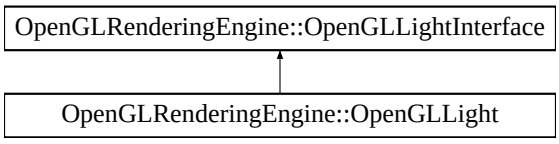
14.0.0.0

5.71 OpenGLRenderingEngine::OpenGLLight Class Reference

This class encapsulates usage of an OpenGL light (directional or positional).

```
#include <OpenGLLight.h>
```

Inheritance diagram for OpenGLRenderingEngine::OpenGLLight:



Public Member Functions

- void **setAmbientColor** (float r, float g, float b, float a)
- void **setAmbientColor** (const float ambientColor[4])
- void **setDiffuseColor** (float r, float g, float b, float a)
- void **setDiffuseColor** (const float diffuseColor[4])
- void **setSpecularColor** (float r, float g, float b, float a)
- void **setSpecularColor** (const float specularColor[4])
- void **setDirection** (float x, float y, float z)
- void **setDirection** (const float direction[3])
- void **setPosition** (float x, float y, float z)
- void **setPosition** (const float position[3])
- void **setConstantAttenuation** (float constantAttenuation)
- void **setLinearAttenuation** (float linearAttenuation)
- void **setQuadraticAttenuation** (float quadraticAttenuation)
- void **enable** (int lightId=GL_LIGHT0, bool usePositionalLight=false) override
- void **update** (int lightId=GL_LIGHT0, bool usePositionalLight=false, float updateX=0.0f, float updateY=0.0f, float updateZ=0.0f) override
- void **disable** () const override
- **OpenGLLight** (const OpenGLLight &)=delete
- **OpenGLLight** (OpenGLLight &&)=delete
- OpenGLLight & **operator=** (const OpenGLLight &)=delete
- OpenGLLight & **operator=** (OpenGLLight &&)=delete

Private Attributes

- float **ambientColor_** [4] = { 0.0f }
- float **diffuseColor_** [4] = { 0.0f }
- float **specularColor_** [4] = { 0.0f }
- float **direction_** [4] = { 0.0f }
- float **position_** [4] = { 0.0f }
- float **constantAttenuation_** = 1.0f
- float **linearAttenuation_** = 0.5f
- float **quadraticAttenuation_** = 0.25f
- int **lastLightId_** = GL_LIGHT0

5.71.1 Detailed Description

This class encapsulates usage of an OpenGL light (directional or positional).

Author

Thanos Theo, 2009-2018

Version

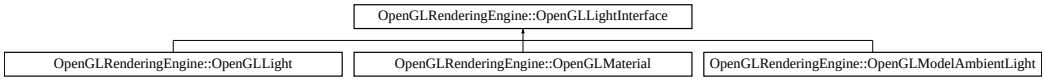
14.0.0.0

5.72 OpenGLRenderingEngine::OpenGLLightInterface Struct Reference

This abstract class encapsulates usage of an OpenGL light.

`#include <OpenGLLightInterface.h>`

Inheritance diagram for OpenGLRenderingEngine::OpenGLLightInterface:



Public Member Functions

- virtual void **enable** (int lightId=GL_LIGHT0, bool usePositionalLight=false)=0
- virtual void **update** (int lightId=GL_LIGHT0, bool usePositionalLight=false, float updateX=0.0f, float update↩ Y=0.0f, float updateZ=0.0f)=0
- virtual void **disable** () const =0
- **OpenGLLightInterface** (const OpenGLLightInterface &)=delete
- **OpenGLLightInterface** (OpenGLLightInterface &&)=delete
- OpenGLLightInterface & **operator=** (const OpenGLLightInterface &)=delete
- OpenGLLightInterface & **operator=** (OpenGLLightInterface &&)=delete

5.72.1 Detailed Description

This abstract class encapsulates usage of an OpenGL light.

To be inherited from usage-specific sub-classes.

Author

Thanos Theo, 2009-2018

Version

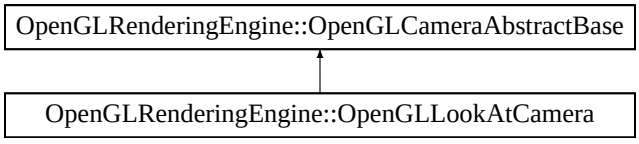
14.0.0.0

5.73 OpenGLRenderingEngine::OpenGLLookAtCamera Class Reference

This class encapsulates usage of an OpenGL LookAt camera.

`#include <OpenGLLookAtCamera.h>`

Inheritance diagram for OpenGLRenderingEngine::OpenGLLookAtCamera:



Public Member Functions

- void **setMatrices** () const override
- void **setLookAtCamera** (double positionX, double positionY, double positionZ, double lookAtX, double lookAtY, double lookAtZ, double upX, double upY, double upZ)
- **OpenGLLookAtCamera** (float fieldOfView) noexcept
- **OpenGLLookAtCamera** (const OpenGLLookAtCamera &)=delete
- **OpenGLLookAtCamera** (OpenGLLookAtCamera &&)=delete
- OpenGLLookAtCamera & **operator=** (const OpenGLLookAtCamera &)=delete
- OpenGLLookAtCamera & **operator=** (OpenGLLookAtCamera &&)=delete

Additional Inherited Members

5.73.1 Detailed Description

This class encapsulates usage of an OpenGL LookAt camera.

Author

Thanos Theo, 2009-2018

Version

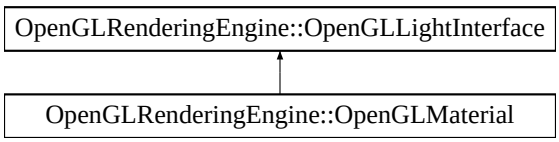
14.0.0.0

5.74 OpenGLRenderingEngine::OpenGLMaterial Class Reference

This class encapsulates usage of OpenGL materials.

`#include <OpenGLMaterial.h>`

Inheritance diagram for OpenGLRenderingEngine::OpenGLMaterial:



Public Member Functions

- void **setAmbientColor** (float r, float g, float b, float a)
- void **setAmbientColor** (const float ambientColor[4])
- void **setDiffuseColor** (float r, float g, float b, float a)
- void **setDiffuseColor** (const float diffuseColor[4])
- void **setSpecularColor** (float r, float g, float b, float a)
- void **setSpecularColor** (const float specularColor[4])
- void **setShininess** (float shininess)
- void **enable** (int lightId=GL_LIGHT0, bool usePositionalLight=false) override
- void **update** (int lightId=GL_LIGHT0, bool usePositionalLight=false, float updateX=0.0f, float updateY=0.0f, float updateZ=0.0f) override
- void **disable** () const override
- **OpenGLMaterial** (bool useColorMaterial) noexcept
- **OpenGLMaterial** (const OpenGLMaterial &)=delete
- **OpenGLMaterial** (OpenGLMaterial &&)=delete
- OpenGLMaterial & **operator=** (const OpenGLMaterial &)=delete
- OpenGLMaterial & **operator=** (OpenGLMaterial &&)=delete

Private Attributes

- float **ambientColor_** [4] = { 0.0f }
- float **diffuseColor_** [4] = { 0.0f }
- float **specularColor_** [4] = { 0.0f }
- float **shininess_** = 0.0f
- bool **useColorMaterial_** = false

5.74.1 Detailed Description

This class encapsulates usage of OpenGL materials.

Author

Thanos Theo, 2009-2018

Version

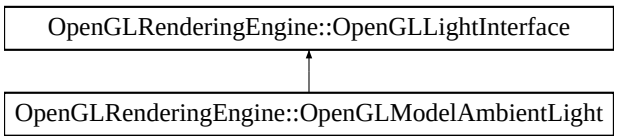
14.0.0.0

5.75 OpenGLRenderingEngine::OpenGLModelAmbientLight Class Reference

This class encapsulates usage of an OpenGL model ambient light.

`#include <OpenGLModelAmbientLight.h>`

Inheritance diagram for OpenGLRenderingEngine::OpenGLModelAmbientLight:



Public Member Functions

- void **setModelAmbientColor** (float r, float g, float b, float a)
- void **setModelAmbientColor** (const float modelAmbientColor[4])
- void **enable** (int lightId=GL_LIGHT0, bool usePositionalLight=false) override
- void **update** (int lightId=GL_LIGHT0, bool usePositionalLight=false, float updateX=0.0f, float updateY=0.0f, float updateZ=0.0f) override
- void **disable** () const override
- **OpenGLModelAmbientLight** (const OpenGLModelAmbientLight &)=delete
- **OpenGLModelAmbientLight** (OpenGLModelAmbientLight &&)=delete
- OpenGLModelAmbientLight & **operator=** (const OpenGLModelAmbientLight &)=delete
- OpenGLModelAmbientLight & **operator=** (OpenGLModelAmbientLight &&)=delete

Private Attributes

- float **modelAmbientColor_** [4] = { 0.0f }

5.75.1 Detailed Description

This class encapsulates usage of an OpenGL model ambient light.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.76 OpenGLRenderingEngine::OpenGLPerlinNoise Class Reference

This class provides Perlin Noise functionality for GLSL Shaders usage through a 3D OpenGL texture.

`#include <OpenGLPerlinNoise.h>`

Public Member Functions

- void initPerlinNoise (int frequency, std::vector< int > &p, std::vector< double > &g1, std::vector< std::vector< double >> &g2, std::vector< std::vector< double >> &g3)

  Initializes all the Perlin Noise arrays.
- double perlinNoise1 (double xIn, int frequency)

  1D Perlin Noise function.
- double perlinNoise2 (double xIn, double yIn, int frequency)

  2D Perlin Noise function.
- double perlinNoise3 (double xIn, double yIn, double zIn, int frequency)

  3D Perlin Noise function.
- double perlinNoise1D (double xIn, double alpha, double beta, int n, int frequency)

  1D Perlin Noise harmonic summing function.
- double perlinNoise2D (double xIn, double yIn, double alpha, double beta, int n, int frequency)

  2D Perlin Noise harmonic summing function.
- double perlinNoise3D (double xIn, double yIn, double zIn, double alpha, double beta, int n, int frequency)

  3D Perlin Noise harmonic summing function.
- void makePerlinNoise3DTexture ()

  Makes the Perlin Noise 3D texture.
- void initPerlinNoise3DTexture (GLenum textureUnit=0)

  Initializes the Perlin Noise 3D texture with a given active texture unit.
- **OpenGLPerlinNoise** (bool useSimplexNoise=false) noexcept
- **OpenGLPerlinNoise** (const OpenGLPerlinNoise &)=delete
- **OpenGLPerlinNoise** (OpenGLPerlinNoise &&)=delete
- OpenGLPerlinNoise & **operator=** (const OpenGLPerlinNoise &)=delete
- OpenGLPerlinNoise & **operator=** (OpenGLPerlinNoise &&)=delete

Private Member Functions

- void **makePerlinNoise3DTextureCalculation** (int f, int inc, int frequency, double amp)
- void **bindTexture** (GLenum textureUnit=0) const
- void **unbind** (GLenum textureUnit) const

Private Attributes

- bool **useSimplexNoise_** = false
- GLuint **perlinNoise3DTextureID_** = 0
- Utils::Randomizers::RandomRNGWELL512 **random_**
- std::vector< GLubyte > **perlinNoise3DTextureData_**

5.76.1 Detailed Description

This class provides Perlin Noise functionality for GLSL Shaders usage through a 3D OpenGL texture.

Coherent Perlin Noise function over 1, 2 or 3 dimensions (original author Prof. Ken Perlin). Modifications by John Kessenich (GLSL Orange Book & setNoiseFrequency() support). N-CP conversion by Thanos Theo.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.76.2 Member Function Documentation

5.76.2.1 initPerlinNoise3DTexture()

```
void OpenGLPerlinNoise::initPerlinNoise3DTexture (
            GLenum textureUnit = 0 )
```

Initializes the Perlin Noise 3D texture with a given active texture unit.

Overloaded version of the method above that selects an active texture unit for the Perlin Noise 3D texture.

5.76.2.2 perlinNoise1D()

```
double OpenGLPerlinNoise::perlinNoise1D (
            double xIn,
            double alpha,
            double beta,
            int n,
            int frequency )
```

1D Perlin Noise harmonic summing function.

In what follows "alpha" is the weight when the sum is formed. Typically it is 2, as this approaches 1 the function is noisier. "beta" is the harmonic scaling/spacing, typically 2.

5.76.2.3 perlinNoise2D()

```
double OpenGLPerlinNoise::perlinNoise2D (
            double xIn,
            double yIn,
            double alpha,
            double beta,
            int n,
            int frequency )
```

2D Perlin Noise harmonic summing function.

In what follows "alpha" is the weight when the sum is formed. Typically it is 2, as this approaches 1 the function is noisier. "beta" is the harmonic scaling/spacing, typically 2.

5.76.2.4 perlinNoise3D()

```
double OpenGLPerlinNoise::perlinNoise3D (
            double xIn,
            double yIn,
            double zIn,
            double alpha,
            double beta,
            int n,
            int frequency )
```

3D Perlin Noise harmonic summing function.

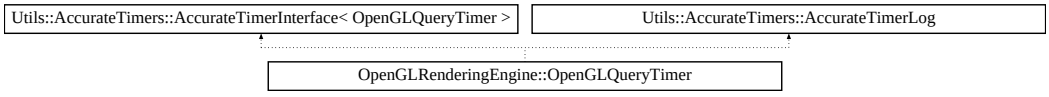
In what follows "alpha" is the weight when the sum is formed. Typically it is 2, as this approaches 1 the function is noisier. "beta" is the harmonic scaling/spacing, typically 2.

5.77 OpenGLRenderingEngine::OpenGLQueryTimer Class Reference

This class contains an AccurateTimers encapsulation of OpenGL query timers.

`#include <OpenGLQueryTimer.h>`

Inheritance diagram for OpenGLRenderingEngine::OpenGLQueryTimer:



Public Member Functions

- void **startTimer** ()
- void **stopTimer** ()
- double **getElapsedTimeInNanoSecs** ()
- double **getElapsedTimeInMicroSecs** ()
- double **getElapsedTimeInMilliSecs** ()
- double **getElapsedTimeInSecs** ()
- double **getMeanTimeInNanoSecs** ()
- double **getMeanTimeInMicroSecs** ()
- double **getMeanTimeInMilliSecs** ()
- double **getMeanTimeInSecs** ()
- double **getDecimalElapsedTimeInMicroSecs** ()
- double **getDecimalElapsedTimeInMilliSecs** ()
- double **getDecimalElapsedTimeInSecs** ()
- double **getDecimalMeanTimeInMicroSecs** ()
- double **getDecimalMeanTimeInMilliSecs** ()
- double **getDecimalMeanTimeInSecs** ()
- **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

5.77.1 Detailed Description

This class contains an AccurateTimers encapsulation of OpenGL query timers.

OpenGLQueryTimer.h:

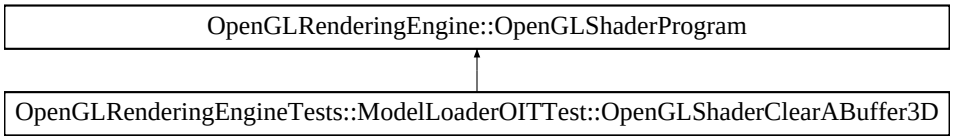
This class contains an AccurateTimers encapsulation of OpenGL query timers. Note: no virtual destructor is needed for data-oriented design ie no up-casting should ever be used.

Author

Thanos Theo, 2018

5.78 OpenGLRenderingEngineTests::ModelLoaderOITTest::OpenGLShaderClearABuffer3D Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ModelLoaderOITTest::OpenGLShaderClearABuffer3D:



Public Member Functions

- **OpenGLShaderClearABuffer3D** (ModelLoaderOITTest *thisModelLoaderOITTest, OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

- void initializeShaderProgram () override

  initialize shader program function to override

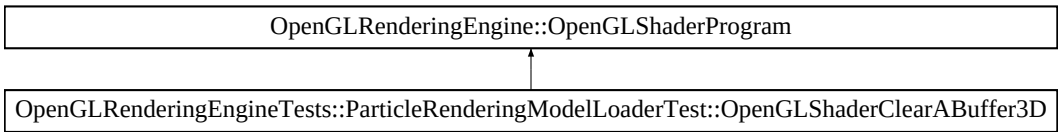
Private Attributes

- ModelLoaderOITTest * **modelLoaderOITTest** = nullptr

Additional Inherited Members

5.79 OpenGLRenderingEngineTests::ParticleRenderingModelLoaderTest::OpenGLShaderClearABuffer3D Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ParticleRenderingModelLoaderTest::OpenGLShaderClearABuffer3D:



Public Member Functions

- **OpenGLShaderClearABuffer3D** (ParticleRenderingModelLoaderTest *thisParticleRenderingModelLoaderTest, OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

- void initializeShaderProgram () override

  initialize shader program function to override

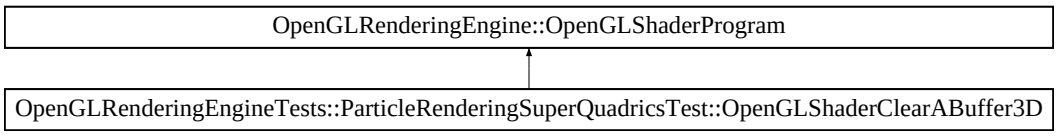
Private Attributes

- ParticleRenderingModelLoaderTest * **particleRenderingModelLoaderTest** = nullptr

Additional Inherited Members

5.80 OpenGLRenderingEngineTests::ParticleRenderingSuperQuadricsTest::OpenGL↩ ShaderClearABuffer3D Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ParticleRenderingSuperQuadricsTest::OpenGLShader↩ ClearABuffer3D:



Public Member Functions

- **OpenGLShaderClearABuffer3D** (ParticleRenderingSuperQuadricsTest *thisParticleRenderingSuper↩ QuadricsTest, OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

- void initializeShaderProgram () override

  initialize shader program function to override

Private Attributes

- ParticleRenderingSuperQuadricsTest * **particleRenderingSuperQuadricsTest** = nullptr

Additional Inherited Members

5.81 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 &shaderLibraryName, int shaderType)

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

  link shader program
- **OpenGLShaderCompileAndLink** (OpenGLDriverInfo *openGLDriverInfo, OpenGLShaderGLSLPreProcessorCommands *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_**

5.81.1 Detailed Description

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

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.81.2 Member Function Documentation

5.81.2.1 checkInfoLog()

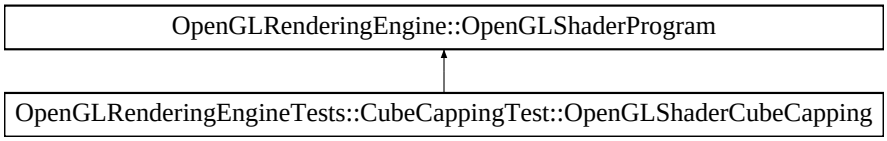
```
void OpenGLShaderCompileAndLink::checkInfoLog (
            const std::string & shaderName,
            GLuint obj ) const  [private]
```

check GL info log function

Checks the OpenGL info log of the shader loading process.

5.82 OpenGLRenderingEngineTests::CubeCappingTest::OpenGLShaderCubeCapping Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::CubeCappingTest::OpenGLShaderCubeCapping:



Public Member Functions

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

Private Member Functions

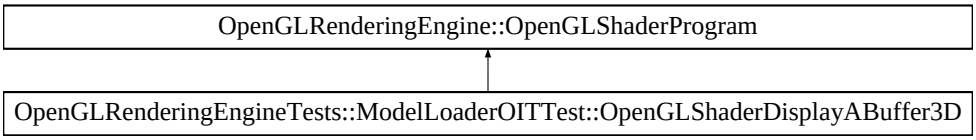
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.83 OpenGLRenderingEngineTests::ModelLoaderOITTest::OpenGLShaderDisplayABuffer3D Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ModelLoaderOITTest::OpenGLShaderDisplayABuffer3D:



Public Member Functions

- **OpenGLShaderDisplayABuffer3D** (ModelLoaderOITTest *thisModelLoaderOITTest, OpenGLRendering↩ Engine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

- void initializeShaderProgram () override

  initialize shader program function to override

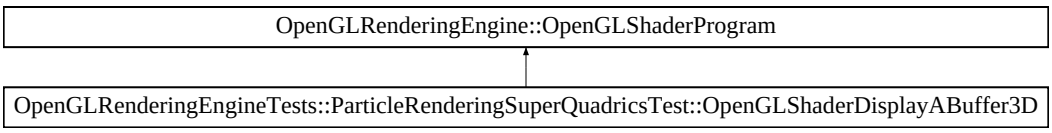
Private Attributes

- ModelLoaderOITTest * **modelLoaderOITTest** = nullptr

Additional Inherited Members

5.84 OpenGLRenderingEngineTests::ParticleRenderingSuperQuadricsTest::OpenGL↩ ShaderDisplayABuffer3D Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ParticleRenderingSuperQuadricsTest::OpenGLShader↩ DisplayABuffer3D:



Public Member Functions

- **OpenGLShaderDisplayABuffer3D** (ParticleRenderingSuperQuadricsTest *thisParticleRenderingSuper↩ QuadricsTest, OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

- void initializeShaderProgram () override

  initialize shader program function to override

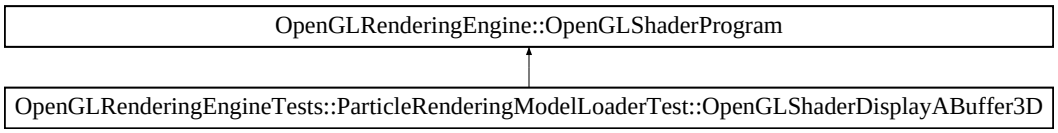
Private Attributes

- ParticleRenderingSuperQuadricsTest * **particleRenderingSuperQuadricsTest** = nullptr

Additional Inherited Members

5.85 OpenGLRenderingEngineTests::ParticleRenderingModelLoaderTest::OpenGL↩ ShaderDisplayABuffer3D Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ParticleRenderingModelLoaderTest::OpenGLShader↩ DisplayABuffer3D:



Public Member Functions

- **OpenGLShaderDisplayABuffer3D** (ParticleRenderingModelLoaderTest ∗thisParticleRenderingModel↩ LoaderTest, OpenGLRenderingEngine::OpenGLDriverInfo ∗openGLDriverInfo)

Private Member Functions

- void initializeShaderProgram () override

  initialize shader program function to override

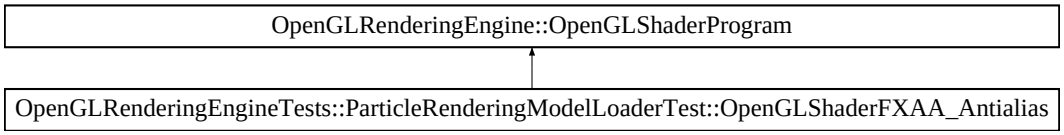
Private Attributes

- ParticleRenderingModelLoaderTest ∗ **particleRenderingModelLoaderTest** = nullptr

Additional Inherited Members

5.86 OpenGLRenderingEngineTests::ParticleRenderingModelLoaderTest::OpenGL↩ ShaderFXAA_Antialias Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ParticleRenderingModelLoaderTest::OpenGLShaderFX↩ AA_Antialias:



Public Member Functions

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

Private Member Functions

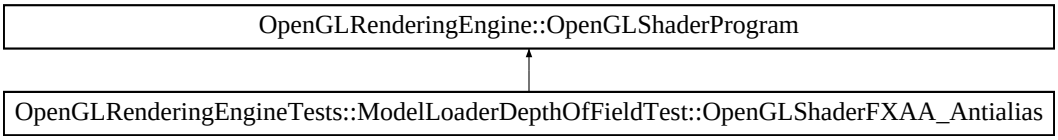
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.87 OpenGLRenderingEngineTests::ModelLoaderDepthOfFieldTest::OpenGLShaderFXAA_Antialias Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ModelLoaderDepthOfFieldTest::OpenGLShaderFXAA_Antialias:



Public Member Functions

- **OpenGLShaderFXAA_Antialias** (OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

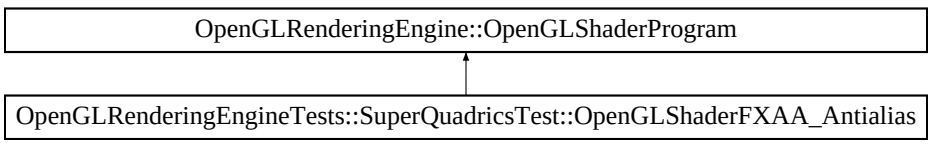
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.88 OpenGLRenderingEngineTests::SuperQuadricsTest::OpenGLShaderFXAA_Antialias Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::SuperQuadricsTest::OpenGLShaderFXAA_Antialias:



Public Member Functions

- **OpenGLShaderFXAA_Antialias** (OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

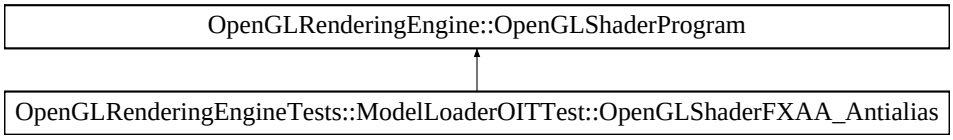
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.89 OpenGLRenderingEngineTests::ModelLoaderOITTest::OpenGLShaderFXAA_Antialias Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ModelLoaderOITTest::OpenGLShaderFXAA_Antialias:



Public Member Functions

- **OpenGLShaderFXAA_Antialias** (OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

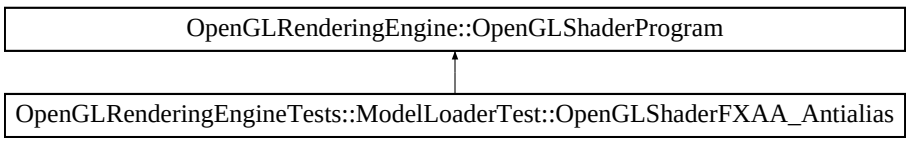
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.90 OpenGLRenderingEngineTests::ModelLoaderTest::OpenGLShaderFXAA_Antialias Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ModelLoaderTest::OpenGLShaderFXAA_Antialias:



Public Member Functions

- **OpenGLShaderFXAA_Antialias** (OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

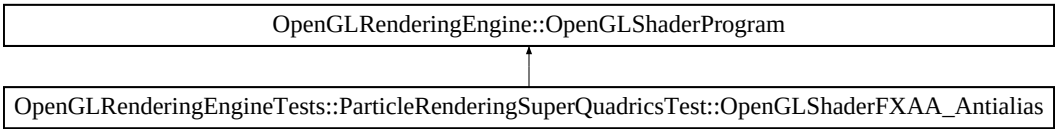
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.91 OpenGLRenderingEngineTests::ParticleRenderingSuperQuadricsTest::OpenGL↩ ShaderFXAA_Antialias Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ParticleRenderingSuperQuadricsTest::OpenGLShaderF↩ XAA_Antialias:



Public Member Functions

- **OpenGLShaderFXAA_Antialias** (OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

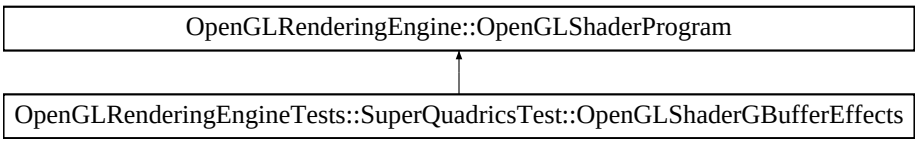
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.92 OpenGLRenderingEngineTests::SuperQuadricsTest::OpenGLShaderGBufferEffects Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::SuperQuadricsTest::OpenGLShaderGBufferEffects:



Public Member Functions

- **OpenGLShaderGBufferEffects** (OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

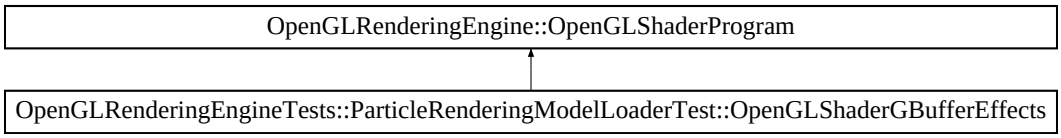
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.93 OpenGLRenderingEngineTests::ParticleRenderingModelLoaderTest::OpenGLShaderGBufferEffects Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ParticleRenderingModelLoaderTest::OpenGLShaderGBufferEffects:



Public Member Functions

- **OpenGLShaderGBufferEffects** (OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

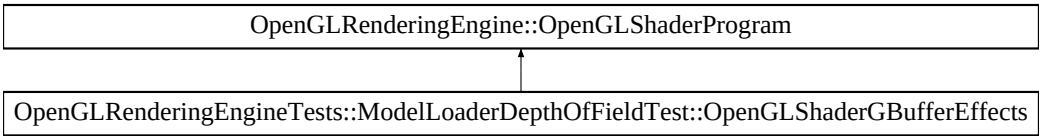
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.94 OpenGLRenderingEngineTests::ModelLoaderDepthOfFieldTest::OpenGLShaderGBufferEffects Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ModelLoaderDepthOfFieldTest::OpenGLShaderGBufferEffects:



Public Member Functions

- **OpenGLShaderGBufferEffects** (OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

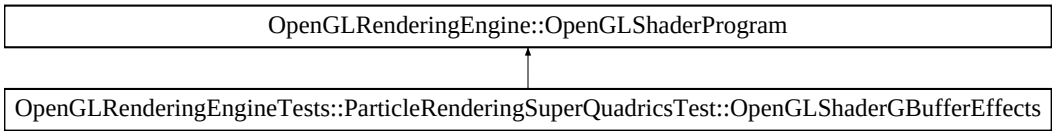
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.95 OpenGLRenderingEngineTests::ParticleRenderingSuperQuadricsTest::OpenGLShaderGBufferEffects Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ParticleRenderingSuperQuadricsTest::OpenGLShaderGBufferEffects:



Public Member Functions

- **OpenGLShaderGBufferEffects** (OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

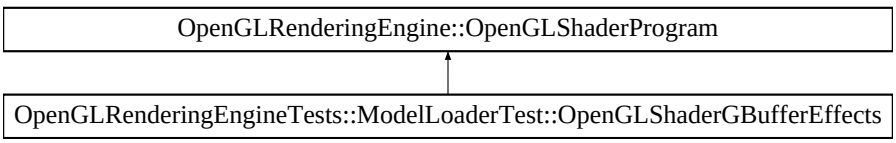
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.96 OpenGLRenderingEngineTests::ModelLoaderTest::OpenGLShaderGBufferEffects Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ModelLoaderTest::OpenGLShaderGBufferEffects:



Public Member Functions

- **OpenGLShaderGBufferEffects** (OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

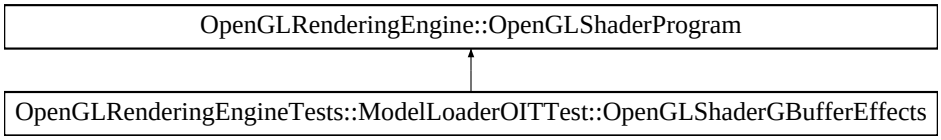
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.97 OpenGLRenderingEngineTests::ModelLoaderOITTest::OpenGLShaderGBuffer↩ Effects Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ModelLoaderOITTest::OpenGLShaderGBufferEffects:



Public Member Functions

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

Private Member Functions

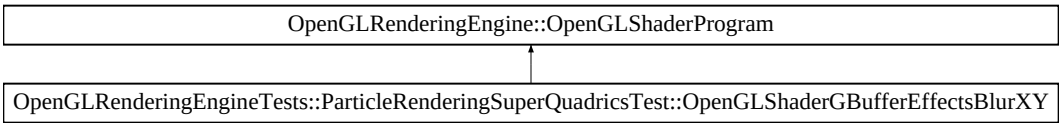
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.98 OpenGLRenderingEngineTests::ParticleRenderingSuperQuadricsTest::OpenGL↩ ShaderGBufferEffectsBlurXY Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ParticleRenderingSuperQuadricsTest::OpenGLShaderG↩ BufferEffectsBlurXY:



Public Member Functions

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

Private Member Functions

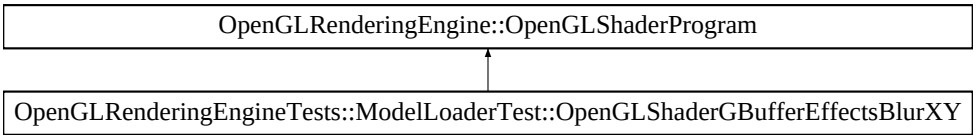
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.99 OpenGLRenderingEngineTests::ModelLoaderTest::OpenGLShaderGBufferEffectsBlurXY Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ModelLoaderTest::OpenGLShaderGBufferEffectsBlurXY:



Public Member Functions

- **OpenGLShaderGBufferEffectsBlurXY** (OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

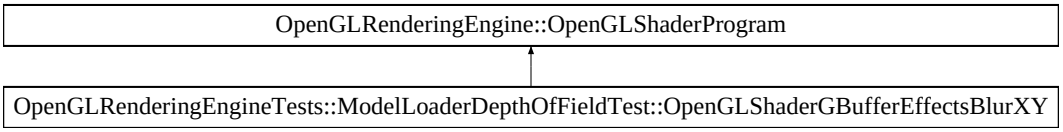
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.100 OpenGLRenderingEngineTests::ModelLoaderDepthOfFieldTest::OpenGLShaderGBufferEffectsBlurXY Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ModelLoaderDepthOfFieldTest::OpenGLShaderGBufferEffectsBlurXY:



Public Member Functions

- **OpenGLShaderGBufferEffectsBlurXY** (OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

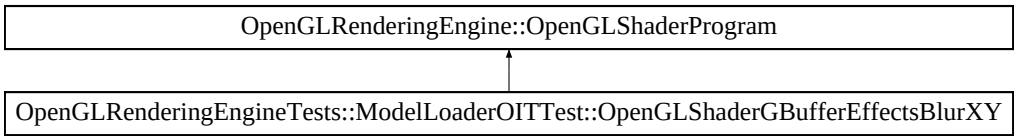
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.101 OpenGLRenderingEngineTests::ModelLoaderOITTest::OpenGLShaderGBufferEffectsBlurXY Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ModelLoaderOITTest::OpenGLShaderGBufferEffectsBlurXY:



Public Member Functions

- **OpenGLShaderGBufferEffectsBlurXY** (OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

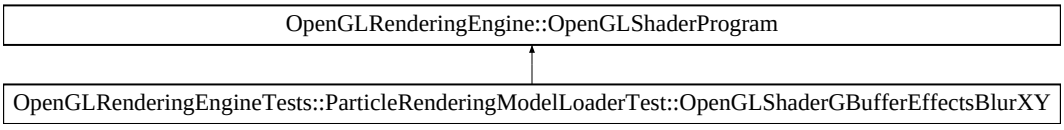
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.102 OpenGLRenderingEngineTests::ParticleRenderingModelLoaderTest::OpenGLShaderGBufferEffectsBlurXY Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ParticleRenderingModelLoaderTest::OpenGLShaderGBufferEffectsBlurXY:



Public Member Functions

- **OpenGLShaderGBufferEffectsBlurXY** (OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

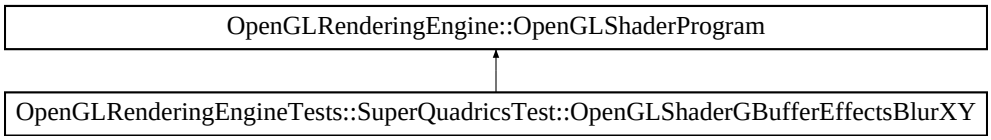
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.103 OpenGLRenderingEngineTests::SuperQuadricsTest::OpenGLShaderGBufferEffectsBlurXY Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::SuperQuadricsTest::OpenGLShaderGBufferEffectsBlurXY:



Public Member Functions

- **OpenGLShaderGBufferEffectsBlurXY** (OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

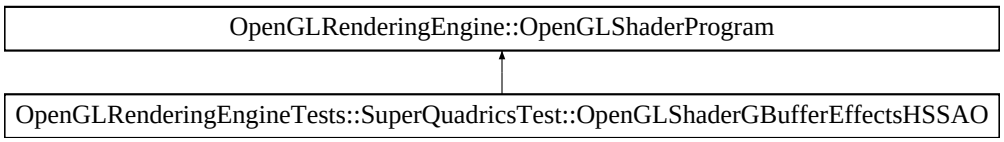
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.104 OpenGLRenderingEngineTests::SuperQuadricsTest::OpenGLShaderGBufferEffectsHSSAO Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::SuperQuadricsTest::OpenGLShaderGBufferEffectsHSSAO:



Public Member Functions

- **OpenGLShaderGBufferEffectsHSSAO** (OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

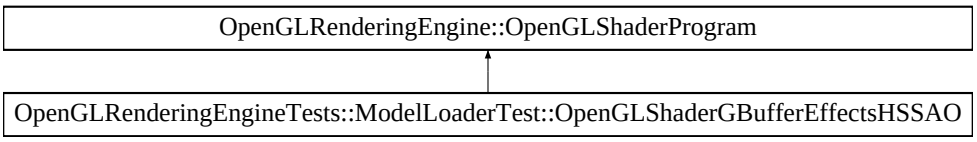
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.105 OpenGLRenderingEngineTests::ModelLoaderTest::OpenGLShaderGBufferEffectsHSSAO Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ModelLoaderTest::OpenGLShaderGBufferEffectsHSSAO:



Public Member Functions

- **OpenGLShaderGBufferEffectsHSSAO** (OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

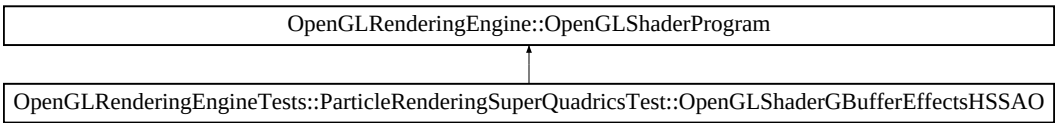
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.106 OpenGLRenderingEngineTests::ParticleRenderingSuperQuadricsTest::OpenGLShaderGBufferEffectsHSSAO Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ParticleRenderingSuperQuadricsTest::OpenGLShaderGBufferEffectsHSSAO:



Public Member Functions

- **OpenGLShaderGBufferEffectsHSSAO** (OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

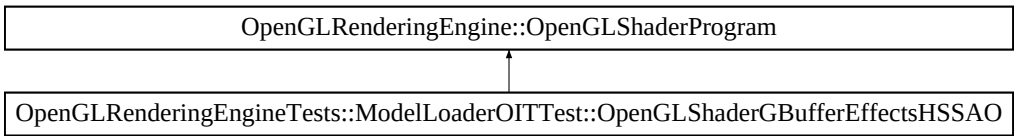
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.107 OpenGLRenderingEngineTests::ModelLoaderOITTest::OpenGLShaderGBufferEffectsHSSAO Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ModelLoaderOITTest::OpenGLShaderGBufferEffectsHSSAO:



Public Member Functions

- **OpenGLShaderGBufferEffectsHSSAO** (OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

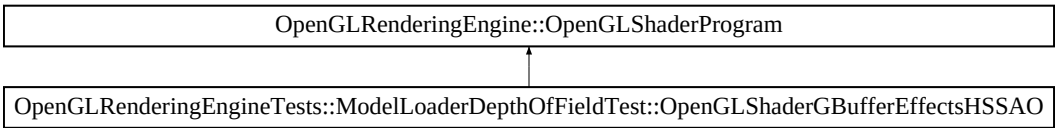
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.108 OpenGLRenderingEngineTests::ModelLoaderDepthOfFieldTest::OpenGLShaderGBufferEffectsHSSAO Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ModelLoaderDepthOfFieldTest::OpenGLShaderGBufferEffectsHSSAO:



Public Member Functions

- **OpenGLShaderGBufferEffectsHSSAO** (OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

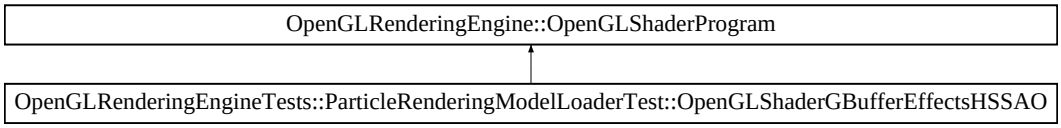
- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.109 OpenGLRenderingEngineTests::ParticleRenderingModelLoaderTest::OpenGL↩ ShaderGBufferEffectsHSSAO Class Reference

Inheritance diagram for OpenGLRenderingEngineTests::ParticleRenderingModelLoaderTest::OpenGLShaderG↩ BufferEffectsHSSAO:



Public Member Functions

- **OpenGLShaderGBufferEffectsHSSAO** (OpenGLRenderingEngine::OpenGLDriverInfo *openGLDriverInfo)

Private Member Functions

- void initializeShaderProgram () override

  initialize shader program function to override

Additional Inherited Members

5.110 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_**

5.110.1 Detailed Description

This class is responsible for the GLSL shader preprocessor process.

Author

Thanos Theo, 2009-2018

Version

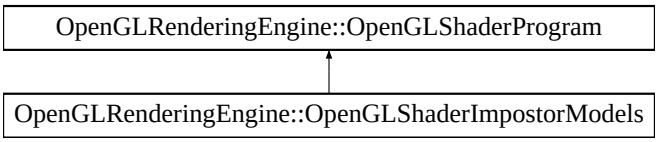
14.0.0.0

5.111 OpenGLRenderingEngine::OpenGLShaderImpostorModels Class Reference

This class loads & encapsulates usage of the GLSL shader point sphere models.

`#include <OpenGLShaderImpostorModels.h>`

Inheritance diagram for OpenGLRenderingEngine::OpenGLShaderImpostorModels:



Public Member Functions

- void initializeShaderProgram () override

  initialize shader program function to override
- void **useProgramAndUniforms** (GLint numberOfLights, GLint lightingModel, GLint useOrenNayarDiffuse↩ Model, GLint useColorMaterial, GLfloat sceneScaleFactor, GLint width, GLint height, GLfloat radiusFactor, GLint pointAntialias, GLint perspectiveCorrection, GLint texturing, GLuint activeTextureUnitFor2DTexture, G↩ Lint sphericalMapping, GLint environmentMapping, GLint fog, GLint gammaCorrection, GLfloat opacity, GLint useGBuffer, GLint screenWidth=512, GLint screenHeight=512, GLint aBuffer3DSize=32, GLint useOwn↩ PackingMethodsForScreenshots=0, GLint useRandomlyGeneratedCylinderVariables=1, GLfloat cylinder↩ Radius=1.0f/250.0f)
- **OpenGLShaderImpostorModels** (OpenGLDriverInfo *openGLDriverInfo, bool isQuadSphereModel, bool useGeometryShader, bool useTessellationShader, bool useTessellationQuadShader, bool useCylinder↩ Impostor=false, bool useFP64=false, bool useOITShaders=false, GLuint aBuffer3DMode=1, GLuint a↩ Buffer3DCounterUnit=1, GLuint aBuffer3DDataUnit=2, GLuint aBuffer3DLinkedListAtomicCounterUnit=3, GLuint aBuffer3DLinkedListOffsetUnit=4, GLuint aBuffer3DLinkedListUnit=5, GLuint oitUseDeferred↩ Shading=0) noexcept
- **OpenGLShaderImpostorModels** (const OpenGLShaderImpostorModels &)=delete
- **OpenGLShaderImpostorModels** (OpenGLShaderImpostorModels &&)=delete
- OpenGLShaderImpostorModels & **operator=** (const OpenGLShaderImpostorModels &)=delete
- OpenGLShaderImpostorModels & **operator=** (OpenGLShaderImpostorModels &&)=delete

Private Attributes

- bool **isQuadSphereModel_** = false
- bool **useGeometryShader_** = false
- bool **useTessellationShader_** = false
- bool **useTessellationQuadShader_** = false
- bool **useCylinderImpostor_** = false
- bool **useFP64_** = false
- bool **useOITShaders_** = false
- GLuint **aBuffer3DMode_** = 0
- GLuint **aBuffer3DCounterUnit_** = 0
- GLuint **aBuffer3DDataUnit_** = 0
- GLuint **aBuffer3DLinkedListAtomicCounterUnit_** = 0
- GLuint **aBuffer3DLinkedListOffsetUnit_** = 0
- GLuint **aBuffer3DLinkedListUnit_** = 0
- GLuint **oitUseDeferredShading_** = 0

Additional Inherited Members

5.111.1 Detailed Description

This class loads & encapsulates usage of the GLSL shader point sphere models.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.112 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

5.112.1 Detailed Description

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

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.113 OpenGLRenderingEngine::OpenGLShaderProgram Class Reference

This abstract class encapsulates usage of a GLSL program.

`#include <OpenGLShaderProgram.h>`

Inheritance diagram for OpenGLRenderingEngine::OpenGLShaderProgram:

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_**

5.113.1 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

5.113.2 Member Function Documentation

5.113.2.1 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

5.113.2.2 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

5.113.2.3 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

5.113.2.4 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

5.113.2.5 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.

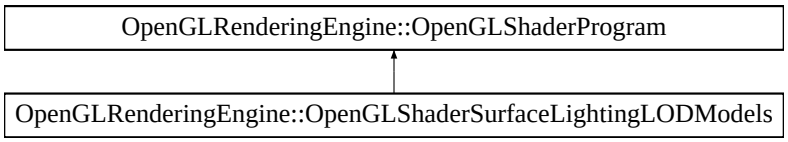
This parameter must be GL_INT_10_10_10_2 or GL_UNSIGNED_INT_10_10_10_2 to specify signed or unsigned data, respectively normalized parameter: if GL_TRUE, then the values are to be converted to floating point values by normalizing. Otherwise, they are converted directly to floating point values

5.114 OpenGLRenderingEngine::OpenGLShaderSurfaceLightingLODModels Class Reference

This class loads & encapsulates usage of the 5 GLSL shader lighting models with LOD (real-time adaptive tessellation with GL 4.0+): Phong, Blinn-Phong, Gaussian, Toon & Gooch.

`#include <OpenGLShaderSurfaceLightingLODModels.h>`

Inheritance diagram for OpenGLRenderingEngine::OpenGLShaderSurfaceLightingLODModels:



Public Member Functions

- void initializeShaderProgram () override

  initialize shader program function to override
- void **useProgramAndUniforms** (GLint LODModel, GLfloat tessellationAlpha, GLfloat tessellationLevel, GLint useAdaptiveTessellation, GLint useTrianglePositionAdaptiveTessellationMetric, GLint useLightAdaptiveTessellationMetric, GLint useContourAdaptiveTessellationMetric, GLint numberOfLights, GLint lightingModel, GLint useOrenNayarDiffuseModel, GLint useFresnelFactorSchlickApproximationSpecularModel, GLint useColorMaterial, GLfloat sceneScaleFactor, GLint shrinkTriangles, GLint normals, GLint texturing, GLuint activeTextureUnitFor2DTexture, GLint sphericalMapping, GLint environmentMapping, GLint enforceColor, GLint blackOrWhite, GLint fog, GLint gammaCorrection, GLfloat opacity, GLfloat timer, GLint useGBuffer, GLint screenWidth=512, GLint screenHeight=512, GLint aBuffer3DSize=32, GLint useOwnPackingMethodsForScreenshots=4)
- **OpenGLShaderSurfaceLightingLODModels** (OpenGLDriverInfo *openGLDriverInfo, bool useGeometryShader, bool applyNormalsGeometry, bool useOITShaders=false, GLuint aBuffer3DMode=1, GLuint aBuffer3DCounterUnit=1, GLuint aBuffer3DDataUnit=2, GLuint aBuffer3DLinkedListAtomicCounterUnit=3, GLuint aBuffer3DLinkedListOffsetUnit=4, GLuint aBuffer3DLinkedListUnit=5, GLuint oitUseDeferredShading=0) noexcept
- **OpenGLShaderSurfaceLightingLODModels** (const OpenGLShaderSurfaceLightingLODModels &)=delete
- **OpenGLShaderSurfaceLightingLODModels** (OpenGLShaderSurfaceLightingLODModels &&)=delete
- OpenGLShaderSurfaceLightingLODModels & **operator=** (const OpenGLShaderSurfaceLightingLODModels &)=delete
- OpenGLShaderSurfaceLightingLODModels & **operator=** (OpenGLShaderSurfaceLightingLODModels &&)=delete

Private Attributes

- bool **useGeometryShader_** = false
- bool **applyNormalsGeometry_** = false
- bool **useOITShaders_** = false
- GLuint **aBuffer3DMode_** = 0
- GLuint **aBuffer3DCounterUnit_** = 0
- GLuint **aBuffer3DDataUnit_** = 0
- GLuint **aBuffer3DLinkedListAtomicCounterUnit_** = 0
- GLuint **aBuffer3DLinkedListOffsetUnit_** = 0
- GLuint **aBuffer3DLinkedListUnit_** = 0
- GLuint **oitUseDeferredShading_** = 0

Additional Inherited Members

5.114.1 Detailed Description

This class loads & encapsulates usage of the 5 GLSL shader lighting models with LOD (real-time adaptive tessellation with GL 4.0+): Phong, Blinn-Phong, Gaussian, Toon & Gooch.

Author

Thanos Theo, 2009-2018

Version

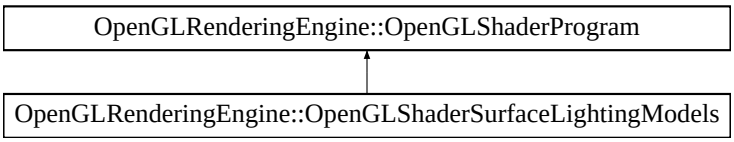
14.0.0.0

5.115 OpenGLRenderingEngine::OpenGLShaderSurfaceLightingModels Class Reference

This class loads & encapsulates usage of the 5 GLSL shader lighting models: Phong, Blinn-Phong, Gaussian, Toon & Gooch.

`#include <OpenGLShaderSurfaceLightingModels.h>`

Inheritance diagram for OpenGLRenderingEngine::OpenGLShaderSurfaceLightingModels:



Public Member Functions

- void initializeShaderProgram () override

  initialize shader program function to override
- void **useProgramAndUniforms** (GLint numberOfLights, GLint lightingModel, GLint useOrenNayar↩ DiffuseModel, GLint useFresnelFactorSchlickApproximationSpecularModel, GLint useColorMaterial, GLfloat sceneScaleFactor, GLint shrinkTriangles, GLint normals, GLint texturing, GLuint activeTextureUnitFor2D↩ Texture, GLint sphericalMapping, GLint environmentMapping, GLint enforceColor, GLint blackOrWhite, GLint fog, GLint gammaCorrection, GLfloat opacity, GLfloat timer, GLint useGBuffer, GLint screenWidth=512, GLint screenHeight=512, GLint aBuffer3DSize=32, GLint useOwnPackingMethodsForScreenshots=0)
- **OpenGLShaderSurfaceLightingModels** (OpenGLDriverInfo *openGLDriverInfo, bool useGeometryShader, bool applyNormalsGeometry, bool useOITShaders=false, GLuint aBuffer3DMode=1, GLuint aBuffer3D↩ CounterUnit=1, GLuint aBuffer3DDataUnit=2, GLuint aBuffer3DLinkedListAtomicCounterUnit=3, GLuint a↩ Buffer3DLinkedListOffsetUnit=4, GLuint aBuffer3DLinkedListUnit=5, GLuint oitUseDeferredShading=0) noexcept
- **OpenGLShaderSurfaceLightingModels** (const OpenGLShaderSurfaceLightingModels &)=delete
- **OpenGLShaderSurfaceLightingModels** (OpenGLShaderSurfaceLightingModels &&)=delete
- OpenGLShaderSurfaceLightingModels & **operator=** (const OpenGLShaderSurfaceLightingModels &)=delete
- OpenGLShaderSurfaceLightingModels & **operator=** (OpenGLShaderSurfaceLightingModels &&)=delete

Private Attributes

- bool **useGeometryShader_** = false
- bool **applyNormalsGeometry_** = false
- bool **useOITShaders_** = false
- GLuint **aBuffer3DMode_** = 0
- GLuint **aBuffer3DCounterUnit_** = 0
- GLuint **aBuffer3DDataUnit_** = 0
- GLuint **aBuffer3DLinkedListAtomicCounterUnit_** = 0
- GLuint **aBuffer3DLinkedListOffsetUnit_** = 0
- GLuint **aBuffer3DLinkedListUnit_** = 0
- GLuint **oitUseDeferredShading_** = 0

Additional Inherited Members

5.115.1 Detailed Description

This class loads & encapsulates usage of the 5 GLSL shader lighting models: Phong, Blinn-Phong, Gaussian, Toon & Gooch.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.116 OpenGLRenderingEngine::OpenGLSimplexNoise Struct Reference

This class includes the improved Simplex Perlin Noise static methods (original author Prof.

`#include <OpenGLSimplexNoise.h>`

Public Member Functions

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

Static Public Member Functions

- static double perlinNoise2 (double xIn, double yIn)

  2D Simplex Perlin Noise.
- static double perlinNoise3 (double xIn, double yIn, double zIn)

  3D Simplex Perlin Noise.
- static double perlinNoise4 (double xIn, double yIn, double zIn, double wIn)

  4D Simplex Perlin Noise.

5.116.1 Detailed Description

This class includes the improved Simplex Perlin Noise static methods (original author Prof.

Ken Perlin). Reference implementation of Simplex Noise, Stefan Gustavson, 2005.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.117 OpenGLRenderingEngine::OpenGLUniqueColorsGenerator Class Reference

This class encapsulates usage of creating unique colors based on a prime number generator.

`#include <OpenGLUniqueColorsGenerator.h>`

Public Member Functions

- std::vector< Utils::VectorTypes::float4 > createUniqueColorsBasedOnPrimeNumbers (std::size_t number↩ OfColors)

  Creates unique colors based on a prime number generator, same color hues reproducible for every run.
- **OpenGLUniqueColorsGenerator** (const OpenGLUniqueColorsGenerator &)=delete
- **OpenGLUniqueColorsGenerator** (OpenGLUniqueColorsGenerator &&)=delete
- OpenGLUniqueColorsGenerator & **operator=** (const OpenGLUniqueColorsGenerator &)=delete
- OpenGLUniqueColorsGenerator & **operator=** (OpenGLUniqueColorsGenerator &&)=delete

Private Attributes

- std::size_t primeNumberIndex1_ = 3

  prime numbers color generator values
- std::size_t **primeNumberIndex2_** = 18
- std::size_t **primeNumberIndex3_** = 27
- Utils::Randomizers::RandomRNGWELL512 random_

  the randomizer used for random colors

5.117.1 Detailed Description

This class encapsulates usage of creating unique colors based on a prime number generator.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.117.2 Member Function Documentation

5.117.2.1 createUniqueColorsBasedOnPrimeNumbers()

```
vector< float4 > OpenGLUniqueColorsGenerator::createUniqueColorsBasedOnPrimeNumbers (
            std::size_t numberOfColors )
```

Creates unique colors based on a prime number generator, same color hues reproducible for every run.

Verified for 6023986 uniquely generated colors.

5.118 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

5.118.1 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

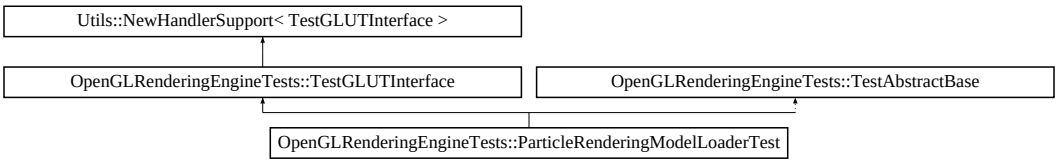
14.0.0.0

5.119 OpenGLRenderingEngineTests::ParticleRenderingModelLoaderTest Class Reference

ParticleRenderingModelLoaderTest is the 5th set of OpenGL rendering tests.

```
#include <ParticleRenderingModelLoaderTest.h>
```

Inheritance diagram for OpenGLRenderingEngineTests::ParticleRenderingModelLoaderTest:



Classes

- class OpenGLShaderClearABuffer3D
- class OpenGLShaderDisplayABuffer3D
- class OpenGLShaderFXAA_Antialias
- class OpenGLShaderGBufferEffects
- class OpenGLShaderGBufferEffectsBlurXY
- class OpenGLShaderGBufferEffectsHSSAO

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
- **ParticleRenderingModelLoaderTest** (int screenWidth, int screenHeight, const std::string &textureFileName, const std::string &modelFileName, const std::string &modelLoaderDescriptorFileName, bool multisample) noexcept
- **ParticleRenderingModelLoaderTest** (const ParticleRenderingModelLoaderTest &)=delete
- **ParticleRenderingModelLoaderTest** (ParticleRenderingModelLoaderTest &&)=delete
- ParticleRenderingModelLoaderTest & **operator=** (const ParticleRenderingModelLoaderTest &)=delete
- ParticleRenderingModelLoaderTest & **operator=** (ParticleRenderingModelLoaderTest &&)=delete

Private Types

- enum **AllCachedRenderingTests** : std::size_t {
  USE_NO_CACHING_METHOD_1 = 0, **USE_NO_CACHING_METHOD_2** = 1, **USE_VERTEX_ARRAYS** = 2, **USE_DISPLAY_LISTS** = 3,
  USE_VBOS = 4 }
- enum **AllLightingModels** : std::size_t {
  SIMPLE_LIGHTING = 0, **PHONG** = 1, **BLINN_PHONG** = 2, **GAUSSIAN** = 3,
  TOON = 4, **GOOCH** = 5 }
- enum **ImpostorShaderUsage** : std::size_t {
  POINT_SPHERE = 0, **QUAD_SPHERE_VS_FS** = 1, **QUAD_SPHERE_VS_GS_FS** = 2, **QUAD_SPHERE_VS_TESS_LOD_FS** = 3,
  QUAD_SPHERE_VS_TESS_QUAD_FS = 4, **CYLINDER_VS_GS_FS** = 5 }
- enum **ImpostorWireframeUsage** : std::size_t { **FULL_IMPOSTOR_WIREFRAME** = 0, **IMPOSTOR_POINTS_ONLY** = 1, **IMPOSTOR_LINES_ONLY** = 2, **CYLINDER_EXPLOSION** = 3 }
- enum **AllOITModes** : std::size_t {
  NO_OIT = 0, **A_BUFFER_3D_USE_TEXTURES** = 1, **A_BUFFER_3D_USE_BUFFERS** = 2, **A_BUFFER_3D_USE_LINKED_LIST_TEXTURES** = 3,
  A_BUFFER_3D_USE_LINKED_LIST_BUFFERS = 4 }
- enum **GBUfferEffectTypes** : std::size_t {
  NO_GBUFFER_EFFECT = 0, **DEPTH_OF_FIELD** = 1, **EDGE_ENHANCEMENT** = 2, **HSSAO** = 3,
  HSSAO_PLUS = 4 }

Private Member Functions

- void **prepareFog** () const
- void **prepareLighting** ()
- void **updateLighting** ()
- void **preparePointSphereShaders** ()
- void **prepareABuffer3DShaders** ()
- void **prepareGBufferEffectsShaders** ()
- void **prepareFXAA_AntialiasShaders** ()
- void **prepareTexture** ()
- void **prepareScene** ()
- void **prepareParticles** ()
- void **prepareEnvironmentMappingFBO** ()
- void **initEnviromentMappingFBO** () const
- void **drawEnvironmentMappingFBO** ()
- void **prepareGBufferEffectsFBOs** ()
- void **prepareHSSAOData** ()
- void **initGBufferEffectsFBOs** () const
- void **processGBufferEffectsFBOs** ()
- void **drawGBufferEffectsFBOs** ()
- void **prepareFXAA_AntialiasFBO** ()
- void **initFXAA_AntialiasFBO** () const
- void **drawFXAA_AntialiasFBO** ()
- void **initABuffer3D** ()
- bool **autoManageABuffer3D** ()
- void **deleteABuffer3D** ()
- void **writeABuffer3DBufferToFile** (GLbitfield barriers) const
- void **createParticlesVerticesArray** ()
- void **createParticlesColorsArray** ()
- void **createPointCylinderVerticesAndDirectionsArray** ()
- void **createParticlesCylinderColorsArray** ()
- void **createQuadColorsArray** ()

- void **createQuadTextureCoordsArray** ()
- void **createQuadVerticesArray** ()
- void **prepareVBOs** ()
- void **deleteVBOs** ()
- void **prepareCylinderVBOs** ()
- void **deleteCylinderVBOs** ()
- void **renderNoCaching** (bool skipReInitModelDataForRendering)
- void **renderVertexArrays** ()
- void **renderDisplayLists** ()
- void **renderVBOs** ()
- void **renderCylinderNoCaching** (bool skipReInitModelDataForRendering)
- void **renderCylinderVertexArrays** ()
- void **renderCylinderDisplayLists** ()
- void **renderCylinderVBOs** ()
- void **clearScreen** (bool useGBuffer=false) const
- void **disableLights** (bool disableAllLightSources=true) const
- void **enableLights** (bool enableAllLightSources=true)
- void **renderParticleScene** (bool environmentMappingRenderPass, bool useGBuffer=false)
- void **renderParticleGeometry** (bool environmentMappingRenderPass, bool useGBuffer)
- bool **isQuadVSFS** () const
- bool **isQuadVSTESSFS** () const
- void **drawString** (const char *str, int x, int y, const float color[4], void *font) const
- void **drawString3D** (const char *str, float position[3], const float color[4], void *font) const
- void **showInfo** ()
- int **howMuchCurrentlyAvailableVRAMMemory** () const
- std::string **howMuchCurrentlyAvailableVRAMMemoryString** () const
- void **showFPS** ()

Private Attributes

- AllCachedRenderingTests **currentCachedRenderingTest** = USE_DISPLAY_LISTS
- AllLightingModels **currentLightingModel** = BLINN_PHONG
- ImpostorShaderUsage **currentImpostorShader** = QUAD_SPHERE_VS_GS_FS
- ImpostorWireframeUsage **currentImpostorWireframe** = FULL_IMPOSTOR_WIREFRAME
- AllOITModes **currentOITMode** = NO_OIT
- GBUfferEffectTypes **currentGBUfferEffectType** = NO_GBUFFER_EFFECT
- OpenGLRenderingEngineTests::ParticleRenderingModelLoaderTest::OpenGLShaderClearABuffer3D * **openGLShaderClearABuffer3D** = nullptr
- OpenGLRenderingEngineTests::ParticleRenderingModelLoaderTest::OpenGLShaderDisplayABuffer3D * **openGLShaderDisplayABuffer3D** = nullptr
- OpenGLRenderingEngineTests::ParticleRenderingModelLoaderTest::OpenGLShaderGBufferEffectsBlurXY * **openGLShaderGBufferEffectsBlurXY** = nullptr
- OpenGLRenderingEngineTests::ParticleRenderingModelLoaderTest::OpenGLShaderGBufferEffectsHSSAO * **openGLShaderGBufferEffectsHSSAO** = nullptr
- OpenGLRenderingEngineTests::ParticleRenderingModelLoaderTest::OpenGLShaderGBufferEffects * **openGLShaderGBufferEffects** = nullptr
- OpenGLRenderingEngineTests::ParticleRenderingModelLoaderTest::OpenGLShaderFXAA_Antialias * **openGLShaderFXAA_Antialias** = nullptr
- OpenGLRenderingEngine::OpenGLModelAmbientLight * **openGLModelAmbientLight** = nullptr
- OpenGLRenderingEngine::OpenGLLight * **openGLLights** [NUMBER_OF_LIGHTS] = { nullptr }
- OpenGLRenderingEngine::OpenGLMaterial * **openGLMaterial** = nullptr
- OpenGLRenderingEngine::OpenGLShaderImpostorModels * **openGLShaderPointSphereModels** = nullptr
- OpenGLRenderingEngine::OpenGLShaderImpostorModels * **openGLShaderPointSphereModelsQuad↩ VFFS** = nullptr

- OpenGLRenderingEngine::OpenGLShaderImpostorModels * **openGLShaderPointSphereModelsQuad↩VFGSFS** = nullptr
- OpenGLRenderingEngine::OpenGLShaderImpostorModels * **openGLShaderPointSphereModelsQuad↩VFTESSLODFS** = nullptr
- OpenGLRenderingEngine::OpenGLShaderImpostorModels * **openGLShaderPointSphereModelsQuad↩VFTESSQuadFS** = nullptr
- OpenGLRenderingEngine::OpenGLShaderImpostorModels * **openGLShaderCylinderVFGSFS** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectForEnvironment↩Mapping** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectForGBufferPass1** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectRenderPass2** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectRenderPass3** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectForFXAA_Antialias** = nullptr
- OpenGLRenderingEngine::OpenGLILTexture * **openGLILTexture** = nullptr
- OpenGLRenderingEngine::OpenGLAssimpModelLoader * **openGLAssimpModelLoader** = nullptr
- float **radiusFactor** = 1.0f
- int **usePointAntialias** = 0
- int **usePerspectiveCorrection** = 0
- int **previousUsePerspectiveCorrection** = 0
- int **depthOfFieldBlurXYNumberOfSamplesRange** = 8
- float **depthOfFieldRange** = 10.0f
- float **depthOfFieldZFocus** = 0.5f
- GLuint **rotationTextureID** = 0
- std::vector< float > **rotationTextureData**
- std::vector< float > **depthSamplesData**
- bool **useTransparency** = false
- bool **reInitColorArrays** = false
- std::vector< GLfloat > **colors**
- std::vector< GLfloat > **vertices**
- std::vector< GLfloat > **cylinderVertices**
- std::vector< GLfloat > **cylinderDirections**
- std::vector< GLfloat > **cylinderColors**
- GLuint **allModelDataDisplayList** = 0
- GLuint **VBOColorsID** = 0
- GLuint **VBOTexCoordsID** = 0
- GLuint **VBOVerticesAndRadiiID** = 0
- GLuint **allModelCylinderDataDisplayList** = 0
- GLuint **cylinderVBOColorsID** = 0
- GLuint **cylinderVBODirectionsID** = 0
- GLuint **cylinderVBOVerticesAndHeightsID** = 0
- std::vector< GLfloat > **glCachedColors**
- std::vector< GLfloat > **glCachedTexCoords**
- std::vector< GLfloat > **glCachedPointsAndRadii**
- GLsizei **aBuffer3DSize** = 16
- GLuint **aBuffer3DCounterTextureID** = 0
- GLuint **aBuffer3DTextureID** = 0
- GLuint **aBuffer3DCounterBufferID** = 0
- GLuint **aBuffer3DBufferID** = 0
- GLuint64EXT **aBuffer3DLinkedListSize** = 1 << 21
- GLsizei **aBuffer3DinkedListMaxSize** = 16
- GLuint **aBuffer3DLinkedListAtomicCounterBufferID** = 0

- GLuint **aBuffer3DLinkedListOffsetTextureID** = 0
- GLuint **aBuffer3DLinkedListTextureID** = 0
- GLuint **aBuffer3DLinkedListOffsetBufferID** = 0
- GLuint **aBuffer3DLinkedListBufferID** = 0
- bool **oitAutoManageABuffer3D** = 1
- bool **oitSortFragments** = false
- bool **oitResolveTranslucency** = false
- float **oitTranslucencyAbsorptionCoefficient** = 10.0f
- float **opacity** = 0.0f
- bool **oitUseDeferredShading** = false
- bool **oitCaptureABuffer3DBufferToFile** = false
- bool **reInitOITShaders** = false
- GLfloat **allLightsPositions** [NUMBER_OF_LIGHTS][3] = { { 0.0f } }
- int **useFog** = 0
- int **useTexturing** = 0
- int **useSphericalMapping** = 0
- int **useEnvironmentMapping** = 0
- int **usePositionalLights** = 0
- int **useOrenNayarDiffuseModel** = 0
- int **useColoredObjects** = 1
- int **useSpecularLighting** = 1
- int **useBackFaceCulling** = 0
- int **previousAvailableMemory** = 0
- bool **reInitModelDataForRendering** = true
- bool **reInitModelCylinderDataForRendering** = true
- int **printGLMemoryInfoOnSecondUpdate** = 2
- bool **isNavigating** = false
- float **sinLightAngle** = 0.0f
- float **cosLightAngle** = 0.0f
- GLuint **dummyVao** = 0

Additional Inherited Members

5.119.1 Detailed Description

ParticleRenderingModelLoaderTest is the 5th set of OpenGL rendering tests.

Author

Thanos Theo, 2009-2018

Version

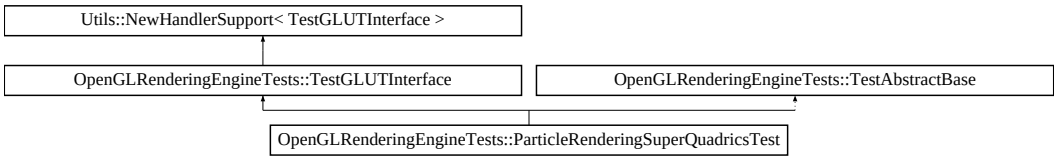
14.0.0.0

5.120 OpenGLRenderingEngineTests::ParticleRenderingSuperQuadricsTest Class Reference

ParticleRenderingSuperQuadricsTest is the 4th set of OpenGL rendering tests.

```
#include <ParticleRenderingSuperQuadricsTest.h>
```

Inheritance diagram for OpenGLRenderingEngineTests::ParticleRenderingSuperQuadricsTest:



Classes

- class OpenGLShaderClearABuffer3D
- class OpenGLShaderDisplayABuffer3D
- class OpenGLShaderFXAA_Antialias
- class OpenGLShaderGBufferEffects
- class OpenGLShaderGBufferEffectsBlurXY
- class OpenGLShaderGBufferEffectsHSSAO

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
- **ParticleRenderingSuperQuadricsTest** (int screenWidth, int screenHeight, const std::string &textureFile↩ Name, const std::string &modelFileName, const std::string &modelLoaderDescriptorFileName, bool multi-sample) noexcept
- **ParticleRenderingSuperQuadricsTest** (const ParticleRenderingSuperQuadricsTest &)=delete
- **ParticleRenderingSuperQuadricsTest** (ParticleRenderingSuperQuadricsTest &&)=delete
- ParticleRenderingSuperQuadricsTest & **operator=** (const ParticleRenderingSuperQuadricsTest &)=delete
- ParticleRenderingSuperQuadricsTest & **operator=** (ParticleRenderingSuperQuadricsTest &&)=delete

Private Types

- enum **AllCachedRenderingTests** : std::size_t {
  USE_NO_CACHING_METHOD_1 = 0, **USE_NO_CACHING_METHOD_2** = 1, **USE_VERTEX_ARRAYS** = 2, **USE_DISPLAY_LISTS** = 3,
  USE_VBOS = 4 }
- enum **AllLightingModels** : std::size_t {
  SIMPLE_LIGHTING = 0, **PHONG** = 1, **BLINN_PHONG** = 2, **GAUSSIAN** = 3,
  TOON = 4, **GOOCH** = 5 }
- enum **ImpostorShaderUsage** : std::size_t {
  POINT_SPHERE = 0, **QUAD_SPHERE_VS_FS** = 1, **QUAD_SPHERE_VS_GS_FS** = 2, **QUAD_SPHERE_VS_TESS_LOD_FS** = 3,
  QUAD_SPHERE_VS_TESS_QUAD_FS = 4, **CYLINDER_VS_GS_FS** = 5 }
- enum **ImpostorWireframeUsage** : std::size_t { **FULL_IMPOSTOR_WIREFRAME** = 0, **IMPOSTOR_POINTS_ONLY** = 1, **IMPOSTOR_LINES_ONLY** = 2, **CYLINDER_EXPLOSION** = 3 }
- enum **AllOITModes** : std::size_t {
  NO_OIT = 0, **A_BUFFER_3D_USE_TEXTURES** = 1, **A_BUFFER_3D_USE_BUFFERS** = 2, **A_BUFFER_3D_USE_LINKED_LIST_TEXTURES** = 3,
  A_BUFFER_3D_USE_LINKED_LIST_BUFFERS = 4 }
- enum **GBUfferEffectTypes** : std::size_t {
  NO_GBUFFER_EFFECT = 0, **DEPTH_OF_FIELD** = 1, **EDGE_ENHANCEMENT** = 2, **HSSAO** = 3,
  HSSAO_PLUS = 4 }

Private Member Functions

- void **prepareFog** () const
- void **prepareLighting** ()
- void **updateLighting** ()
- void **preparePointSphereShaders** ()
- void **prepareABuffer3DShaders** ()
- void **prepareGBufferEffectsShaders** ()
- void **prepareFXAA_AntialiasShaders** ()
- void **prepareTexture** ()
- void **prepareEnvironmentMappingFBO** ()
- void **initEnviromentMappingFBO** () const
- void **drawEnvironmentMappingFBO** ()
- void **prepareGBufferEffectsFBOs** ()
- void **prepareHSSAOData** ()
- void **initGBufferEffectsFBOs** () const
- void **processGBufferEffectsFBOs** ()
- void **drawGBufferEffectsFBOs** ()
- void **prepareFXAA_AntialiasFBO** ()
- void **initFXAA_AntialiasFBO** () const
- void **drawFXAA_AntialiasFBO** ()
- void **initABuffer3D** ()
- bool **autoManageABuffer3D** ()
- void **deleteABuffer3D** ()
- void **writeABuffer3DBufferToFile** (GLbitfield barriers) const
- void **initSuperQuadricsShapes** ()
- void **createPointVerticesArray** ()
- void **createPointColorsArray** ()
- void **createPointCylinderVerticesAndDirectionsArray** ()
- void **createPointCylinderColorsArray** ()
- void **createQuadColorsArray** ()
- void **createQuadTextureCoordsArray** ()

- void **createQuadVerticesArray** ()
- void **prepareVBOs** ()
- void **deleteVBOs** ()
- void **prepareCylinderVBOs** ()
- void **deleteCylinderVBOs** ()
- void **renderNoCaching** (bool skipReInitModelDataForRendering)
- void **renderVertexArrays** ()
- void **renderDisplayLists** ()
- void **renderVBOs** ()
- void **renderCylinderNoCaching** (bool skipReInitModelDataForRendering)
- void **renderCylinderVertexArrays** ()
- void **renderCylinderDisplayLists** ()
- void **renderCylinderVBOs** ()
- void **clearScreen** (bool useGBuffer=false) const
- void **disableLights** (bool disableAllLightSources=true) const
- void **enableLights** (bool enableAllLightSources=true)
- void **renderParticleScene** (bool environmentMappingRenderPass, bool useGBuffer=false)
- void **renderParticleGeometry** (bool environmentMappingRenderPass, bool useGBuffer)
- bool **isQuadVSFS** () const
- bool **isQuadVSTESSFS** () const
- void **drawString** (const char ∗str, int x, int y, const float color[4], void ∗font) const
- void **drawString3D** (const char ∗str, float position[3], const float color[4], void ∗font) const
- void **showInfo** ()
- int **howMuchCurrentlyAvailableVRAMMemory** () const
- std::string **howMuchCurrentlyAvailableVRAMMemoryString** () const
- void **showFPS** ()

Private Attributes

- AllCachedRenderingTests **currentCachedRenderingTest** = USE_DISPLAY_LISTS
- AllLightingModels **currentLightingModel** = BLINN_PHONG
- ImpostorShaderUsage **currentImpostorShader** = QUAD_SPHERE_VS_GS_FS
- ImpostorWireframeUsage **currentImpostorWireframe** = FULL_IMPOSTOR_WIREFRAME
- AllOITModes **currentOITMode** = NO_OIT
- GBUfferEffectTypes **currentGBUfferEffectType** = NO_GBUFFER_EFFECT
- OpenGLRenderingEngineTests::ParticleRenderingSuperQuadricsTest::OpenGLShaderClearABuffer3D ∗ **openGLShaderClearABuffer3D** = nullptr
- OpenGLRenderingEngineTests::ParticleRenderingSuperQuadricsTest::OpenGLShaderDisplayABuffer3D ∗ **openGLShaderDisplayABuffer3D** = nullptr
- OpenGLRenderingEngineTests::ParticleRenderingSuperQuadricsTest::OpenGLShaderGBufferEffectsBlurXY ∗ **openGLShaderGBufferEffectsBlurXY** = nullptr
- OpenGLRenderingEngineTests::ParticleRenderingSuperQuadricsTest::OpenGLShaderGBufferEffectsHSSAO ∗ **openGLShaderGBufferEffectsHSSAO** = nullptr
- OpenGLRenderingEngineTests::ParticleRenderingSuperQuadricsTest::OpenGLShaderGBufferEffects ∗ **openGLShaderGBufferEffects** = nullptr
- OpenGLRenderingEngineTests::ParticleRenderingSuperQuadricsTest::OpenGLShaderFXAA_Antialias ∗ **openGLShaderFXAA_Antialias** = nullptr
- OpenGLRenderingEngine::OpenGLModelAmbientLight ∗ **openGLModelAmbientLight** = nullptr
- OpenGLRenderingEngine::OpenGLLight ∗ **openGLLights** [NUMBER_OF_LIGHTS] = { nullptr }
- OpenGLRenderingEngine::OpenGLMaterial ∗ **openGLMaterial** = nullptr
- OpenGLRenderingEngine::OpenGLShaderImpostorModels ∗ **openGLShaderPointSphereModels** = nullptr
- OpenGLRenderingEngine::OpenGLShaderImpostorModels ∗ **openGLShaderPointSphereModelsQuadVFFS** = nullptr

- OpenGLRenderingEngine::OpenGLShaderImpostorModels * **openGLShaderPointSphereModelsQuad↩VFGSFS** = nullptr
- OpenGLRenderingEngine::OpenGLShaderImpostorModels * **openGLShaderPointSphereModelsQuad↩VFTESSLODFS** = nullptr
- OpenGLRenderingEngine::OpenGLShaderImpostorModels * **openGLShaderPointSphereModelsQuad↩VFTESSQuadFS** = nullptr
- OpenGLRenderingEngine::OpenGLShaderImpostorModels * **openGLShaderCylinderVFGSFS** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectForEnvironment↩Mapping** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectForGBufferPass1** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectRenderPass2** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectRenderPass3** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectForFXAA_Antialias** = nullptr
- OpenGLRenderingEngine::OpenGLILTexture * **openGLILTexture** = nullptr
- float **radiusFactor** = 1.0f
- int **usePointAntialias** = 0
- int **usePerspectiveCorrection** = 0
- int **previousUsePerspectiveCorrection** = 0
- int **depthOfFieldBlurXYNumberOfSamplesRange** = 8
- float **depthOfFieldRange** = 10.0f
- float **depthOfFieldZFocus** = 0.5f
- GLuint **rotationTextureID** = 0
- std::vector< float > **rotationTextureData**
- std::vector< float > **depthSamplesData**
- bool **useTransparency** = false
- bool **reInitColorArrays** = false
- std::vector< GLdouble > **superQuadricVertices**
- std::vector< GLfloat > **vertices**
- std::vector< GLfloat > **colors**
- std::vector< GLfloat > **cylinderVertices**
- std::vector< GLfloat > **cylinderDirections**
- std::vector< GLfloat > **cylinderColors**
- GLuint **allSuperQuadricsDataDisplayList** = 0
- GLuint **VBOColorsID** = 0
- GLuint **VBOTexCoordsID** = 0
- GLuint **VBOVerticesAndRadiiID** = 0
- GLuint **allSuperQuadricsCylinderDataDisplayList** = 0
- GLuint **cylinderVBOColorsID** = 0
- GLuint **cylinderVBODirectionsID** = 0
- GLuint **cylinderVBOVerticesAndHeightsID** = 0
- std::vector< GLfloat > **glCachedColors**
- std::vector< GLfloat > **glCachedTexCoords**
- std::vector< GLfloat > **glCachedPointsAndRadii**
- GLsizei **aBuffer3DSize** = 16
- GLuint **aBuffer3DCounterTextureID** = 0
- GLuint **aBuffer3DTextureID** = 0
- GLuint **aBuffer3DCounterBufferID** = 0
- GLuint **aBuffer3DBufferID** = 0
- GLuint64EXT **aBuffer3DLinkedListSize** = 1 << 21
- GLsizei **aBuffer3DinkedListMaxSize** = 16
- GLuint **aBuffer3DLinkedListAtomicCounterBufferID** = 0

- GLuint **aBuffer3DLinkedListOffsetTextureID** = 0
- GLuint **aBuffer3DLinkedListTextureID** = 0
- GLuint **aBuffer3DLinkedListOffsetBufferID** = 0
- GLuint **aBuffer3DLinkedListBufferID** = 0
- bool **oitAutoManageABuffer3D** = 1
- bool **oitSortFragments** = false
- bool **oitResolveTranslucency** = false
- float **oitTranslucencyAbsorptionCoefficient** = 10.0f
- float **opacity** = 0.0f
- bool **oitUseDeferredShading** = false
- bool **oitCaptureABuffer3DBufferToFile** = false
- bool **reInitOITShaders** = false
- GLfloat **allLightsPositions** [NUMBER_OF_LIGHTS][3] = { { 0.0f } }
- int **useFog** = 0
- int **useTexturing** = 0
- int **useSphericalMapping** = 0
- int **useEnvironmentMapping** = 0
- int **usePositionalLights** = 0
- int **useOrenNayarDiffuseModel** = 0
- int **useColoredObjects** = 1
- int **useSpecularLighting** = 1
- int **useBackFaceCulling** = 0
- int **useGeometryExplosion** = 0
- int **previousAvailableMemory** = 0
- int **slices** = 0
- int **segments** = 0
- int **geometryRepetition** = 0
- bool **reInitSuperQuadricsDataForRendering** = true
- bool **reInitSuperQuadricsCylinderDataForRendering** = true
- int **printGLMemoryInfoOnSecondUpdate** = 2
- bool **isNavigating** = false
- float **sinLightAngle** = 0.0f
- float **cosLightAngle** = 0.0f
- GLuint **dummyVao** = 0

Additional Inherited Members

5.120.1 Detailed Description

ParticleRenderingSuperQuadricsTest is the 4th set of OpenGL rendering tests.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.121 UtilsCUDA::PinnedDeleter< T > Struct Template Reference

Custom deleter class for (pinned) host memory.

`#include <CUDAMemoryHandlers.h>`

Public Member Functions

- **PinnedDeleter** (bool useDeleter=true) noexcept
- void **operator()** (T *ptr) noexcept

Public Attributes

- bool **useDeleter_** = false

5.121.1 Detailed Description

template<typename T>
struct UtilsCUDA::PinnedDeleter< T >

Custom deleter class for (pinned) host memory.

Author

David Lenz, Thanos Theo, 2018

Version

14.0.0.0

5.122 Utils::AccurateTimers::ProfileCPUTimer Class Reference

ProfileCPUTimer profiling helper class using RAII.

`#include <AccurateTimers.h>`

Public Member Functions

- **ProfileCPUTimer** (const std::string &message="")
- double **getElapsedTimeInMilliSecs** ()
- **ProfileCPUTimer** (const ProfileCPUTimer &)=delete
- **ProfileCPUTimer** (ProfileCPUTimer &&)=delete
- ProfileCPUTimer & **operator=** (const ProfileCPUTimer &)=delete
- ProfileCPUTimer & **operator=** (ProfileCPUTimer &&)=delete

Private Attributes

- std::string **message_**
- AccurateCPUTimer **timer_**

5.122.1 Detailed Description

ProfileCPUTimer profiling helper class using RAII.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.123 UtilsCUDA::ProfileGPUTimer Class Reference

ProfileGPUTimer profiling helper class using RAII.

`#include <CUDAEventTimer.h>`

Public Member Functions

- **ProfileGPUTimer** (const std::string &message="", int device=0, const cudaStream_t &cudaStream=nullptr)
- double **getElapsedTimeInMilliSecs** ()
- **ProfileGPUTimer** (const ProfileGPUTimer &)=delete
- **ProfileGPUTimer** (ProfileGPUTimer &&)=delete
- ProfileGPUTimer & **operator=** (const ProfileGPUTimer &)=delete
- ProfileGPUTimer & **operator=** (ProfileGPUTimer &&)=delete

Private Attributes

- std::string **message_**
- CUDAEventTimer **timer_**

5.123.1 Detailed Description

ProfileGPUTimer profiling helper class using RAII.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.124 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

5.124.1 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

5.125 UtilsCUDA::RawDeviceMemory< T > Class Template Reference

Helper class for the Span class below.

`#include <CUDAMemoryWrappers.h>`

Public Types

- using **NonConstType** = typename std::remove_const< T >::type
- using **ConstType** = const NonConstType

Public Member Functions

- __forceinline__ __host__ __device__ **operator T*** () const
- __forceinline__ __host__ __device__ T * **operator->** () const
- __forceinline__ __host__ __device__ T & **operator[ ]** (std::size_t index) const
- __forceinline__ __host__ __device__ T * data () const

  Get the raw pointer from the wrapper without any checks.
- __forceinline__ __host__ __device__ **RawDeviceMemory** (T *ptr)
- __forceinline__ __host__ __device__ **RawDeviceMemory** (const RawDeviceMemory< NonConstType > &other) noexcept
- __forceinline__ __host__ __device__ **RawDeviceMemory** (const RawDeviceMemory< ConstType > &other) noexcept
- **RawDeviceMemory** (RawDeviceMemory &&)=default
- RawDeviceMemory & **operator=** (const RawDeviceMemory &)=default
- RawDeviceMemory & **operator=** (RawDeviceMemory &&)=default

Private Attributes

- T * **ptr_** = nullptr

5.125.1 Detailed Description

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

Helper class for the Span class below.

Helper class to ensure that a pointer is referencing a valid device memory area. Its main purpose is obtaining type-safety at compile time when mixing host and device memory:

- it prevents users from accidentally accessing the memory on the host side
- allows function arguments (or struct members etc) to state the intent whether it expects device memory or host memory

Template Parameters

T	underlying type of the array

Author

David Lenz, 2019

Version

14.0.0.0

5.126 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**

5.126.1 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

5.127 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

5.127.1 Detailed Description

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

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.128 UtilsCUDA::Span< T, SIZE > Class Template Reference

Helper class implementing the "span" paradigm from the CppCoreGuidelines applied to device arrays.

`#include <CUDAMemoryWrappers.h>`

Public Types

- using **NonConstType** = typename std::remove_const< T >::type
- using **ConstType** = const NonConstType

Public Member Functions

- __forceinline__ __host__ __device__ bool **empty** () const noexcept
- __forceinline__ __host__ __device__ std::size_t **size** () const noexcept
- __forceinline__ __host__ __device__ T * **data** () const noexcept
- __forceinline__ __host__ __device__ **operator T*** () const
- __forceinline__ __host__ __device__ Span< T > subSpan (std::size_t startIndex, std::size_t size) const

  Create a subspan, i.e.
- template<std::size_t SUBSPAN_SIZE>
  __forceinline__ __host__ __device__ Span< T, SUBSPAN_SIZE > subSpan (std::size_t startIndex) const

  Same functionality.
- void resize (std::size_t newSize)

  Resize the span.
- __forceinline__ __host__ __device__ **Span** (T *ptr, std::size_t count)
- __forceinline__ __host__ __device__ **Span** (T *ptr)
- __forceinline__ __host__ __device__ **Span** (const Span< NonConstType, SIZE > &other) noexcept
- __forceinline__ __host__ __device__ **Span** (const Span< ConstType, SIZE > &other) noexcept
- template<std::size_t OTHER_SIZE>
  __forceinline__ __host__ __device__ **Span** (const Span< NonConstType, OTHER_SIZE > &other) noexcept
- template<std::size_t OTHER_SIZE>
  __forceinline__ __host__ __device__ **Span** (const Span< ConstType, OTHER_SIZE > &other) noexcept
- template<std::size_t OTHER_SIZE>
  __host__ **Span** (const std::array< NonConstType, OTHER_SIZE > &other) noexcept
- template<std::size_t OTHER_SIZE>
  __host__ **Span** (const std::array< ConstType, OTHER_SIZE > &other) noexcept
- **Span** (Span &&)=default
- Span & **operator=** (const Span &)=default
- Span & **operator=** (Span &&)=default

- __forceinline__ __host__ __device__ T * begin () const

  Enable ranged base for loops.
- __forceinline__ __host__ __device__ T * **end** () const

- __forceinline__ __host__ __device__ T & operator[ ] (std::size_t index) const

  Accessor functions for single elements.
- __forceinline__ __host__ __device__ T & **at** (std::size_t index) const
- __forceinline__ __host__ __device__ T & **operator()** (std::size_t index) const

Private Attributes

- SpanStorage< T, SIZE > **storage_** {}

5.128.1 Detailed Description

template<typename T, std::size_t SIZE = DYNAMIC_SIZE>
class UtilsCUDA::Span< T, SIZE >

Helper class implementing the "span" paradigm from the CppCoreGuidelines applied to device arrays.

This class provides a view on an array hosted in device memory. It serves the following purposes:

- simplifies repeated function calls with void func(T*, size_t count) to void func(Span span)
- allows automatic error and bounds checking for debug or other instrumented builds

Note

As it is to be used for device memory, all the stl iterator "Container" requirements are not fulfilled for now

Template Parameters

T	underlying type of the array
SIZE	the size of the underlying memory (in number of elements) if known at compile time or DYNAMIC_SIZE to allow sizes at runtime

Author

David Lenz, 2019

Version

14.0.0.0

5.128.2 Member Function Documentation

5.128.2.1 resize()

```
template<typename T, std::size_t SIZE = DYNAMIC_SIZE>
void UtilsCUDA::Span< T, SIZE >::resize (
            std::size_t newSize )  [inline]
```

Resize the span.

Note

this will not alter the underlying memory, but only alter the view on that memory
only do this, if you know that the underlying memory has at least this size.

5.128.2.2 subSpan() [1/2]

```
template<typename T, std::size_t SIZE = DYNAMIC_SIZE>
__forceinline__ __host__ __device__ Span<T> UtilsCUDA::Span< T, SIZE >::subSpan (
            std::size_t startIndex,
            std::size_t size ) const  [inline]
```

Create a subspan, i.e.

a view on a subset of the memory pointed by the current span The returned span will point to the [ptr + startIndex -> ptr + startIndex + size] The function assumes, that this memory region is overlapping the valid memory region for this span

5.128.2.3 subSpan() [2/2]

```
template<typename T, std::size_t SIZE = DYNAMIC_SIZE>
template<std::size_t SUBSPAN_SIZE>
__forceinline__ __host__ __device__ Span<T, SUBSPAN_SIZE> UtilsCUDA::Span< T, SIZE >::subSpan
(
            std::size_t startIndex ) const  [inline]
```

Same functionality.

See also

subSpan but returning a fixed size span

Template Parameters

SUBSPAN_SIZE	

Parameters

startIndex	

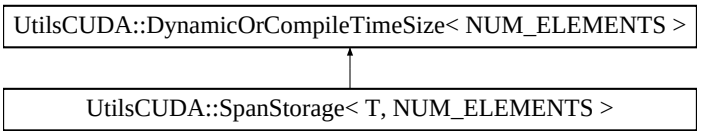
Returns

5.129 UtilsCUDA::SpanStorage< T, NUM_ELEMENTS > Class Template Reference

Helper class to store a pointer and an associated size with it.

`#include <CUDAMemoryWrappers.h>`

Inheritance diagram for UtilsCUDA::SpanStorage< T, NUM_ELEMENTS >:



Public Member Functions

- __forceinline__ __host__ __device__ **SpanStorage** (T *data, std::size_t size)
- __forceinline__ __host__ __device__ T * **data** () const

Private Attributes

- T * **rawPtr_** = nullptr

5.129.1 Detailed Description

template<typename T, std::size_t NUM_ELEMENTS = DYNAMIC_SIZE>
class UtilsCUDA::SpanStorage< T, NUM_ELEMENTS >

Helper class to store a pointer and an associated size with it.

In case of compile-time sizes, the size of this helper class is only the size of the pointer. storing of the size does not need any data

See also

DynamicOrCompileTimeSize

Template Parameters

T	datatype of the pointer thats supposed to be stored
NUM_ELEMENTS	compile time size or DYNAMIC_SIZE for runtime sized arrays

Author

David Lenz, 2019

Version

14.0.0.0

5.130 Utils::UtilityFunctions::StdAuxiliaryFunctions Struct Reference

The StdAuxiliaryFunctions struct 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<std::size_t N, typename T >
  static void insertionSort (T *__restrict arrayData)

  Sort an array using insertion sort with a constant small size of N.

5.130.1 Detailed Description

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

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.130.2 Member Function Documentation

5.130.2.1 insertionSort()

```
template<std::size_t N, typename T >
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.

5.131 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 void writeTextFile (const std::string &fullpathWithFileName, const std::list< std::string > &textToWrite, std::ios_base::openmode mode=std::ios::out)

  Writes a text file with a given list of texts.
- 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 arraySize)

  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 &archiveName, void *&data, std::size_t &dataSize)

  zipExtractArchiveFileToHeap() reads a single file from an archive into a heap block.

5.131.1 Detailed Description

The StdReadWriteFileFunctions class provides additional i/o functionality.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.131.2 Member Function Documentation

5.131.2.1 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.

5.131.2.2 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.

5.132 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... Args>
  static std::string printfToString (const char *format, const Args... args)

  String manipulation auxiliary function (Args...
- 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** = 2048
- static constexpr std::size_t **CHARACTER_BUFFER_SIZE** = 64

5.132.1 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

5.132.2 Member Function Documentation

5.132.2.1 printfToString()

```
template<typename...  Args>
static std::string Utils::UtilityFunctions::StringAuxiliaryFunctions::printfToString (
            const char * format,
            const Args...  args )  [inline], [static]
```

String manipulation auxiliary function (Args...

'as a variadic template' encapsulation for printf version).

5.132.2.2 tokenize()

```
template<typename Container >
static Container Utils::UtilityFunctions::StringAuxiliaryFunctions::tokenize (
            const std::string & str,
            const std::string & delimiters = " " )  [inline], [static]
```

String manipulation auxiliary function.

Note: the Container has to support the 'insert()' function (std::vector, ordered & unordered std::map/std::set, std↩
::dequeue, but NOT std::queue).

5.133 SuperQuadrics::SuperQuadricSettings Struct Reference

SuperQuadricSettings is the class that acts as a placeholder of the SuperQuadricShape variables needed for the parametric polar coordinate equations.

```
#include <SuperQuadricSettings.h>
```

Public Types

- enum SuperQuadricShapeTypes : std::size_t { **ELLIPSOID** = 0, **HYPERBOLOID_ONE_SHEET** = 1, **HYP↩ ERBOLOID_TWO_SHEETS** = 2, **TOROID** = 3 }

  SuperQuadricShapeTypes enumeration class which holds information for the shape type of a SuperQuadricShape.

Public Member Functions

- **SuperQuadricSettings** (const SuperQuadricSettings &)=default
- **SuperQuadricSettings** (SuperQuadricSettings &&)=default
- SuperQuadricSettings & **operator=** (const SuperQuadricSettings &)=default
- SuperQuadricSettings & **operator=** (SuperQuadricSettings &&)=default

Public Attributes

- SuperQuadricShapeTypes **superQuadricShapeType** = ELLIPSOID
- double a1 = 1.0

  Scaling factor for x.
- double a2 = 1.0

  Scaling factor for y.
- double a3 = 1.0

  Scaling factor for z.
- double alpha = 2.0

  For generating toroids.
- double n = 1.0

  North-South Roundness factor.
- double e = 1.0

  East-West Squareness factor.
- double u1 = -Utils::PI_DBL / 2.0

  Initial U value.
- double u2 = Utils::PI_DBL / 2.0

  Final U value.
- double v1 = -Utils::PI_DBL

  Initial V value.
- double v2 = Utils::PI_DBL

  Final V value.
- std::size_t uSegments = 8

  Number of segments for U.
- std::size_t vSegments = 8

  Number of segments for V.
- double s1 = 0.0

  Initial S value.
- double s2 = 1.0

  Final S value.
- double t1 = 0.0

  Initial T value.
- double t2 = 1.0

  Final T value.

5.133.1 Detailed Description

SuperQuadricSettings is the class that acts as a placeholder of the SuperQuadricShape variables needed for the parametric polar coordinate equations.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.133.2 Member Data Documentation

5.133.2.1 alpha

```
double SuperQuadrics::SuperQuadricSettings::alpha = 2.0
```

For generating toroids.

This is the inner radius.

5.134 SuperQuadrics::SuperQuadricShape Class Reference

A SuperQuadricShape is the class that creates SuperQuadric-based shapes using parametric polar coordinate equations.

```
#include <SuperQuadricShape.h>
```

Public Member Functions

- void **createGeometry** (double dU, double dV, double dS, double dT, std::size_t position)
- double **insideOut** (double x, double y, double z) const
- **SuperQuadricShape** (const SuperQuadricSettings &superQuadricSettings, std::size_t superQuadric↩ ShapeType, const ModelSettings &modelSettings, double ∗vertices, double ∗normals=nullptr, double ∗tex↩ Coords=nullptr) noexcept
- **SuperQuadricShape** (const SuperQuadricShape &)=delete
- **SuperQuadricShape** (SuperQuadricShape &&)=delete
- SuperQuadricShape & **operator=** (const SuperQuadricShape &)=delete
- SuperQuadricShape & **operator=** (SuperQuadricShape &&)=delete

Private Member Functions

- void **sqEllipsoid** (double a1, double a2, double a3, double u, double v, double n, double e, std::size_t verticesIndex) const
- void **sqHyperboloidOneSheet** (double a1, double a2, double a3, double u, double v, double n, double e, std::size_t verticesIndex) const
- void **sqHyperboloidTwoSheets** (double a1, double a2, double a3, double u, double v, double n, double e, std::size_t verticesIndex) const
- void **sqToroid** (double a1, double a2, double a3, double u, double v, double n, double e, double alpha, std::size_t verticesIndex) const
- void **createSuperQuadric** (double u, double v, std::size_t verticesIndex, double s, double t, std::size_t texCoordsIndex) const
- void **applyTriangleGeometryShrinking** (std::size_t verticesTriangle1Index, std::size_t verticesTriangle2Index, std::size_t verticesTriangle3Index)
- double **sqEllipsoidInsideOut** (double x, double y, double z) const
- double **sqHyperboloidOneSheetInsideOut** (double x, double y, double z) const
- double **sqHyperboloidTwoSheetsInsideOut** (double x, double y, double z) const
- double **sqToroidInsideOut** (double x, double y, double z) const

Private Attributes

- SuperQuadricSettings superQuadricSettings_

  Variable used for choosing between the SuperQuadric shape types.
- std::size_t superQuadricShapeType_ = 0

  The SuperQuadric reference stores all SuperQuadric related variables.
- ModelSettings modelSettings_

  The SuperQuadric reference stores all SuperQuadric related variables.
- double * vertices_ = nullptr

  The vertices memory pointer.
- double * normals_ = nullptr

  The normals memory pointer.
- double * texCoords_ = nullptr

  The texCoords memory pointer.
- Utils::Randomizers::RandomRNGWELL512 **random_**

5.134.1 Detailed Description

A SuperQuadricShape is the class that creates SuperQuadric-based shapes using parametric polar coordinate equations.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.135 SuperQuadrics::SuperQuadricShapesProducer Struct Reference

Many examples using the SuperQuadricShapesProducer for producing various SuperQuadric-based shapes.

`#include <SuperQuadricShapesProducer.h>`

Public Types

- enum **SuperQuadricShapeObjects** : std::size_t {
  SPHERE = 0, **CYLINDER** = 1, **STAR** = 2, **DOUBLE_PYRAMID** = 3,
  TORUS = 4, **PINEAPPLE_SLICE** = 5, **PILLOW** = 6, **SQUARE_TORUS** = 7,
  PINCHED_TORUS = 8, **ROUND_CUBE** = 9, **HYPERBOLOID_ONE_SHEET** = 10, **HYPERBOLOID_TW↩O_SHEETS** = 11 }

Public Member Functions

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

Static Public Member Functions

- static void createSuperQuadricShape (std::size_t slices, std::size_t segments, double shrinkGeometry↩Factor, bool usingGeometryExplosionFactor, std::size_t startingIndex, SuperQuadricShapesProducer::↩SuperQuadricShapeObjects superQuadricShapeObject, std::vector< double > &vertices)

  Creates a given Super Quadric shape with vertices only.
- static void createSuperQuadricShape (std::size_t slices, std::size_t segments, double shrinkGeometry↩Factor, bool usingGeometryExplosionFactor, std::size_t startingIndex, SuperQuadricShapesProducer::↩SuperQuadricShapeObjects superQuadricShapeObject, std::vector< double > &vertices, std::vector< double > &normals)

  Creates a given Super Quadric shape with vertices & normals.
- static void createSuperQuadricShape (std::size_t slices, std::size_t segments, double shrinkGeometry↩Factor, bool usingGeometryExplosionFactor, std::size_t startingIndex, SuperQuadricShapesProducer::↩SuperQuadricShapeObjects superQuadricShapeObject, std::vector< double > &vertices, std::vector< double > &normals, std::vector< double > &texCoords)

  Creates a given Super Quadric shape with vertices, normals & texture coordinates.
- static std::tuple< std::vector< double >, std::vector< std::uint32_t > > calculateVertexIndices (const std↩::vector< double > &vertices)

  Calculates a vector of unique vertices along with its matching vertex indices.

Static Public Attributes

- static constexpr std::size_t **NUMBER_OF_SHAPES** = 12

5.135.1 Detailed Description

Many examples using the SuperQuadricShapesProducer for producing various SuperQuadric-based shapes.

Using threads for N-CP processing.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.135.2 Member Function Documentation

5.135.2.1 calculateVertexIndices()

```
tuple< vector< double >, vector< uint32_t > > SuperQuadricShapesProducer::calculateVertex↩
Indices (
            const std::vector< double > & vertices )  [static]
```

Calculates a vector of unique vertices along with its matching vertex indices.

Returns

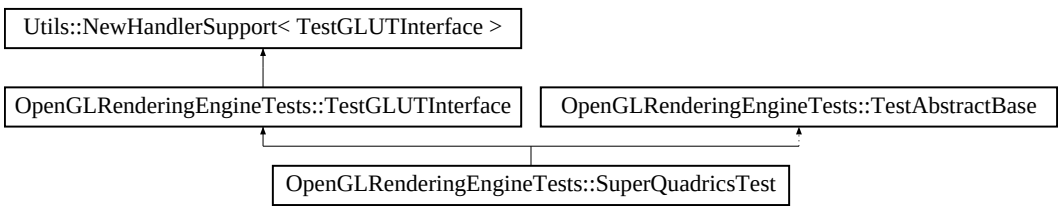
tuple with element one as vertex positions and element two as vertex indices.

5.136 OpenGLRenderingEngineTests::SuperQuadricsTest Class Reference

SuperQuadricsTest is the 2nd set of OpenGL rendering tests.

```
#include <SuperQuadricsTest.h>
```

Inheritance diagram for OpenGLRenderingEngineTests::SuperQuadricsTest:



Classes

- class OpenGLShaderFXAA_Antialias
- class OpenGLShaderGBufferEffects
- class OpenGLShaderGBufferEffectsBlurXY
- class OpenGLShaderGBufferEffectsHSSAO

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
- **SuperQuadricsTest** (int screenWidth, int screenHeight, const std::string &textureFileName, bool multisample) noexcept
- **SuperQuadricsTest** (const SuperQuadricsTest &)=delete
- **SuperQuadricsTest** (SuperQuadricsTest &&)=delete
- SuperQuadricsTest & **operator=** (const SuperQuadricsTest &)=delete
- SuperQuadricsTest & **operator=** (SuperQuadricsTest &&)=delete

Private Types

- enum **AllCachedRenderingTests** : std::size_t {
  USE_NO_CACHING_METHOD_1 = 0, **USE_NO_CACHING_METHOD_2** = 1, **USE_VERTEX_ARRAYS** = 2, **USE_DISPLAY_LISTS** = 3,
  USE_VBOS = 4 }
- enum **AllLightingModels** : std::size_t {
  FIXED_PIPELINE = 0, **PHONG** = 1, **BLINN_PHONG** = 2, **GAUSSIAN** = 3,
  TOON = 4, **GOOCH** = 5 }
- enum **GeometryShaderUsage** : std::size_t { **NO_USE** = 0, **PASS_THRU** = 1, **CREATE_NORMAL_GEO↩METRY** = 2 }
- enum **AllLODModels** : std::size_t { **PN_TRIANGLES** = 0, **PHONG_TESSELLATION** = 1 }
- enum **AllAdaptiveTesselationModes** : std::size_t {
  NO_ADAPTIVE_TESSELLATION = 0, **ADAPTIVE_TESSELLATION_TRIANGLE_POSITION_METRIC** = 1, **ADAPTIVE_TESSELLATION_LIGHT_METRIC** = 2, **ADAPTIVE_TESSELLATION_CONTOUR_METRIC** = 3,
  ADAPTIVE_TESSELLATION_ALL_METRICS = 4 }
- enum **GBUfferEffectTypes** : std::size_t {
  NO_GBUFFER_EFFECT = 0, **DEPTH_OF_FIELD** = 1, **EDGE_ENHANCEMENT** = 2, **HSSAO** = 3,
  HSSAO_PLUS = 4 }

Private Member Functions

- void **prepareFog** () const
- void **prepareLighting** ()
- void **updateLighting** ()
- void **prepareShaders** ()
- void **prepareLODShaders** ()
- void **prepareGBufferEffectsShaders** ()
- void **prepareFXAA_AntialiasShaders** ()
- void **prepareTexture** ()
- void **prepareDepthStencilFBO** ()
- void **initDepthStencilFBO** () const
- void **prepareGBufferEffectsFBOs** ()
- void **prepareHSSAOData** ()
- void **initGBufferEffectsFBOs** () const
- void **processGBufferEffectsFBOs** ()
- void **drawGBufferEffectsFBOs** ()

- void **prepareFXAA_AntialiasFBO** ()
- void **initFXAA_AntialiasFBO** () const
- void **drawFXAA_AntialiasFBO** ()
- void **initSuperQuadricsShapes** ()
- void **prepareVBOs** ()
- void **deleteVBOs** () const
- void **renderNoCaching** (bool skipReInitSuperQuadricsData, bool enableShapeColors, bool useGBuffer)
- void **renderVertexArrays** (bool enableShapeColors, bool useGBuffer)
- void **renderDisplayLists** (bool enableShapeColors, bool useGBuffer)
- void **renderVBOs** (bool enableShapeColors, bool useGBuffer)
- void **clearScreen** (bool useGBuffer=false) const
- void **disableLights** (bool disableAllLightSources=true)
- void **enableLights** (bool enableAllLightSources=true)
- void **renderSuperQuadricsScene** (bool useGBuffer=false)
- void **renderSuperQuadricsGeometry** (bool useGBuffer, bool enableStencilBufferColor=false)
- bool **isUsingLOD** () 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** ()
- int **howMuchCurrentlyAvailableVRAMMemory** () const
- std::string **howMuchCurrentlyAvailableVRAMMemoryString** () const
- void **showFPS** ()

Private Attributes

- AllCachedRenderingTests **currentCachedRenderingTest** = USE_DISPLAY_LISTS
- AllLightingModels **currentLightingModel** = BLINN_PHONG
- GeometryShaderUsage **currentGeometryShader** = NO_USE
- AllLODModels **currentLODModel** = PN_TRIANGLES
- AllAdaptiveTesselationModes **currentAdaptiveTesselationMode** = ADAPTIVE_TESSELLATION_TRIAN↩ GLE_POSITION_METRIC
- GBUfferEffectTypes **currentGBUfferEffectType** = NO_GBUFFER_EFFECT
- OpenGLRenderingEngineTests::SuperQuadricsTest::OpenGLShaderGBufferEffectsBlurXY ∗ **openGL↩ ShaderGBufferEffectsBlurXY** = nullptr
- OpenGLRenderingEngineTests::SuperQuadricsTest::OpenGLShaderGBufferEffectsHSSAO ∗ **openGL↩ ShaderGBufferEffectsHSSAO** = nullptr
- OpenGLRenderingEngineTests::SuperQuadricsTest::OpenGLShaderGBufferEffects ∗ **openGLShaderG↩ BufferEffects** = nullptr
- OpenGLRenderingEngineTests::SuperQuadricsTest::OpenGLShaderFXAA_Antialias ∗ **openGLShaderF↩ XAA_Antialias** = nullptr
- OpenGLRenderingEngine::OpenGLModelAmbientLight ∗ **openGLModelAmbientLight** = nullptr
- OpenGLRenderingEngine::OpenGLLight ∗ **openGLLights** [NUMBER_OF_LIGHTS] = { nullptr }
- OpenGLRenderingEngine::OpenGLMaterial ∗ **openGLMaterial** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingModels ∗ **openGLShaderSurfaceLighting↩ Models** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingModels ∗ **openGLShaderSurfaceLighting↩ ModelsWithGeometry** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingModels ∗ **openGLShaderSurfaceLighting↩ ModelsWithGeometryAndNormals** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingLODModels ∗ **openGLShaderSurface↩ LightingLODModels** = nullptr
- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingLODModels ∗ **openGLShaderSurface↩ LightingLODModelsWithGeometry** = nullptr

- OpenGLRenderingEngine::OpenGLShaderSurfaceLightingLODModels * **openGLShaderSurfaceLightingLODModelsWithGeometryAndNormals** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectForHighQualityOutline** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectForGBufferPass1** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectRenderPass2** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectRenderPass3** = nullptr
- OpenGLRenderingEngine::OpenGLFrameBufferObject * **openGLFrameBufferObjectForFXAA_Antialias** = nullptr
- OpenGLRenderingEngine::OpenGLILTexture * **openGLILTexture** = nullptr
- bool **useLODModels** = false
- float **tessellationAlpha** = 1.0f
- float **tessellationLevel** = 32.0f
- bool **useAdaptiveTessellation** = true
- bool **useTrianglePositionAdaptiveTessellationMetric** = true
- bool **useLightAdaptiveTessellationMetric** = false
- bool **useContourAdaptiveTessellationMetric** = false
- int **depthOfFieldBlurXYNumberOfSamplesRange** = 8
- float **depthOfFieldRange** = 10.0f
- float **depthOfFieldZFocus** = 0.5f
- GLuint **rotationTextureID** = 0
- std::vector< float > **rotationTextureData**
- std::vector< float > **depthSamplesData**
- GLuint **allSuperQuadricsDataDisplayList** = 0
- GLuint **VBOVerticesID** = 0
- GLuint **VBONormalsID** = 0
- GLuint **VBOTexCoordsID** = 0
- std::vector< GLdouble > **vertices**
- std::vector< GLdouble > **normals**
- std::vector< GLdouble > **texCoords**
- GLfloat **allLightsPositions** [NUMBER_OF_LIGHTS][3] = { { 0.0f } }
- int **useFog** = 0
- int **useTexturing** = 0
- int **useSphericalMapping** = 0
- int **useOutline** = 0
- int **useHighQualityOutline** = 0
- int **usePositionalLights** = 0
- int **useOrenNayarDiffuseModel** = 0
- int **useFresnelFactorSchlickApproximationSpecularModel** = 0
- int **useColoredObjects** = 1
- int **useGeometryExplosion** = 0
- int **useGouraudShading** = 1
- int **previousAvailableMemory** = 0
- int **slices** = 0
- int **segments** = 0
- int **geometryRepetition** = 0
- bool **reInitSuperQuadricsDataForRendering** = true
- int **printGLMemoryInfoOnSecondUpdate** = 2
- float **sinLightAngle** = 0.0f
- float **cosLightAngle** = 0.0f

Additional Inherited Members

5.136.1 Detailed Description

SuperQuadricsTest is the 2nd set of OpenGL rendering tests.

Author

Thanos Theo, 2009-2018

Version

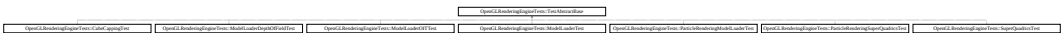
14.0.0.0

5.137 OpenGLRenderingEngineTests::TestAbstractBase Class Reference

TestAbstractBase is the abstract base class for all GLUT tests.

```
#include <TestAbstractBase.h>
```

Inheritance diagram for OpenGLRenderingEngineTests::TestAbstractBase:



Public Member Functions

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

Protected Member Functions

- void **writeScreenshotToFile** (bool useDevIL=true) 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 constexpr std::size_t **GLUT_TEXT_WIDTH** = 9
- static constexpr std::size_t **GLUT_TEXT_HEIGHT** = 15
- static constexpr std::size_t **NUMBER_OF_LIGHTS** = 2
- static constexpr bool **USE_COLOR_MATERIAL** = true
- static constexpr std::size_t **ENVIRONMENT_MAPPING_RATIO_FACTOR** = 1
- static constexpr std::size_t **DEPTH_STENCIL_RATIO_FACTOR** = 1
- static constexpr std::size_t **DEPTH_STENCIL_MULTIPLICATION_FACTOR** = 2
- static constexpr std::size_t **FULLSCREEN_MAPPING_RATIO_FACTOR** = 1
- static constexpr std::size_t **A_BUFFER_3D_MAX_SIZE** = 256
- static constexpr GLuint **ACTIVE_TEXTURE_UNIT_FOR_2D_TEXTURE** = 0
- static constexpr GLuint **ACTIVE_TEXTURE_UNIT_FOR_BLUR_XY_TEXTURE** = 1
- static constexpr GLuint **ACTIVE_TEXTURE_UNIT_FOR_PERLIN_NOISE_3D_TEXTURE** = 2
- static constexpr GLuint **ACTIVE_TEXTURE_UNIT_FOR_A_BUFFER_3D_COUNTER** = 3
- static constexpr GLuint **ACTIVE_TEXTURE_UNIT_FOR_A_BUFFER_3D** = 4

- static constexpr GLuint **ACTIVE_TEXTURE_UNIT_FOR_A_BUFFER_3D_LINKED_LIST_ATOMIC_COU↩ NTER** = 5
- static constexpr GLuint **ACTIVE_TEXTURE_UNIT_FOR_A_BUFFER_3D_LINKED_LIST_OFFSET** = 6
- static constexpr GLuint **ACTIVE_TEXTURE_UNIT_FOR_A_BUFFER_3D_LINKED_LIST** = 7
- static constexpr bool **DEPTH_OF_FIELD_DEBUG_MODE** = false
- static constexpr GLuint **NORMAL_SHADING_BUFFER_ELEMENTS** = 3
- static constexpr GLuint **NORMAL_SHADING_LINKED_LIST_BUFFER_ELEMENTS** = 4
- static constexpr GLuint **DEFERRED_SHADING_BUFFER_ELEMENTS** = 9
- static constexpr GLuint **DEFERRED_SHADING_LINKED_LIST_BUFFER_ELEMENTS** = 10

Private Member Functions

- void **performAllGLInitializations** ()

5.137.1 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

5.138 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:

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

5.138.1 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

5.139 Utils::CPUParallelism::ThreadBarrier Class Reference

This class encapsulates usage of a thread barrier.

`#include <ThreadBarrier.h>`

Public Member Functions

- **ThreadBarrier** (std::size_t threadCount)
- void **wait** ()
- **ThreadBarrier** (const ThreadBarrier &)=delete
- **ThreadBarrier** (ThreadBarrier &&)=delete
- ThreadBarrier & **operator=** (const ThreadBarrier &)=delete
- ThreadBarrier & **operator=** (ThreadBarrier &&)=delete

Private Attributes

- std::size_t **threadCount_** = 0
- pthread_barrier_t **barrier_** {}

5.139.1 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/.
Note: On non-Windows platforms, the pthread_barrier_t struct is being used instead of the generic C++11 implementation.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.140 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_**

5.140.1 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

5.141 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

5.141.1 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

5.142 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(), std::uint64_t affinityMask=AFFINITY_MASK_ALL, std::size_t priority=PRIORITY_NONE)
- template<typename FunctionType >
  void **submit** (const FunctionType &function)
- std::size_t **getNumberOfThreads** () const
- ThreadBarrier & **getBarrier** ()
- void **setAffinity** (std::size_t threadIdx, std::uint64_t affinityMask)
- bool **getAffinity** (std::size_t threadIdx)
- void **setPriority** (std::size_t threadIdx, std::size_t priority)
- std::size_t **getPriority** (std::size_t threadIdx)
- **ThreadPool** (const ThreadPool &)=delete
- **ThreadPool** (ThreadPool &&)=delete
- ThreadPool & **operator=** (const ThreadPool &)=delete
- ThreadPool & **operator=** (ThreadPool &&)=delete

Private Member Functions

- void **checkThreadIdx** (std::size_t threadIdx)
- void **runPendingTask** ()
- void **workerThread** ()

Private Attributes

- std::size_t **numberOfThreads_** = 0
- std::atomic< bool > **done_** = { false }
- ConcurrentBlockingQueue< std::function< void()> > **workQueue_**
- std::unique_ptr< std::thread[ ]> **threads_** = nullptr
- ThreadJoiner **joiner_**
- ThreadBarrier **barrier_**
- std::mutex **mutex_**
- std::condition_variable **conditionVariable_**

5.142.1 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

5.143 Utils::UtilityFunctions::TrigonometricFunctions Struct Reference

The TrigonometricFunctions struct covers essential operations involving angles on the trigonometric circle.

`#include <UtilityFunctions.h>`

Public Member Functions

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

Static Public Member Functions

- 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 , typename AngleRange >
  static T normalizeRadAngle (const T angle, std::enable_if_t< std::is_floating_point< T >::value > *=nullptr)

  Normalizes an angle argument into a range covering the full trigonometric circle.
- template<typename T >
  static T normalizePI (const T angle, std::enable_if_t< std::is_floating_point< T >::value > *=nullptr)

  Normalizes an angle argument into the [-PI, PI) range.
- template<typename T >
  static T normalize2PI (const T angle, std::enable_if_t< std::is_floating_point< T >::value > *=nullptr)

  *Normalizes an angle argument into the [0, 2*PI) range.*
- template<typename T >
  static double getAngularDifference (const T fromAngle, const T targetAngle, std::enable_if_t< std::is_floating_point< T >::value > *=nullptr)

  *Computes the shortest arc length in rads on the trigonometric circle for two given input angles expressed in the [0, 2*PI) range.*

5.143.1 Detailed Description

The TrigonometricFunctions struct covers essential operations involving angles on the trigonometric circle.

Author

Jacobo Cabaleiro, Teodor Cioaca, 2019

Version

1.0.0.0

5.143.2 Member Function Documentation

5.143.2.1 normalize2PI()

```
template<typename T >
static T Utils::UtilityFunctions::TrigonometricFunctions::normalize2PI (
            const T angle,
            std::enable_if_t< std::is_floating_point< T >::value > *  = nullptr )  [inline],
[static]
```

Normalizes an angle argument into the [0, 2*PI) range.

Parameters

angle	in radians

Returns

the corresponding normalized angle

5.143.2.2 normalizePI()

```
template<typename T >
static T Utils::UtilityFunctions::TrigonometricFunctions::normalizePI (
            const T angle,
            std::enable_if_t< std::is_floating_point< T >::value > *  = nullptr )  [inline],
[static]
```

Normalizes an angle argument into the [-PI, PI) range.

Parameters

angle	in radians

Returns

the corresponding normalized angle

5.143.2.3 normalizeRadAngle()

```
template<typename T , typename AngleRange >
static T Utils::UtilityFunctions::TrigonometricFunctions::normalizeRadAngle (
            const T angle,
            std::enable_if_t< std::is_floating_point< T >::value > *  = nullptr )  [inline],
[static]
```

Normalizes an angle argument into a range covering the full trigonometric circle.

Template Parameters

T	input argument data type
AngleRange	structure type exposing the boundaries of the reference angular range

Parameters

angle	the input angular argument

Returns

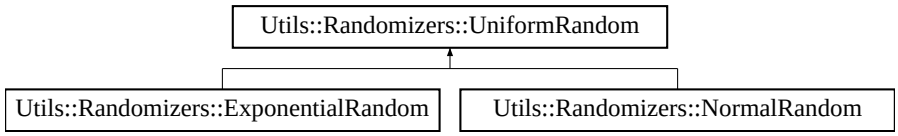
the corresponding normalized angle

5.144 Utils::Randomizers::UniformRandom Class Reference

The UniformRandom class provides a uniform random number generator.

`#include <Randomizers.h>`

Inheritance diagram for Utils::Randomizers::UniformRandom:



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_**

5.144.1 Detailed Description

The UniformRandom class provides a uniform random number generator.

Author

Thanos Theo, 2009-2018

Version

14.0.0.0

5.145 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 >:

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 error=getDefaultError())
- static bool **checkRelativeError** (T a, T b, T relativeError=getDefaultError())
- static bool **checkComplexAbsoluteError** (std::complex< T > a, std::complex< T > b, T error=getDefault↩ Error())
- static bool **checkComplexRelativeError** (std::complex< T > a, std::complex< T > b, T relativeError=get↩ DefaultError())
- template<typename I , typename W >
  static bool checkComplexRootMeanSquaredError (const I *__restrict arrayA, const W *__restrict arrayB, std::size_t arraySize, T squaredError=getDefaultError(), std::enable_if_t<!std::is_floating_point< I >::value &&!std::is_floating_point< W >::value > *=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 error=getDefaultError(), std::enable_if_t<!std::is_floating_point< I >::value &&!std::is_floating↩ _point< W >::value > *=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 timeError=getDefaultTimeError(), T frequencyError=get↩ DefaultFrequencyError())
- 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 timeError=getDefaultTimeError(), T frequencyError=get↩ DefaultFrequencyError(), std::enable_if_t<!std::is_floating_point< I >::value &&!std::is_floating_point< W >::value > *=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 error=getDefaultError(), std::enable_if_t<!std::is_floating_point< I >::value &&!std↩ ::is_floating_point< W >::value > *=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 relativeError=getDefaultError(), std::enable_if_t<!std::is_floating_point< I >::value &&!std::is_floating_point< W >::value > *=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 **getDefaultError** ()
- static T **getDefaultTimeError** ()
- static T **getDefaultFrequencyError** ()

5.145.1 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

5.145.2 Member Function Documentation

5.145.2.1 checkComplexRootMeanSquaredError()

```
template<typename T >
template<typename I , typename W >
static bool Utils::UnitTests::UnitTestUtilityFunctions< T >::checkComplexRootMeanSquaredError
(
            const I *__restrict arrayA,
            const W *__restrict arrayB,
            std::size_t arraySize,
            T squaredError = getDefaultError(),
            std::enable_if_t<!std::is_floating_point< I >::value &&!std::is_floating_point<
W >::value > *  = 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.

5.145.2.2 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 error = getDefaultError(),
            std::enable_if_t<!std::is_floating_point< I >::value &&!std::is_floating_point<
W >::value > *  = 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.

5.145.2.3 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 timeError = getDefaultTimeError(),
            T frequencyError = getDefaultFrequencyError(),
            std::enable_if_t<!std::is_floating_point< I >::value &&!std::is_floating_point<
W >::value > *  = 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.

5.145.2.4 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 error = getDefaultError(),
            std::enable_if_t<!std::is_floating_point< I >::value &&!std::is_floating_point<
W >::value > *  = 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.

5.145.2.5 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 relativeError = getDefaultError(),
            std::enable_if_t<!std::is_floating_point< I >::value &&!std::is_floating_point<
W >::value > *  = nullptr )  [inline], [static]
```

Note: Template types I & W should not be decimals but only complex numbers of types std::complex, fftw & cufft.

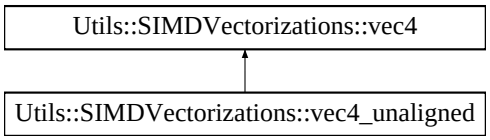
These template I & W types are not checked with template metaprogramming (besides if not decimal), so as to avoid dependencies to non-std complex numbers structs (fftw & cufft) in the Utils component.

5.146 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:



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)

5.146.1 Detailed Description

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

Author

Thanos Theo, 2009-2018

Version

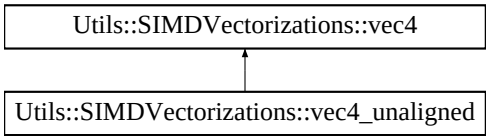
14.0.0.0

5.147 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:



Public Member Functions

- **vec4_unaligned** (const float *__restrict src)

5.147.1 Detailed Description

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

Author

Thanos Theo, 2009-2018

Version

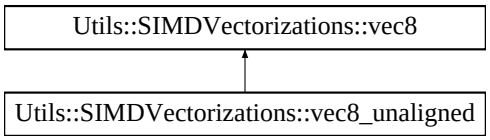
14.0.0.0

5.148 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:



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)

5.148.1 Detailed Description

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

Author

Thanos Theo, 2009-2018

Version

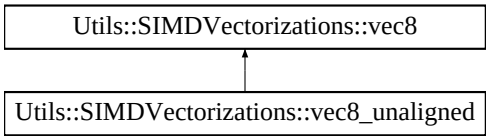
14.0.0.0

5.149 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:



Public Member Functions

- **vec8_unaligned** (const float *__restrict src)

5.149.1 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