Tellusim Core SDK

Generated by Doxygen 1.8.13

Contents

1 Namespace Index

1.1 Namespace List

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

2 Hierarchical Index

2.1 Class Hierarchy

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

3 Class Index

3.1 Class List

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

4 Namespace Documentation

4.1 Tellusim::Allocator Namespace Reference

Functions

- void * allocate (size_t size)

  raw allocation
- void * **reallocate** (void *ptr, size_t old_size, size_t new_size)
- void **free** (const void *ptr, size_t size)
- size_t getMemory ()

  memory statistics
- size_t **getAllocations** ()

4.1.1 Detailed Description

The Allocator namespace provides a flexible and efficient memory management interface for both raw and typed allocations. All allocations are guaranteed to be 32-byte aligned on binaries with AVX support; otherwise, allocations are 16-byte aligned. It supports low-level memory operations as well as high-level utilities for constructing and destroying C++ objects and arrays. For non-POD (plain old data) types, constructors and destructors are explicitly called to ensure proper object lifecycle management. The namespace also includes functions to retrieve memory usage statistics, to track the amount of total allocated memory and the number of active allocations.

4.1.2 Function Documentation

4.1.2.1 allocate()

```
static Type * Tellusim::Allocator::allocate (
            size_t size )
```

raw allocation

array allocation

4.2 Tellusim::Android Namespace Reference

Typedefs

- using Main = int32_t(int32_t argc, char **argv)

  Android native activity.

Functions

- ANativeActivity * getActivity ()

  Android activity.
- bool **isCreated** ()
- bool **isResumed** ()
- bool **isFocused** ()
- ANativeWindow * getWindow ()

  Android window.
- int32_t **getWidth** ()
- int32_t **getHeight** ()
- int32_t **getFormat** ()
- String getPackageName ()

  application package name
- String getHomeDirectory ()

  application directories
- String **getFilesDirectory** ()
- String **getCacheDirectory** ()
- String **getCardDirectory** ()
- String **getObbDirectory** ()
- void **onCreate** (ANativeActivity *activity, void *state, size_t size, Main *main)

4.2.1 Detailed Description

The Android namespace provides access to essential Android platform features, including native activity management, lifecycle status checks, window and screen properties, application package and directory paths, and initialization of the native entry point via the onCreate function.

4.3 Tellusim::Basis Namespace Reference

4.3.1 Detailed Description

Basis utils

4.4 Tellusim::Emscripten Namespace Reference

Typedefs

- using LoadedCallback = Function< void(Blob blob)>

  load user file
- using ProgressCallback = Function< void(uint32_t progress, uint32_t total)>

  asynchronous fetch
- using **FetchedCallback** = Function< void(const uint8_t *data, size_t size, uint32_t status)>

Functions

- void run (const char *src)

  run script
- int32_t **runi32** (const char *src)
- const char * **runs** (const char *src)
- void alert (const char *message)

  alert dialog
- void **alert** (const String &message)
- void **alertf** (const char *format,...) 1(1
- void void save (const Blob &blob, const char *mime=nullptr)

  save user file
- void **load** (const LoadedCallback &func, const char *type=nullptr)
- void **fetch** (const char *name, const FetchedCallback &fetched_func, const ProgressCallback &progress_↩ func, bool cache=false)
- void **fetch** (const String &name, const FetchedCallback &fetched_func, const ProgressCallback &progress↩ _func, bool cache=false)

4.4.1 Detailed Description

The Emscripten namespace provides functions for interacting with JavaScript and the browser in an Emscripten environment. It includes utilities for running scripts, displaying alert dialogs, saving and loading user files, and performing asynchronous fetch operations. These features enable seamless integration of native code with web-based functionalities.

4.5 Tellusim::Expression Namespace Reference

Functions

- int64_t getScalari64 (const char *src)
  scalar expressions
- uint64_t **getScalaru64** (const char *src)
- float32_t **getScalarf32** (const char *src)
- float64_t **getScalarf64** (const char *src)
- Vector2f getVector2f (const char *src, const char *type="Vector2f")
  vector expressions
- Vector3f **getVector3f** (const char *src, const char *type="Vector3f")
- Vector4f **getVector4f** (const char *src, const char *type="Vector4f")
- Matrix3x2f getMatrix3x2f (const char *src, const char *type="Matrix3x2f")
  matrix expressions
- Matrix4x3f **getMatrix4x3f** (const char *src, const char *type="Matrix4x3f")
- Matrix4x4f **getMatrix4x4f** (const char *src, const char *type="Matrix4x4f")

4.5.1 Detailed Description

Expression utils

4.6 Tellusim::iOS Namespace Reference

Enumerations

- enum Orientation {
  OrientationUnknown = 0,
  OrientationPortrait,
  OrientationPortraitUpsideDown,
  OrientationLandscapeLeft,
  OrientationLandscapeRight,
  OrientationFaceUp,
  OrientationFaceDown }
  Device orientation.

Functions

- void * getApplication ()
  iOS application
- bool **isCreated** ()
- bool **isFocused** ()
- uint32_t getWidth ()
  screen size
- uint32_t **getHeight** ()
- float32_t **getScale** ()
- String getModel ()
  device info
- Orientation **getOrientation** ()
- bool setKeyboardHidden (bool hidden)

virtual keyboard

- bool **isKeyboardHidden** ()
- String getHomeDirectory ()

  application home directory
- String getCachesDirectory ()

  application caches directory
- String getDocumentsDirectory ()

  application documents directory
- bool openUrl (const char *name)

  open url
- bool **openUrl** (const String &name)

4.6.1 Detailed Description

The iOS namespace provides access to key iOS platform features, including device orientation, screen size, device information, virtual keyboard management, and application directory paths. It also includes functions for managing the application lifecycle and opening URLs.

4.7 Tellusim::Line Namespace Reference

4.7.1 Detailed Description

Line utils

4.8 Tellusim::Log Namespace Reference

Typedefs

- using Callback = bool(Level, uint64_t time, const char *str, void *data)

  print callback

Enumerations

- enum Level {
  Fatal = 0,
  Error,
  Warning,
  Message,
  Verbose,
  Unknown,
  NumLevels }

  Log level.

Functions

- void setLevel (Level level)

  current Log level
- Level **getLevel** ()
- void **setCallback** (Callback *callback, void *data=nullptr)
- Callback * **getCallback** ()
- void * **getCallbackData** ()
- void **unlockCallback** ()
- void **lockCallback** ()
- void print (const char *str)

  print message
- void **vprintf** (const char *str, va_list args)
- void **printf** (const char *format,...) 1(1
- void void print (Level level, const char *str)

  print message with Level
- void **printe** (Level level, const char *str)
- void **vprintf** (Level level, const char *str, va_list args)
- void **vprintef** (Level level, const char *str, va_list args)
- void **printf** (Level level, const char *format,...) 1(2
- void void **printef** (Level level, const char *format,...) 1(2

4.8.1 Detailed Description

The Log namespace provides a system for managing and printing log messages at various levels of severity. It allows setting the current log level and printing messages with different severity levels, such as Fatal, Error, Warning, Message, and Verbose. The logging system supports both simple and formatted message output and includes the ability to use callbacks for custom logging behavior.

4.9 Tellusim::LU Namespace Reference

4.9.1 Detailed Description

Lower Upper Decomposition

4.10 Tellusim::MeshGraph Namespace Reference

Typedefs

- using ProgressCallback = Function< bool(uint32_t progress)>

  progress callback

Functions

- bool create (Mesh &dest, Mesh &src, uint32_t max_attributes, uint32_t max_primitives, const ProgressCallback *func=nullptr, Async *async=nullptr)
- bool **create** (Array< MeshGeometry > &dest, MeshGeometry &src, uint32_t max_attributes, uint32_t max_primitives, const ProgressCallback *func=nullptr, Async *async=nullptr)

4.10.1 Detailed Description

The MeshGraph namespace enables the creation of a hierarchical structure of geometries designed to perform smooth Level of Detail (LOD) transitions while preserving key details. It allows for reducing the complexity of a mesh by controlling the number of vertices and indices in each geometry, based on predefined limits. This hierarchy supports seamless transitions between different levels of detail, ensuring that higher detail is maintained where necessary, and lower detail is used in distant or less critical areas.

4.10.2 Function Documentation

4.10.2.1 create()

```
bool Tellusim::MeshGraph::create (
            Mesh & dest,
            Mesh & src,
            uint32_t max_attributes,
            uint32_t max_primitives,
            const ProgressCallback * func = nullptr,
            Async * async = nullptr )
```

mesh graph reduction

Parameters

max_attributes	Maximum number of attributes per geometry
max_primitives	Maximum number of primitives per geometry

4.11 Tellusim::MeshReduce Namespace Reference

Typedefs

- using ProgressCallback = Function< bool(uint32_t progress)>

  Progress callback.

Functions

- bool collapse (Mesh &dest, const Mesh &src, float32_t ratio, float32_t threshold=0.0f, const ProgressCallback *func=nullptr)
- bool **collapse** (MeshGeometry &dest, const MeshGeometry &src, float32_t ratio, float32_t threshold=0.0f, const ProgressCallback *func=nullptr, uint32_t position=Maxu32)

4.11.1 Detailed Description

The MeshReduce namespace contains algorithms for reducing the complexity of 3D meshes, by collapsing vertices and edges to reduce the number of triangles. This is commonly used in mesh optimization workflows, such as generating level-of-detail versions of models.

4.11.2 Function Documentation

4.11.2.1 collapse()

```
bool Tellusim::MeshReduce::collapse (
            Mesh & dest,
            const Mesh & src,
            float32_t ratio,
            float32_t threshold = 0.0f,
            const ProgressCallback * func = nullptr )
```

Performs mesh reduction

Parameters

ratio	Triangle reduction ratio (use negative ratio for border reduction)
threshold	Edge collapse threshold
position	Position attribute index

4.12 Tellusim::MeshRefine Namespace Reference

Functions

- bool subdiv (Mesh &dest, const Mesh &src, uint32_t steps)
- bool **subdiv** (MeshGeometry &dest, const MeshGeometry &src, uint32_t steps, uint32_t position=Maxu32)

4.12.1 Detailed Description

The MeshRefine namespace provides algorithms for refining or subdividing 3D meshes. Subdivision increases the resolution of a mesh by splitting its faces, allowing for smoother geometry.

4.12.2 Function Documentation

4.12.2.1 subdiv()

```
bool Tellusim::MeshRefine::subdiv (
            Mesh & dest,
            const Mesh & src,
            uint32_t steps )
```

Performs mesh subdivision

Parameters

steps	Number of subdivision steps. Each step increases the triangle count.

4.13 Tellusim::MeshSolid Namespace Reference

Typedefs

- using ProgressCallback = Function< bool(uint32_t progress)>

  progress callback

Functions

- bool create (Mesh &dest, const Mesh &src, float32_t ratio=1.0f, float32_t threshold=0.9f, const Progress↩ Callback *func=nullptr)
- bool **create** (MeshGeometry &dest, const MeshGeometry &src, float32_t ratio=1.0f, float32_t threshold=0.9f, const ProgressCallback *func=nullptr, uint32_t position=Maxu32)

4.13.1 Detailed Description

The MeshSolid namespace provides algorithms for converting surface meshes into volumetric tetrahedral meshes using an advancing front generation method.

4.13.2 Function Documentation

4.13.2.1 create()

```
bool Tellusim::MeshSolid::create (
            Mesh & dest,
            const Mesh & src,
            float32_t ratio = 1.0f,
            float32_t threshold = 0.9f,
            const ProgressCallback * func = nullptr )
```

Generates a solid (tetrahedral) mesh from a surface Mesh using an advancing front algorithm.

Parameters

ratio	Tetrahedron height ratio
threshold	Delaunay radius threshold
position	Position attribute index

4.14 Tellusim::Noise Namespace Reference

Functions

- template<class Type >
  void mod289 (Type &x)

  utils

- template<class Type >
  void **perm** (Type &x)
- template<class Type >
  Type **fract** (const Type &x)
- template<class Type >
  Type **rsqrt_fast** (const Type &x)
- template<class Type >
  Type perlin (const Type &x, const Type &y, bool cubic=false)

  2D Perlin noise
- template<class Type >
  Type fractal (Type x, Type y, uint32_t steps=5, float32_t scale=0.5f)

  2D Fractal noise

4.14.1 Detailed Description

Noise

4.15 Tellusim::Order Namespace Reference

Functions

- template<class Type = uint32_t>
  Type hilbert2 (uint32_t size, uint32_t x, uint32_t y)

  Hilbert curve.
- template<class Type = uint32_t>
  void ihilbert2 (uint32_t size, Type index, uint32_t &x, uint32_t &y)

  inverse Hilbert curve

4.15.1 Detailed Description

Order utils

4.16 Tellusim::Parser Namespace Reference

Functions

- bool isSpace (char c)

  character types
- bool **isAlpha** (char c)
- bool **isLower** (char c)
- bool **isUpper** (char c)
- bool **isLiteral** (char c)
- bool **isDecimal** (char c)
- bool **isHexadecimal** (char c)
- bool isComment (const char *str)

  string types
- bool **isNumber** (const char *str)
- bool **isFloat** (const char *str)
- bool **isSigned** (const char *str)

- bool **isUnsigned** (const char *str)
- bool **isBom** (const char *str)
- uint32_t skipSpaces (const char *str)

  skip symbols
- uint32_t **skipSpaces** (const char *str, uint32_t &line)
- uint32_t skipComment (const char *str)

  skip comment
- uint32_t **skipComment** (const char *str, uint32_t &line)
- char getSymbol (const char *str)

  expect symbols
- uint32_t **expectSymbol** (const char *str, char c)
- uint32_t **expectSymbol** (const char *str, char c, uint32_t &line)
- uint32_t skipToken (const char *str)
- uint32_t **skipToken** (const char *str, const char *term)
- uint32_t **readToken** (const char *str, String &dest, bool append=false)
- uint32_t **readToken** (const char *str, String &dest, const char *term, bool append=false)
- uint32_t expectToken (const char *str, const char *token)

  expect space-separated token
- uint32_t **expectToken** (const char *str, const char *token, const char *term)
- uint32_t skipName (const char *str)
- uint32_t **skipName** (const char *str, const char *pass)
- uint32_t **readName** (const char *str, String &dest, bool append=false)
- uint32_t **readName** (const char *str, String &dest, const char *pass, bool append=false)
- uint32_t expectName (const char *str, const char *name)

  expect literal name
- uint32_t **expectName** (const char *str, const char *name, const char *pass)
- uint32_t skipFloat (const char *str)
- uint32_t **readFloat** (const char *str, String &dest, bool append=false)
- uint32_t skipDecimal (const char *str)
- uint32_t **readDecimal** (const char *str, String &dest, bool append=false)
- uint32_t skipHexadecimal (const char *str)
- uint32_t **readHexadecimal** (const char *str, String &dest, bool append=false)
- uint32_t skipNumber (const char *str)
- uint32_t **readNumber** (const char *str, String &dest, bool append=false)
- uint32_t skipSymbol (const char *str)
- uint32_t **readSymbol** (const char *str, String &dest, bool enclose=false, bool append=false)
- uint32_t skipString (const char *str)
- uint32_t **readString** (const char *str, String &dest, bool enclose=false, bool append=false)
- uint32_t readRegion (const char *str, String &dest, char from, char to, bool enclose=false, bool append=false)
- uint32_t skipLines (const char *str, uint32_t lines)
- uint32_t **skipLine** (const char *str, bool escape=false)
- uint32_t **readLine** (const char *str, String &dest, bool escape=false, bool append=false)
- uint32_t readBom (const char *str, String &dest, bool append=false)

  read string with byte ordered mark
- void error (const char *format,...) 1(1

  throw Parser error

4.16.1 Detailed Description

The Parser namespace provides functions for handling and processing ASCII text. It supports a variety of operations, such as recognizing specific characters, reading tokens, and skipping over content like whitespace, comments, or symbols. These functions are crucial for parsing strings in a structured manner, typically used in the context of processing programming languages, configuration files, or other structured text formats.

4.16.2 Function Documentation

4.16.2.1 skipToken()

```
uint32_t Tellusim::Parser::skipToken (
            const char * str )
```

read space-separated token

Parameters

append	Append to the destination

4.16.2.2 skipName()

```
uint32_t Tellusim::Parser::skipName (
            const char * str )
```

read literal name

Parameters

append	Append to the destination

4.16.2.3 skipFloat()

```
uint32_t Tellusim::Parser::skipFloat (
            const char * str )
```

read floating-point number

Parameters

append	Append to the destination

4.16.2.4 skipDecimal()

```
uint32_t Tellusim::Parser::skipDecimal (
            const char * str )
```

read decimal number

Parameters

append	Append to the destination

4.16.2.5 skipHexadecimal()

```
uint32_t Tellusim::Parser::skipHexadecimal (
            const char * str )
```

read hexadecimal number

Parameters

append	Append to the destination

4.16.2.6 skipNumber()

```
uint32_t Tellusim::Parser::skipNumber (
            const char * str )
```

read floating-point or integer number

Parameters

append	Append to the destination

4.16.2.7 skipSymbol()

```
uint32_t Tellusim::Parser::skipSymbol (
            const char * str )
```

read single-quoted symbol

Parameters

enclose	Include quote symbols
append	Append to the destination

4.16.2.8 skipString()

```
uint32_t Tellusim::Parser::skipString (
            const char * str )
```

read quoted string

Parameters

enclose	Include quote symbols
append	Append to the destination

4.16.2.9 readRegion()

```
uint32_t Tellusim::Parser::readRegion (
             const char * str,
             String & dest,
             char from,
             char to,
             bool enclose = false,
             bool append = false )
```

read region of symbols

Parameters

enclose	Include from/begin symbols
append	Append to the destination

4.16.2.10 skipLines()

```
uint32_t Tellusim::Parser::skipLines (
             const char * str,
             uint32_t lines )
```

read line of symbols

Parameters

escape	Read multiple lines
append	Append to the destination

4.17 Tellusim::QR Namespace Reference

4.17.1 Detailed Description

Orthogonal Decomposition

4.18 Tellusim::Quadrilateral Namespace Reference

4.18.1 Detailed Description

Quadrilateral utils

4.19 Tellusim::Spatial Namespace Reference

Classes

- struct Node

  Spatial Node.

Typedefs

- using Node2f = Node< BoundRect< float32_t, Vector2< float32_t > >, 2 >

  Node types.
- using **Node2d** = Node< BoundRect< float64_t, Vector2< float64_t > >, 2 >
- using **Node3f** = Node< BoundBox< float32_t, Vector3< float32_t > >, 3 >
- using **Node3d** = Node< BoundBox< float64_t, Vector3< float64_t > >, 3 >
- using **Node4f** = Node< BoundBox< float32_t, Vector4< float32_t > >, 3 >
- using **Node4d** = Node< BoundBox< float64_t, Vector4< float64_t > >, 3 >

Enumerations

- enum {
  MinIterations = 1,
  MaxIterations = 8,
  DefaultIterations = 4 }

  number of iterations

Functions

- template<class Type , class Vector >
  Type get_weight (const BoundRect< Type, Vector > &bound)

  Create spatial tree.
- template<class Type , class Vector >
  Type **get_weight** (const BoundBox< Type, Vector > &bound)
- template<class Bound >
  Bound::Type **get_weight** (const Bound &bound)

4.19.1 Detailed Description

Spatial utils

4.20 Tellusim::SVD Namespace Reference

Functions

- template<class Type >
  Type pythagorean (Type a, Type b)

  Computes sqrt(a^2 + b^2) function.

4.20.1 Detailed Description

Singular Value Decomposition

4.21 Tellusim::System Namespace Reference

Functions

- uint32_t getThreadID ()
- bool setEnvironment (const char *name, const char *value)
- bool **setEnvironment** (const String &name, const char *value)
- String **getEnvironment** (const char *name)
- String **getEnvironment** (const String &name)
- void * loadLibrary (const char *name)
- void * **loadLibrary** (const String &name)
- void * **getFunction** (void *handle, const char *name)
- void * **getFunction** (void *handle, const String &name)
- void **closeLibrary** (void *handle)
- bool exec (const char *command, bool wait=false)
- bool **exec** (const String &command, bool wait=false)
- bool open (const char *command)
- bool **open** (const String &command)

4.21.1 Detailed Description

The System namespace offers functions for system-level operations, including retrieving the current thread identifier, managing environment variables, and working with dynamic libraries. It also provides functionality for executing commands with exec, optionally waiting for completion, and opening resources or applications using the open function.

4.21.2 Function Documentation

4.21.2.1 getThreadID()

```
uint32_t Tellusim::System::getThreadID ( )
```

Thread identifier

4.21.2.2 setEnvironment()

```
bool Tellusim::System::setEnvironment (
            const char * name,
            const char * value )
```

Environment variables

4.21.2.3 loadLibrary()

```
void* Tellusim::System::loadLibrary (
            const char * name )
```

Dynamic libraries

4.21.2.4 exec()

```
bool Tellusim::System::exec (
            const char * command,
            bool wait = false )
```

Execute command

4.21.2.5 open()

```
bool Tellusim::System::open (
            const char * command )
```

Open resource

4.22 Tellusim::Time Namespace Reference

Enumerations

- enum {
  Seconds = 1000000u,
  MSeconds = 1000u,
  USeconds = 1u }

Functions

- uint64_t current ()
  current system time in microseconds
- float64_t seconds ()
  current process time in seconds
- void sleep (uint32_t usec)
  sleep process in microseconds

4.22.1 Detailed Description

The Time namespace provides functions and constants for handling time-related operations.

4.23 Tellusim::Triangle Namespace Reference

4.23.1 Detailed Description

Triangle utils

4.24 Tellusim::tvOS Namespace Reference

Functions

- void * getApplication ()
  tvOS application
- bool **isCreated** ()
- bool **isFocused** ()
- uint32_t getWidth ()
  screen size
- uint32_t **getHeight** ()
- float32_t **getScale** ()
- String getModel ()
  device info
- bool setKeyboardHidden (bool hidden)
  virtual keyboard
- bool **isKeyboardHidden** ()
- String getHomeDirectory ()
  application home directory
- String getCachesDirectory ()
  application caches directory
- String getDocumentsDirectory ()
  application documents directory
- bool openUrl (const char *name)
  open url
- bool **openUrl** (const String &name)

4.24.1 Detailed Description

The tvOS namespace provides access to key tvOS platform features, including screen size, device information, virtual keyboard management, and application directory paths. It also includes functions for managing the application lifecycle and opening URLs.

4.25 Tellusim::WinApp Namespace Reference

Typedefs

- using Main = int32_t(int32_t argc, char **argv)
  WinApp main.

Functions

- void * getInstance ()
  application instance
- int32_t getShowMode ()
  application show mode
- void * getWindow ()
  application window
- String getLocalDirectory ()
  application directories
- String **getCacheDirectory** ()
- String **getTempDirectory** ()
- void **main** (void *instance, void *prev_instance, wchar_t *command, int32_t show_mode, Main *main)

4.25.1 Detailed Description

The WinApp namespace provides functions for managing Windows application instances, show modes, and directories. It allows access to the application window and directories like local, cache, and temporary. The namespace also defines a main entry point for WinApp applications.

4.26 Tellusim::Windows Namespace Reference

Typedefs

- using Main = int32_t(int32_t argc, char **argv)

  Windows main.

Functions

- void * getInstance ()

  application instance
- int32_t getShowMode ()

  application show mode
- void * getConsoleHandle ()

  console window handle
- bool isConsoleCreated ()

  check console status
- bool createConsole (const char *title, uint32_t width=0, uint32_t height=0, int32_t x=Maxi32, int32_t y=Maxi32)

  create console window
- bool **createConsole** (const String &title, uint32_t width=0, uint32_t height=0, int32_t x=Maxi32, int32_t y=Maxi32)
- void setConsoleTitle (const char *title)

  set console title
- void **setConsoleTitle** (const String &title)
- String **getConsoleTitle** ()
- void setConsoleGeometry (uint32_t width, uint32_t height, int32_t x=Maxi32, int32_t y=Maxi32)

  console geometry
- uint32_t **getConsoleWidth** ()
- uint32_t **getConsoleHeight** ()
- int32_t **getConsolePositionX** ()
- int32_t **getConsolePositionY** ()
- void setConsoleHidden (bool hidden)

  hide console window
- bool **isConsoleHidden** ()
- void **main** (void *instance, void *prev_instance, wchar_t *command, int32_t show_mode, Main *main)

4.26.1 Detailed Description

The Windows namespace provides functions for managing Windows application instances, console windows, and their properties. It includes operations to create and configure a console window, set its title and geometry, check its visibility, and access console handle and status. The namespace also defines a main entry point for Windows applications.

5 Class Documentation

5.1 Tellusim::App Class Reference

`#include <TellusimApp.h>`

Public Types

- enum {
  Version_19 = 20221010,
  Version_20 = 20221109,
  Version_21 = 20221122,
  Version_22 = 20221222,
  Version_23 = 20230117,
  Version_24 = 20230217,
  Version_25 = 20230402,
  Version_26 = 20230509,
  Version_27 = 20230612,
  Version_28 = 20230718,
  Version_29 = 20230824,
  Version_30 = 20231029,
  Version_31 = 20231113,
  Version_32 = 20231212,
  Version_33 = 20240116,
  Version_34 = 20240216,
  Version_35 = 20240320,
  Version_36 = 20240427,
  Version_37 = 20240515,
  Version_38 = 20250215,
  Version_39 = 20250322,
  Version_40 = 20250429,
  Version = Version_40 }

  Release version.

Public Member Functions

- **App** (int32_t argc, char **argv)
- void clear ()

  clear application parameters
- Platform getPlatform () const

  default command line parameters
- uint32_t **getDevice** () const
- uint32_t **getWidth** () const
- uint32_t **getHeight** () const
- uint32_t **getMultisample** () const
- uint32_t getNumArguments () const

  custom command line arguments
- const String & **getArgument** (uint32_t num) const
- const Array< String > & **getArguments** () const
- bool isArgument (const char *name) const

  checks if the command line argument is present
- const String & getArgument (const char *name) const
- bool create (Platform platform=PlatformUnknown, uint32_t version=Version)

Static Public Member Functions

- static void setPlatform (Platform platform, uint32_t device=Maxu32)

  set default application parameters (will be overridden by command line parameters)
- static void **setSize** (uint32_t width, uint32_t height, uint32_t multisample=0)
- static bool isBuildDebug ()

  binary info
- static bool **isBuildFloat64** ()
- static String **getBuildDate** ()
- static String **getBuildInfo** ()
- static uint32_t **getVersion** ()
- static uint64_t **getAPIHash** ()

5.1.1 Detailed Description

The App class represents an application that is initialized and configured using command-line arguments and platform-specific settings. It provides mechanisms to retrieve various command-line parameters, such as platform type, device, window dimensions, and multisampling settings. The class also handles the application version control through predefined release versions.

5.1.2 Member Function Documentation

5.1.2.1 getArgument()

```
const String& Tellusim::App::getArgument (
            const char * name ) const
```

get command line argument by name

Parameters

name	Command line argument name.

Returns

Returns String::null if the argument is not present

5.1.2.2 create()

```
bool Tellusim::App::create (
            Platform platform = PlatformUnknown,
            uint32_t version = Version )
```

create application

Returns

Returns false if the platform is not supported or the version is wrong.

5.2 Tellusim::Archive Class Reference

`#include <format/TellusimArchive.h>`

Public Member Functions

- bool open (const char *name, const char *type=nullptr)
  open/close archive
- bool **open** (const String &name, const char *type=nullptr)
- bool **open** (Stream &stream, const char *type=nullptr)
- void **close** ()
- bool isOpened () const
  archive status
- String **getName** () const
- uint32_t getNumFiles () const
  files list
- String **getFileName** (uint32_t index) const
- uint64_t **getFileMTime** (uint32_t index) const
- size_t **getFileSize** (uint32_t index) const
- uint32_t findFile (const char *name) const
  find file
- uint32_t **findFile** (const String &name) const
- bool **isFile** (const char *name) const
- bool **isFile** (const String &name) const
- Stream openFile (const char *name) const
  open file
- Stream **openFile** (const String &name) const
- Stream **openFile** (uint32_t index) const

5.2.1 Detailed Description

The Archive class provides functionality for managing and accessing archive files. It allows opening and closing archives from various sources, including file names, strings, and streams. The class provides methods for retrieving metadata about files within the archive, such as file names, sizes, and modification times. Additionally, it supports searching for specific files by name or index and enables accessing files within the archive through a stream interface for reading.

5.3 Tellusim::ArchiveStream Class Reference

`#include <format/TellusimArchive.h>`

Public Member Functions

- virtual ArchiveStream * instance () const =0
  create instance
- virtual void **destructor** (ArchiveStream *instance) const =0
- virtual bool open (Stream &stream, const char *name)=0
  open archive
- virtual uint32_t getNumFiles () const =0
  files list
- virtual const String & **getFileName** (uint32_t index) const =0
- virtual uint64_t **getFileMTime** (uint32_t index) const =0
- virtual size_t **getFileSize** (uint32_t index) const =0
- virtual Stream openFile (uint32_t index)=0
  open file

Static Public Member Functions

- static bool check (const String &name)
  archive stream formats
- static String getFormats ()
  list of supported formats

Protected Member Functions

- **ArchiveStream** (const char *name)
- **ArchiveStream** (const InitializerList< const char *> &names)

5.3.1 Detailed Description

The ArchiveStream class is a base class designed for creating custom archive formats. It provides virtual methods for opening archive streams, accessing metadata about the files within the archive (such as file names, sizes, and modification times), and opening individual files from the archive. This class serves as a base class for implementing specific archive stream formats, allowing for the customization and extension of archive handling functionality. It also includes static methods for checking supported formats and retrieving a list of compatible formats, providing flexibility in managing and working with various custom archive formats.

5.4 Tellusim::Async Class Reference

```
#include <core/TellusimAsync.h>
```

Classes

- class Task
  Task.

Public Member Functions

- bool init (uint32_t num=0)
  initialize threads
- bool shutdown ()
  shutdown threads
- bool isInitialized () const
  check status
- uint32_t **getNumThreads** () const
- void append (const Task &task)
  temporary tasks
- void **clear** ()
- uint32_t getNumTasks (bool check=false) const
  number of temporary tasks
- Task run (uint32_t mask)
  run function
- Task **run** (uint64_t mask=~0ull)

- template<class Func >
  Task **run** (const Func &func, uint64_t mask=~0ull)
- template<class Func , class A0 >
  Task **run** (const Func &func, A0 a0, uint64_t mask=~0ull)
- template<class Func , class A0 , class A1 >
  Task **run** (const Func &func, A0 a0, A1 a1, uint64_t mask=~0ull)
- template<class Func , class A0 , class A1 , class A2 >
  Task **run** (const Func &func, A0 a0, A1 a1, A2 a2, uint64_t mask=~0ull)
- template<class Func , class A0 , class A1 , class A2 , class A3 >
  Task **run** (const Func &func, A0 a0, A1 a1, A2 a2, A3 a3, uint64_t mask=~0ull)
- template<class Func , class A0 , class A1 , class A2 , class A3 , class A4 >
  Task **run** (const Func &func, A0 a0, A1 a1, A2 a2, A3 a3, A4 a4, uint64_t mask=~0ull)
- template<class Func , class A0 , class A1 , class A2 , class A3 , class A4 , class A5 >
  Task **run** (const Func &func, A0 a0, A1 a1, A2 a2, A3 a3, A4 a4, A5 a5, uint64_t mask=~0ull)
- template<class Func , class A0 , class A1 , class A2 , class A3 , class A4 , class A5 , class A6 >
  Task **run** (const Func &func, A0 a0, A1 a1, A2 a2, A3 a3, A4 a4, A5 a5, A6 a6, uint64_t mask=~0ull)
- template<class Func , class A0 , class A1 , class A2 , class A3 , class A4 , class A5 , class A6 , class A7 >
  Task **run** (const Func &func, A0 a0, A1 a1, A2 a2, A3 a3, A4 a4, A5 a5, A6 a6, A7 a7, uint64_t mask=~0ull)
- template<class Func >
  Task **run** (const Function< Func > &func, uint64_t mask=~0ull)
- bool check (const Task *tasks, uint32_t num) const

  check completion status
- bool **check** (const Array< Task > &tasks) const
- bool **check** () const
- bool wait (const Task *tasks, uint32_t num) const

  waiting for the completion
- bool **wait** (const Array< Task > &tasks) const
- bool **wait** () const

Static Public Member Functions

- static uint32_t getNumCores ()

  number of threads

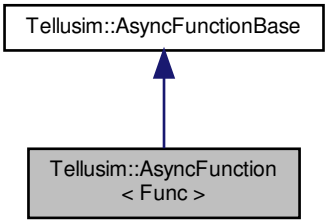
5.4.1 Detailed Description

The Async class in the Tellusim API is a high-performance, multithreaded task scheduler designed for executing functions asynchronously across multiple CPU threads. It allows developers to dispatch tasks-functions with or without return values-either immediately or with a specific CPU affinity mask, enabling fine-grained control over thread usage. Internally, tasks are encapsulated using the AsyncFunction system, which supports various function signatures and handles function invocation and result storage. The Async class includes an inner Task class that represents individual asynchronous jobs. These tasks can be queued, cleared, canceled, or monitored for completion, and they offer mechanisms for retrieving return values in a type-safe way. This architecture is particularly suited for real-time or compute-intensive applications, such as rendering or simulation engines, where efficient CPU utilization and concurrent task execution are critical.

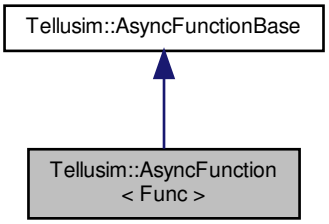
5.5 Tellusim::AsyncFunction< Func > Class Template Reference

`#include <core/TellusimAsync.h>`

Inheritance diagram for Tellusim::AsyncFunction< Func >:



Collaboration diagram for Tellusim::AsyncFunction< Func >:



Public Types

- using Ret = typename Func::Ret

  function return type

Public Member Functions

- **AsyncFunction** (const Func &func)
- virtual void run ()

  run function
- virtual void * get ()

  return value pointer

Additional Inherited Members

5.5.1 Detailed Description

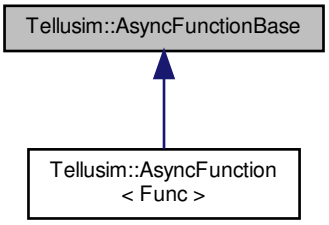
template<class Func>
class Tellusim::AsyncFunction< Func >

Async function template

5.6 Tellusim::AsyncFunctionBase Class Reference

`#include <core/TellusimAsync.h>`

Inheritance diagram for Tellusim::AsyncFunctionBase:



Public Member Functions

- virtual void run ()=0
  run function
- virtual void * get ()=0
  function result

Static Public Member Functions

- static void release (AsyncFunctionBase *func)
  release function pointer

5.6.1 Detailed Description

Async function base class

5.7 Tellusim::AsyncFunctionRet< Type > Struct Template Reference

`#include <core/TellusimAsync.h>`

Public Types

- using **Ret** = Type

5.7.1 Detailed Description

template<class Type>
struct Tellusim::AsyncFunctionRet< Type >

Async function return type

5.8 Tellusim::AsyncFunctionRet< void > Struct Template Reference

Public Types

- using **Ret** = void *

5.9 Tellusim::Atlas< Type > Class Template Reference

```
#include <geometry/TellusimAtlas.h>
```

Classes

- struct Node
  Atlas Node.

Public Types

- enum { **Axes** = Vector::Size }
- using **Bound** = Type
- using **Vector** = typename Type::Vector

Public Member Functions

- Atlas ()
  constructors
- **Atlas** (const Atlas &atlas)
- **Atlas** (Atlas &&atlas)
- **Atlas** (const Vector &size)
- **Atlas** (const Bound &bound)
- void clear ()
  clear atlas
- void set (const Vector &size)
  set atlas size
- void **set** (const Bound &bound)
- Atlas & operator= (const Atlas &atlas)
  assignment operators
- Atlas & **operator=** (Atlas &&atlas)
- const Node * getRoot () const
  atlas root
- Node * insert (const Vector &size)
  insert node into the atlas
- bool remove (Node *node)
  remove node from the atlas

5.9.1 Detailed Description

template<class Type>
class Tellusim::Atlas< Type >

Atlas utils

5.10 Tellusim::AtomicArray< Type > Class Template Reference

`#include <core/TellusimAtomic.h>`

Public Member Functions

- **AtomicArray** (uint32_t size)
- **AtomicArray** (uint32_t size, const Type *array)
- **AtomicArray** (uint32_t size, const Type &value)
- **AtomicArray** (AtomicArray &array)
- void reserve (uint32_t size)
  resize array
- void **resize** (uint32_t size, bool reserve=false, bool discard=false)
- void **resize** (uint32_t size, const Type &value, bool reserve=false)
- void release ()
  clear array
- void **clear** ()
- void swap (AtomicArray &array)
  swap arrays
- void copy (const Type *array, uint32_t size)
  copy value
- void **copy** (AtomicArray &array)
- AtomicArray & operator= (AtomicArray &array)
  assignment operator
- void append (const Type &value)
  append value
- AtomicArray & **append** (const Type *array, uint32_t size)
- AtomicArray & **append** (AtomicArray &array)
- void removeFast (uint32_t pos, uint32_t len=1)
  remove value
- void **remove** (uint32_t pos, uint32_t len=1)
- bool empty ()
  array info
- **operator bool** ()
- uint32_t **bytes** ()
- uint32_t **memory** ()
- uint32_t **size** ()
- Type * get ()
  array data
- Type & **operator[ ]** (uint32_t index)
- Type & **get** (uint32_t index)
- template<class T >
  uint32_t findIndex (const T &value)
  array data
- Type * begin ()
  array iterators
- Type * **end** ()

5.10.1 Detailed Description

template<class Type>
class Tellusim::AtomicArray< Type >

Atomic Array

5.11 Tellusim::Atomici32 Struct Reference

`#include <core/TellusimAtomic.h>`

Public Member Functions

- **Atomici32** (int32_t value=0)
- **operator int32_t** ()
- Atomici32 & **operator=** (int32_t value)
- int32_t operator++ ()

  atomic operators
- int32_t **operator--** ()
- int32_t **operator++** (int32_t)
- int32_t **operator--** (int32_t)
- int32_t **operator+=** (int32_t v)
- int32_t **operator-=** (int32_t v)
- int32_t **operator &=** (int32_t v)
- int32_t **operator|=** (int32_t v)
- void set (int32_t v)

  atomic functions
- int32_t **get** ()
- bool **cas** (int32_t old_value, int32_t new_value)

Public Attributes

- volatile int32_t **value**

5.11.1 Detailed Description

32-bit integer Atomic

5.12 Tellusim::Atomici64 Struct Reference

`#include <core/TellusimAtomic.h>`

Public Member Functions

- **Atomici64** (int64_t value=0)
- **operator int64_t** ()
- Atomici64 & **operator=** (int64_t value)
- int64_t operator++ ()

  atomic operators
- int64_t **operator--** ()
- int64_t **operator++** (int32_t)
- int64_t **operator--** (int32_t)
- int64_t **operator+=** (int64_t v)
- int64_t **operator-=** (int64_t v)
- int64_t **operator &=** (int64_t v)
- int64_t **operator|=** (int64_t v)
- void set (int64_t v)

  atomic functions
- int64_t **get** ()
- bool **cas** (int64_t old_value, int64_t new_value)

Public Attributes

- volatile int64_t **value**

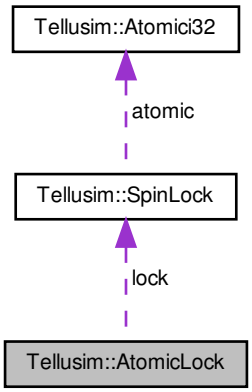
5.12.1 Detailed Description

64-bit integer Atomic

5.13 Tellusim::AtomicLock Struct Reference

`#include <core/TellusimAtomic.h>`

Collaboration diagram for Tellusim::AtomicLock:



Public Member Functions

- **AtomicLock** (SpinLock &lock)

Public Attributes

- SpinLock & **lock**

5.13.1 Detailed Description

AtomicLock class

5.14 Tellusim::AtomicPtr< Type > Struct Template Reference

```
#include <core/TellusimAtomic.h>
```

Public Member Functions

- **AtomicPtr** (Type *ptr=nullptr)
- AtomicPtr & **operator=** (Type *ptr)
- **operator bool** ()
- Type & **operator[ ]** (int32_t index)
- Type & **operator[ ]** (uint32_t index)
- Type * operator++ ()

  atomic operators
- Type * **operator--** ()
- Type * **operator++** (int32_t)
- Type * **operator--** (int32_t)
- Type * **operator+=** (size_t v)
- Type * **operator-=** (size_t v)
- void set (Type *p)

  atomic functions
- Type * **get** ()
- bool **cas** (Type *old_ptr, Type *new_ptr)

Public Attributes

- volatile Type * **ptr**

5.14.1 Detailed Description

template<class Type>
struct Tellusim::AtomicPtr< Type >

Atomic pointer

5.15 Tellusim::BitonicSort Class Reference

`#include <parallel/TellusimBitonicSort.h>`

Classes

- struct DispatchParameters

Public Types

- enum Mode {
  ModeSingle = 0,
  ModeMultiple,
  NumModes }

  Sort modes.
- enum Flags {
  FlagNone = 0,
  FlagSingle = (1 << ModeSingle),
  FlagMultiple = (1 << ModeMultiple),
  FlagIndirect = (1 << (NumModes + 0)),
  FlagOrder = (1 << (NumModes + 1)),
  FlagsAll = (FlagSingle | FlagMultiple | FlagIndirect | FlagOrder) }

  Sort flags.

Public Member Functions

- void clear ()

  clear sort
- bool isCreated (Flags flags) const

  check sort
- uint32_t getDataSize () const

  sort parameters
- uint32_t **getGroupSize** () const
- uint32_t **getSortElements** () const
- uint32_t **getMaxRegions** () const
- bool create (const Device &device, Mode mode, uint32_t size, uint32_t groups=256, uint32_t regions=1, Async *async=nullptr)
- bool **create** (const Device &device, Flags flags, uint32_t size, uint32_t groups=256, uint32_t regions=1, Async *async=nullptr)
- bool dispatch (Compute &compute, Buffer &data, uint32_t keys_offset, uint32_t data_offset, uint32_t size, Flags flags=FlagNone)
- bool dispatch (Compute &compute, Buffer &data, uint32_t count, const uint32_t *keys_offsets, const uint32_t *data_offsets, const uint32_t *sizes, Flags flags=FlagNone)
- bool dispatchIndirect (Compute &compute, Buffer &data, Buffer &dispatch, uint32_t offset, Flags flags=FlagNone)
- bool dispatchIndirect (Compute &compute, Buffer &data, uint32_t count, Buffer &dispatch, uint32_t offset, Flags flags=FlagNone)
- bool dispatchIndirect (Compute &compute, Buffer &data, Buffer &count, Buffer &dispatch, uint32_t count_offset, uint32_t dispatch_offset, Flags flags=FlagNone)

5.15.1 Detailed Description

The BitonicSort class implements an efficient parallel bitonic sorting algorithm designed for sorting large datasets on GPUs. It supports both single and multiple array sorting, and provides flexible configurations through various modes and flags. The class can handle sorting tasks that involve both direct and indirect dispatching, making it suitable for complex scenarios such as sorting multiple regions or handling different element alignments.

5.15.2 Member Function Documentation

5.15.2.1 create()

```
bool Tellusim::BitonicSort::create (
             const Device & device,
             Mode mode,
             uint32_t size,
             uint32_t groups = 256,
             uint32_t regions = 1,
             Async * async = nullptr )
```

create bitonic sort

Parameters

size	Maximum number of sorted elements.
groups	Bitonic sort group size.
regions	Maximum number of multiple regions.

5.15.2.2 dispatch() [1/2]

```
bool Tellusim::BitonicSort::dispatch (
             Compute & compute,
             Buffer & data,
             uint32_t keys_offset,
             uint32_t data_offset,
             uint32_t size,
             Flags flags = FlagNone )
```

dispatch single in-place bitonic sort

Parameters

data	Buffer of uint32_t data elements to sort.
keys_offset	Keys elements offset index (2 aligned).
data_offset	Data elements offset index (2 aligned).
size	Number of uint32_t elements to sort.

5.15.2.3 dispatch() [2/2]

```
bool Tellusim::BitonicSort::dispatch (
            Compute & compute,
            Buffer & data,
            uint32_t count,
            const uint32_t * keys_offsets,
            const uint32_t * data_offsets,
            const uint32_t * sizes,
            Flags flags = FlagNone )
```

dispatch multiple in-place bitonic sorts

Parameters

data	Buffer of uint32_t data elements to sort.
count	Number of regions to sort.
keys_offsets	Keys elements offset index (2 aligned).
data_offsets	Data elements offset index (2 aligned).
sizes	Number of uint32_t elements to sort.

5.15.2.4 dispatchIndirect() [1/3]

```
bool Tellusim::BitonicSort::dispatchIndirect (
            Compute & compute,
            Buffer & data,
            Buffer & dispatch,
            uint32_t offset,
            Flags flags = FlagNone )
```

dispatch single in-place indirect local bitonic sort

Parameters

data	Buffer of uint32_t data elements to sort.
dispatch	Dispatch indirect buffer.
offset	Dispatch indirect buffer offset.

5.15.2.5 dispatchIndirect() [2/3]

```
bool Tellusim::BitonicSort::dispatchIndirect (
            Compute & compute,
            Buffer & data,
            uint32_t count,
            Buffer & dispatch,
            uint32_t offset,
            Flags flags = FlagNone )
```

dispatch multiple in-place indirect local bitonic sorts

Parameters

data	Buffer of uint32_t data elements to sort.
count	Number of regions to sort.
dispatch	Dispatch indirect buffer.
offset	Dispatch indirect buffer offset.

5.15.2.6 dispatchIndirect() [3/3]

```
bool Tellusim::BitonicSort::dispatchIndirect (
            Compute & compute,
            Buffer & data,
            Buffer & count,
            Buffer & dispatch,
            uint32_t count_offset,
            uint32_t dispatch_offset,
            Flags flags = FlagNone )
```

dispatch multiple in-place indirect local bitonic sorts

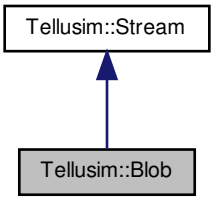
Parameters

data	Buffer of uint32_t data elements to sort.
count	Count indirect buffer.
dispatch	Dispatch indirect buffer.
count_offset	Count indirect buffer offset.
dispatch_offset	Dispatch indirect buffer offset.

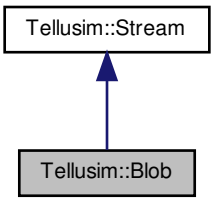
5.16 Tellusim::Blob Class Reference

```
#include <core/TellusimBlob.h>
```

Inheritance diagram for Tellusim::Blob:



Collaboration diagram for Tellusim::Blob:



Public Member Functions

- **Blob** (const char *name=nullptr)
- **Blob** (const String &name)
- **Blob** (const uint8_t *data, size_t size, const char *name=nullptr)
- **Blob** (const uint8_t(*blob)[256], const char *name=nullptr)
- **Blob** (const Blob &blob, bool move)
- void release ()

  clear blob
- void **clear** ()
- void setName (const char *name)

  blob name
- void **setName** (const String &name)
- void setSize (size_t size)

  blob data
- void **setCapacity** (size_t size)
- size_t **getCapacity** () const
- bool **setData** (const uint8_t *data, size_t size)
- bool **setData** (const uint8_t(*blob)[256])
- bool **setData** (const Blob &blob)
- const uint8_t * **getData** () const
- uint8_t * **getData** ()
- String encodeBase64 (size_t size=0)

  base64 encoding
- bool **decodeBase64** (const char *src)
- void getMD5 (uint32_t hash[4], size_t size=0)

  message digest algorithm
- String **getMD5** (size_t size=0)
- void getSHA1 (uint32_t hash[5], size_t size=0)

  secure hash algorithm
- String **getSHA1** (size_t size=0)

Static Public Member Functions

- static String **getMD5** (const String &str)
- static String **getMD5** (const void *src, size_t size)
- static String **getMD5** (Stream &src, size_t size=0)
- static String **getSHA1** (const String &str)
- static String **getSHA1** (const void *src, size_t size)
- static String **getSHA1** (Stream &src, size_t size=0)

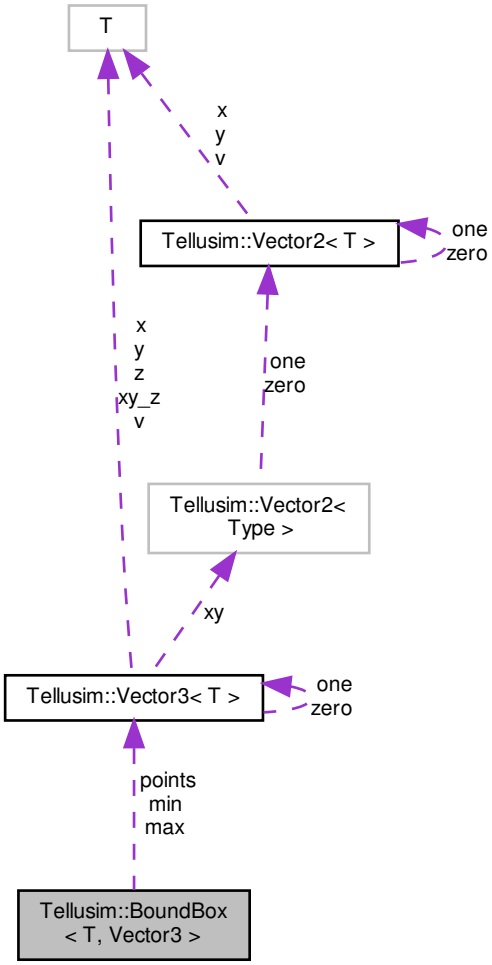
5.16.1 Detailed Description

The Blob class extends the Stream class and provides a flexible way to manage binary data located in the system memory. The class also includes functions for base64 encoding and decoding, as well as calculating message digests using MD5 and SHA1 hash algorithms.

5.17 Tellusim::BoundBox< T, Vector3 > Struct Template Reference

`#include <geometry/TellusimBounds.h>`

Collaboration diagram for Tellusim::BoundBox< T, Vector3 >:



Public Types

- using **Vector** = Vector3
- using **Vector2** = Tellusim::Vector2< Type >
- using **BoundSphere** = Tellusim::BoundSphere< Type, Vector3 >

Public Member Functions

- **BoundBox** (const BoundBox &bb)
- **BoundBox** (const Vector3 &bb_min, const Vector3 &bb_max)
- **BoundBox** (const Vector3 &bs_center, Type bs_radius)
- **BoundBox** (const BoundBox &bb, const BoundSphere &bs)
- **BoundBox** (const BoundSphere &bs)
- template<class CType , class CVector >
  BoundBox (const Tellusim::BoundBox< CType, CVector > &bb)
- template<class CType , class CVector >
  BoundBox (const Tellusim::BoundSphere< CType, CVector > &bs)
- **BoundBox** (const Vector3 *1 points, uint32_t num_points)
- void clear ()

  clear bound box
- bool isValid () const

  check bound box
- **operator bool** () const
- void set (const Vector3 &bb_min, const Vector3 &bb_max)

  set bound box
- void **set** (const BoundBox &bb)
- void set (const Vector3 &bs_center, Type bs_radius)

  set bound sphere
- void **set** (const BoundSphere &bs)
- void set (const Vector3 &bb_min, const Vector3 &bb_max, const Vector3 &bs_center, Type bs_radius)

  set minimal bound box
- void **set** (const BoundBox &bb, const BoundSphere &bs)
- void set (const Vector3 *1 points, uint32_t num_points)

  set bound box
- void expand (const Vector3 &point)

  expand by point
- void expand (const Vector3 &bb_min, const Vector3 &bb_max)

  expand by bound box
- void **expand** (const BoundBox &bb)
- void expand (const Vector3 &bs_center, Type bs_radius)

  expand by bound sphere
- void **expand** (const BoundSphere &bs)
- void expand (const Vector3 &bb_min, const Vector3 &bb_max, const Vector3 &bs_center, Type bs_radius)

  expand by minimal bounds
- void **expand** (const BoundBox &bb, const BoundSphere &bs)
- void shrink (const Vector3 &bb_min, const Vector3 &bb_max)

  shrink by bound box
- void **shrink** (const BoundBox &bb)
- void shrink (const Vector3 &bs_center, Type bs_radius)

  shrink by bound sphere
- void **shrink** (const BoundSphere &bs)
- bool inside (const Vector3 &point) const

  inside point
- bool inside (const Vector3 &bb_min, const Vector3 &bb_max) const

  inside bound box
- bool **inside** (const BoundBox &bb) const
- bool inside (const Vector3 &bs_center, Type bs_radius) const

  inside bound sphere

- bool **inside** (const BoundSphere &bs) const
- Type distance (const Vector3 &point) const

  signed distance to the bound box
- Vector2 trace (const Vector3 &point, const Vector3 &idirection) const

  bound box ray intersection
- Vector3 getCenter () const

  to bound sphere
- Type **getRadius** () const
- Vector3 getSize () const

  bound box parameters
- const Vector3 & **getMin** () const
- const Vector3 & **getMax** () const
- const Vector3 * **getPoints** () const
- Type **getVolume** (Type threshold=1e-8f) const

Public Attributes

-
```
union {
  struct {
    Vector3 min
    Vector3 max
  }
  Vector3 points [2]
};
```

5.17.1 Detailed Description

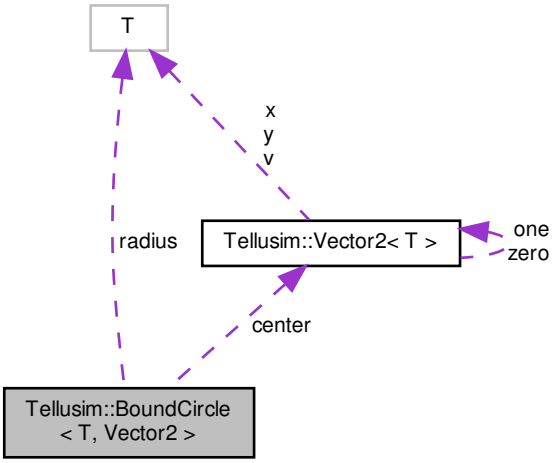
template<class T, class Vector3 = Tellusim::Vector3<T>>
struct Tellusim::BoundBox< T, Vector3 >

BoundBox class

5.18 Tellusim::BoundCircle< T, Vector2 > Struct Template Reference

`#include <geometry/TellusimBounds.h>`

Collaboration diagram for Tellusim::BoundCircle< T, Vector2 >:



Public Types

- using **Vector** = Vector2
- using **BoundRect** = Tellusim::BoundRect< Type, Vector2 >

Public Member Functions

- **BoundCircle** (const BoundCircle &bc)
- **BoundCircle** (const Vector2 &br_min, const Vector2 &br_max)
- **BoundCircle** (const Vector2 &bc_center, Type bc_radius)
- **BoundCircle** (const BoundRect &br)
- template<class CType , class CVector >
  BoundCircle (const Tellusim::BoundRect< CType, CVector > &br)
- template<class CType , class CVector >
  BoundCircle (const Tellusim::BoundCircle< CType, CVector > &bc)
- **BoundCircle** (const Vector2 *1 points, uint32_t num_points)
- void clear ()

  clear bound circle
- bool isValid () const

  check bound circle
- **operator bool** () const
- void set (const Vector2 &br_min, const Vector2 &br_max)

  set bound rect
- void **set** (const BoundRect &br)
- void set (const Vector2 &bc_center, Type bc_radius)

  set bound circle
- void **set** (const BoundCircle &bc)
- void set (const Vector2 *1 points, uint32_t num_points)

  set bound circle

- void expand (const Vector2 &point)
  expand by point
- void expand (const Vector2 &br_min, const Vector2 &br_max)
  expand by bound rect
- void **expand** (const BoundRect &br)
- void expand (const Vector2 &bc_center, Type bc_radius)
  expand by bound circle
- void **expand** (const BoundCircle &bc)
- void expand (const Vector2 &br_min, const Vector2 &br_max, const Vector2 &bc_center, Type bc_radius)
  expand by minimal bounds
- void **expand** (const BoundRect &br, const BoundCircle &bc)
- void expandRadius (const Vector2 &point)
  expand radius by point
- void expandRadius (const Vector2 &br_min, const Vector2 &br_max)
  expand radius by bound rect
- void **expandRadius** (const BoundRect &br)
- void expandRadius (const Vector2 &bc_center, Type bc_radius)
  expand radius by bound circle
- void **expandRadius** (const BoundCircle &bc)
- bool inside (const Vector2 &point) const
  inside point
- bool inside (const Vector2 &br_min, const Vector2 &br_max) const
  inside bound rect
- bool **inside** (const BoundRect &br) const
- bool inside (const Vector2 &bc_center, Type bc_radius) const
  inside bound circle
- bool **inside** (const BoundCircle &bc) const
- Type distance (const Vector2 &point) const
  signed distance to bound circle
- Vector2 getMin () const
  to bound rect
- Vector2 **getMax** () const
- const Vector2 & getCenter () const
  bound circle parameters
- Type **getRadius** () const
- Type **getArea** () const

Public Attributes

- Vector2 **center**
- Type **radius**

5.18.1 Detailed Description

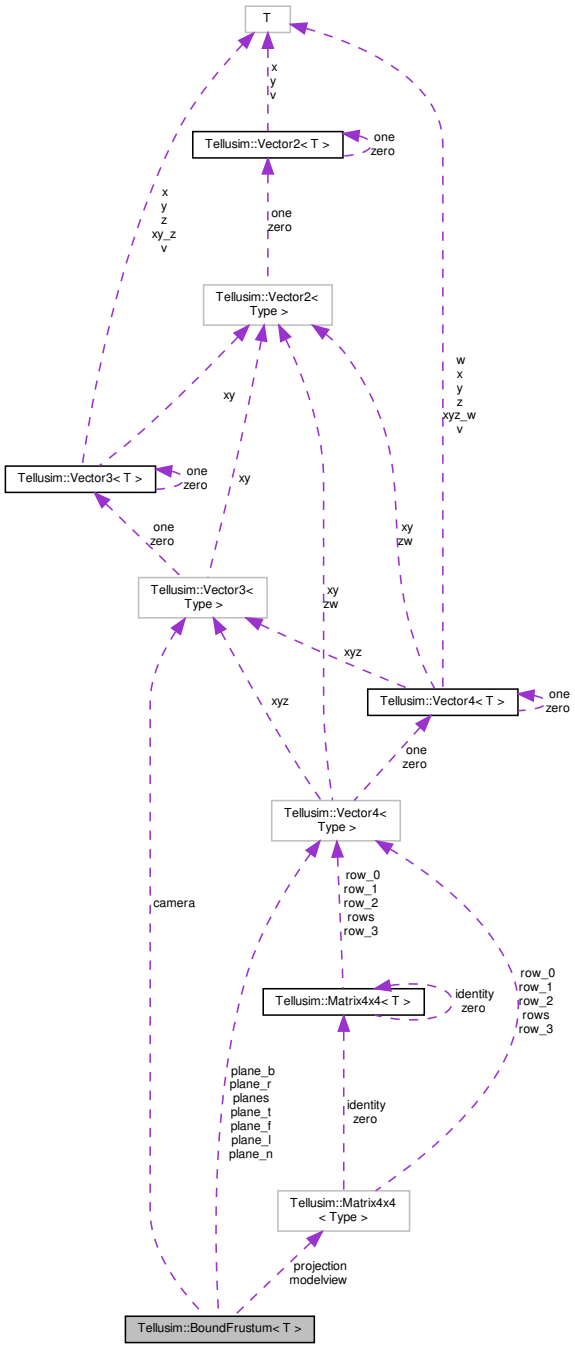
template<class T, class Vector2 = Tellusim::Vector2<T>>
struct Tellusim::BoundCircle< T, Vector2 >

BoundCircle class

5.19 Tellusim::BoundFrustum< T > Struct Template Reference

`#include <geometry/TellusimBounds.h>`

Collaboration diagram for Tellusim::BoundFrustum< T >:



Public Types

- using **Vector3** = Tellusim::Vector3< Type >
- using **Vector4** = Tellusim::Vector4< Type >
- using **Matrix4x4** = Tellusim::Matrix4x4< Type >

Public Member Functions

- **BoundFrustum** (const Matrix4x4 &projection, const Matrix4x4 &modelview, Type aspect=1.0f)
- **BoundFrustum** (const BoundFrustum &bf)
- template<class CType >
  BoundFrustum (const BoundFrustum< CType > &bf)
- void set (const Matrix4x4 &p, const Matrix4x4 &m, Type aspect=1.0f)

  bound frustum from matrix
- template<class BType , class BVector >
  bool inside4 (const BoundBox< BType, BVector > &bb) const

  inside bound box
- template<class BType , class BVector >
  bool **inside** (const BoundBox< BType, BVector > &bb) const
- template<class BType , class BVector >
  bool **insideAll4** (const BoundBox< BType, BVector > &bb) const
- template<class BType , class BVector >
  bool **insideAll** (const BoundBox< BType, BVector > &bb) const
- template<class BType , class BVector >
  bool inside4 (const BoundSphere< BType, BVector > &bs) const

  inside bound sphere
- template<class BType , class BVector >
  bool **inside** (const BoundSphere< BType, BVector > &bs) const
- template<class BType , class BVector >
  bool **insideAll4** (const BoundSphere< BType, BVector > &bs) const
- template<class BType , class BVector >
  bool **insideAll** (const BoundSphere< BType, BVector > &bs) const
- const Matrix4x4 & getProjection () const

  bound frustum parameters
- const Matrix4x4 & **getModelview** () const
- const Vector3 & **getCamera** () const

Public Attributes

- Matrix4x4 **projection**
- Matrix4x4 **modelview**
- Vector3 **camera**
-

```
union {
  struct {
    Vector4 plane_l
    Vector4 plane_r
    Vector4 plane_b
    Vector4 plane_t
    Vector4 plane_n
    Vector4 plane_f
  }
  Vector4 planes [6]
};
```

-

```
union {
  struct {
    uint8_t sign_l [4]
    uint8_t sign_r [4]
    uint8_t sign_b [4]
    uint8_t sign_t [4]
    uint8_t sign_n [4]
    uint8_t sign_f [4]
  }
  uint8_t signs [6][4]
};
```

5.19.1 Detailed Description

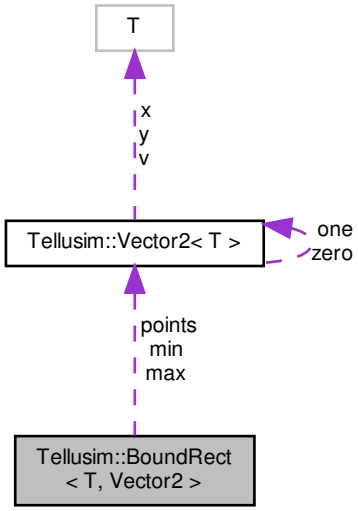
template<**class T**>
struct Tellusim::BoundFrustum< **T** >

BoundFrustum class

5.20 Tellusim::BoundRect< T, Vector2 > Struct Template Reference

`#include <geometry/TellusimBounds.h>`

Collaboration diagram for Tellusim::BoundRect< T, Vector2 >:



Public Types

- using **Vector** = Vector2
- using **BoundCircle** = Tellusim::BoundCircle< Type, Vector2 >

Public Member Functions

- **BoundRect** (const BoundRect &br)
- **BoundRect** (const Vector2 &br_min, const Vector2 &br_max)
- **BoundRect** (const Vector2 &bc_center, Type bc_radius)
- **BoundRect** (const BoundRect &br, const BoundCircle &bc)
- **BoundRect** (const BoundCircle &bc)
- template<class CType , class CVector >
  BoundRect (const Tellusim::BoundRect< CType, CVector > &br)
- template<class CType , class CVector >
  BoundRect (const Tellusim::BoundCircle< CType, CVector > &bc)
- **BoundRect** (const Vector2 *1 points, uint32_t num_points)
- void clear ()

  clear bound rect
- bool isValid () const

  check bound rect
- **operator bool** () const
- void set (const Vector2 &br_min, const Vector2 &br_max)

  set bound rect
- void **set** (const BoundRect &br)
- void set (const Vector2 &bc_center, Type bc_radius)

  set bound circle
- void **set** (const BoundCircle &bc)
- void set (const Vector2 &br_min, const Vector2 &br_max, const Vector2 &bc_center, const Type &bc_radius)

  set minimal bound rect
- void **set** (const BoundRect &br, const BoundCircle &bc)
- void set (const Vector2 *1 points, uint32_t num_points)

  set bound rect
- void expand (const Vector2 &point)

  expand by point
- void expand (const Vector2 &br_min, const Vector2 &br_max)

  expand by bound rect
- void **expand** (const BoundRect &br)
- void expand (const Vector2 &bc_center, Type bc_radius)

  expand by bound circle
- void **expand** (const BoundCircle &bc)
- bool inside (const Vector2 &point) const

  inside point
- bool inside (const Vector2 &br_min, const Vector2 &br_max) const

  inside bound rect
- bool **inside** (const BoundRect &br) const
- bool inside (const Vector2 &bc_center, Type bc_radius) const

  inside bound circle
- bool **inside** (const BoundCircle &bc) const
- Type distance (const Vector2 &point) const

  signed distance to the bound rect
- Vector2 trace (const Vector2 &point, const Vector2 &idirection) const

  bound rect ray intersection
- Vector2 getCenter () const

  to bound circle
- Type **getRadius** () const
- Vector2 getSize () const

bound rect parameters

- const Vector2 & **getMin** () const
- const Vector2 & **getMax** () const
- const Vector2 * **getPoints** () const
- Type **getArea** (Type threshold=1e-8f) const

Public Attributes

-

union {
  struct {
    Vector2 **min**
    Vector2 **max**
  }
  Vector2 **points** [2]
};

5.20.1 Detailed Description

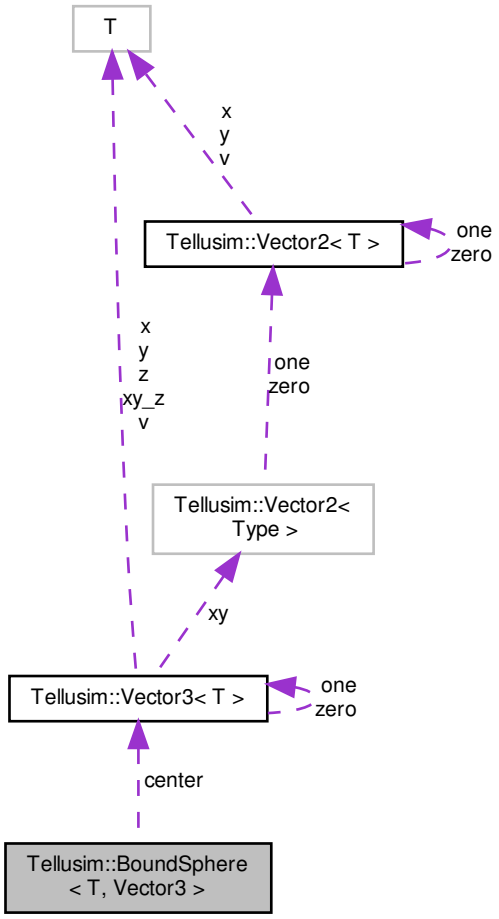
template<class T, class Vector2 = Tellusim::Vector2<T>>
struct Tellusim::BoundRect< T, Vector2 >

BoundRect class

5.21 Tellusim::BoundSphere< T, Vector3 > Struct Template Reference

`#include <geometry/TellusimBounds.h>`

Collaboration diagram for Tellusim::BoundSphere< T, Vector3 >:



Public Types

- using **Vector** = Vector3
- using **BoundBox** = Tellusim::BoundBox< Type, Vector3 >

Public Member Functions

- **BoundSphere** (const BoundSphere &bs)
- **BoundSphere** (const Vector3 &bb_min, const Vector3 &bb_max)
- **BoundSphere** (const Vector3 &bs_center, Type bs_radius)
- **BoundSphere** (const BoundBox &bb)
- template<class CType , class CVector >
  BoundSphere (const Tellusim::BoundBox< CType, CVector > &bb)
- template<class CType , class CVector >
  BoundSphere (const Tellusim::BoundSphere< CType, CVector > &bs)
- **BoundSphere** (const Vector3 *1 points, uint32_t num_points)

- void clear ()

  clear bound sphere
- bool isValid () const

  check bound sphere
- **operator bool** () const
- void set (const Vector3 &bb_min, const Vector3 &bb_max)

  set bound box
- void **set** (const BoundBox &bb)
- void set (const Vector3 &bs_center, Type bs_radius)

  set bound sphere
- void **set** (const BoundSphere &bs)
- void set (const Vector3 *1 points, uint32_t num_points)

  set bound sphere
- void expand (const Vector3 &point)

  expand by point
- void expand (const Vector3 &bb_min, const Vector3 &bb_max)

  expand by bound box
- void **expand** (const BoundBox &bb)
- void expand (const Vector3 &bs_center, Type bs_radius)

  expand by bound sphere
- void **expand** (const BoundSphere &bs)
- void expand (const Vector3 &bb_min, const Vector3 &bb_max, const Vector3 &bs_center, Type bs_radius)

  expand by minimal bounds
- void **expand** (const BoundBox &bb, const BoundSphere &bs)
- void expandRadius (const Vector3 &point)

  expand radius by point
- void expandRadius (const Vector3 &bb_min, const Vector3 &bb_max)

  expand radius by bound box
- void **expandRadius** (const BoundBox &bb)
- void expandRadius (const Vector3 &bs_center, Type bs_radius)

  expand radius by bound sphere
- void **expandRadius** (const BoundSphere &bs)
- bool inside (const Vector3 &point) const

  inside point
- bool inside (const Vector3 &bb_min, const Vector3 &bb_max) const

  inside bound box
- bool **inside** (const BoundBox &bb) const
- bool inside (const Vector3 &bs_center, Type bs_radius) const

  inside bound sphere
- bool **inside** (const BoundSphere &bs) const
- Type distance (const Vector3 &point) const

  signed distance to bound sphere
- Vector3 getMin () const

  to bound box
- Vector3 **getMax** () const
- const Vector3 & getCenter () const

  bound sphere parameters
- Type **getRadius** () const
- Type **getVolume** () const

Public Attributes

- Vector3 **center**
- Type **radius**

5.21.1 Detailed Description

template<class T, class Vector3 = Tellusim::Vector3<T>>
struct Tellusim::BoundSphere< T, Vector3 >

BoundSphere class

5.22 Tellusim::BrepModel Class Reference

```
#include <graphics/TellusimBrepModel.h>
```

Classes

- struct FaceParameters

Public Types

- enum Flags {
  FlagNone = 0,
  FlagVerbose = (1 << 0),
  FlagCurve2 = (1 << 1),
  FlagSurface2 = (1 << 2),
  FlagMaterials = (1 << 3),
  FlagBufferTexture = (1 << 4),
  FlagBufferStorage = (1 << 5),
  FlagBufferTracing = (1 << 6),
  FlagBufferAddress = (1 << 7),
  FlagBufferTexel = (1 << 8),
  DefaultFlags = (FlagVerbose | FlagMaterials | FlagBufferStorage),
  NumFlags = 9 }

  Model flags.
- using Face = BrepFace::Type

  Brep face.

Public Member Functions

- void clear ()
  clear model
- bool isCreated () const
  check model
- Flags getFlags () const
  model flags
- bool **hasFlag** (Flags flags) const
- bool **hasFlags** (Flags flags) const
- bool load (const Device &device, const char *name, Flags flags=DefaultFlags, Async *async=nullptr)
  load model
- bool **load** (const Device &device, Stream &stream, Flags flags=DefaultFlags, Async *async=nullptr)
- bool create (const Device &device, const Brep &brep, Flags flags=DefaultFlags)
  create model
- bool **create** (const Device &device, const BrepGeometry &geometry, Flags flags=DefaultFlags)
- bool **create** (const Device &device, const Array< BrepGeometry > &geometries, Flags flags=DefaultFlags)
- void setBuffers (Command &command, uint32_t index=0, const Pipeline *pipeline=nullptr) const
  set model buffers
- uint32_t getNumVertices () const
  vertices buffer
- Buffer **getVertexBuffer** () const
- Texture **getVertexTexture** () const
- uint32_t getNumIndices () const
  indices buffer
- Buffer **getIndexBuffer** () const
- Texture **getIndexTexture** () const
- uint32_t getNumRanges () const
  ranges buffer
- Buffer **getRangeBuffer** () const
- Texture **getRangeTexture** () const
- uint32_t getNumPrimitives () const
  faces buffer
- Buffer **getFaceBuffer** () const
- Texture **getFaceTexture** () const
- Face getPrimitiveFace (uint32_t index) const
  primitives parameters
- uint32_t **getPrimitiveIndex** (uint32_t index) const
- uint32_t **getPrimitiveGeometry** (uint32_t index) const
- uint32_t **getPrimitiveMaterial** (uint32_t index) const
- uint32_t getBaseFaceMask () const
  primitive masks
- uint32_t **getWrapFaceMask** () const
- bool **hasBaseFace** (Face face) const
- bool **hasWrapFace** (Face face) const
- bool **hasFace** (Face face, bool wrap) const
- uint32_t getNumGeometries () const
  geometries
- uint32_t **getNumGeometryBaseIndices** (uint32_t geometry, Face face) const
- uint32_t **getNumGeometryWrapIndices** (uint32_t geometry, Face face) const
- uint32_t **getNumGeometryIndices** (uint32_t geometry, Face face, bool wrap) const
- uint32_t **getGeometryBaseIndex** (uint32_t geometry, Face face) const
- uint32_t **getGeometryWrapIndex** (uint32_t geometry, Face face) const

- uint32_t **getGeometryIndex** (uint32_t geometry, Face face, bool wrap) const
- uint32_t getNumMaterials (uint32_t geometry) const

  geometry materials
- uint32_t **getNumMaterialBaseIndices** (uint32_t geometry, uint32_t material, Face face) const
- uint32_t **getNumMaterialWrapIndices** (uint32_t geometry, uint32_t material, Face face) const
- uint32_t **getNumMaterialIndices** (uint32_t geometry, uint32_t material, Face face, bool wrap) const
- uint32_t **getMaterialBaseIndex** (uint32_t geometry, uint32_t material, Face face) const
- uint32_t **getMaterialWrapIndex** (uint32_t geometry, uint32_t material, Face face) const
- uint32_t **getMaterialIndex** (uint32_t geometry, uint32_t material, Face face, bool wrap) const
- size_t getMemory () const

  memory usage

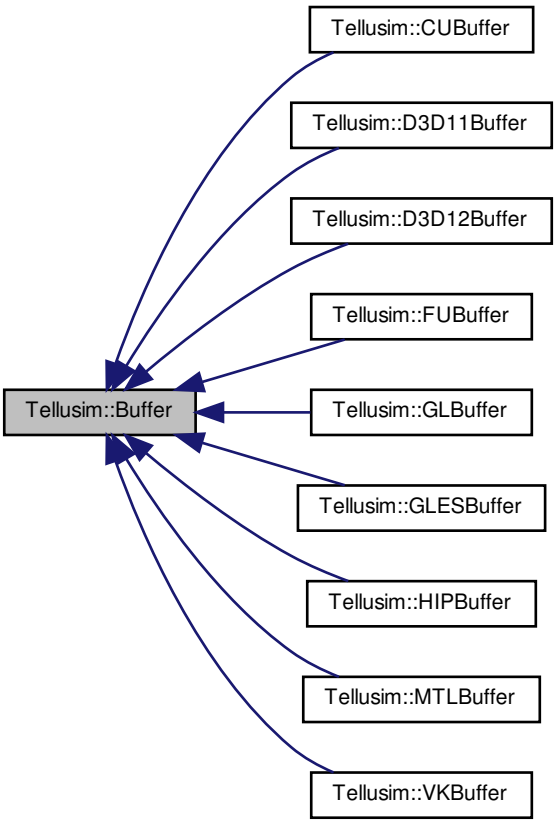
5.22.1 Detailed Description

The BrepModel class represents a GPU-accelerated representation of brep geometry, supporting efficient rendering, material mapping, and geometric analysis. It enables loading or creation of models from brep data with configurable flags for buffer types, material generation, and curve/surface degree limitations.

5.23 Tellusim::Buffer Class Reference

`#include <platform/TellusimBuffer.h>`

Inheritance diagram for Tellusim::Buffer:



Public Types

- enum Flags {
  FlagNone = 0,
  FlagRead = (1 << 0),
  FlagWrite = (1 << 1),
  FlagSource = (1 << 2),
  FlagSparse = (1 << 3),
  FlagShared = (1 << 4),
  FlagMapped = (1 << 5),
  FlagExtern = (1 << 6),
  FlagInterop = (1 << 7),
  FlagDynamic = (1 << 8),
  FlagUniform = (1 << 9),
  FlagStorage = (1 << 10),
  FlagAddress = (1 << 11),
  FlagTracing = (1 << 12),
  FlagScratch = (1 << 13),
  FlagBinding = (1 << 14),
  FlagIndirect = (1 << 15),
  FlagConditional = (1 << 16),
  FlagVertex = (1 << 17),
  FlagIndex = (1 << 18),
  FlagTexel = (1 << 19),
  FlagAccel = (1 << 20),
  DefaultFlags = FlagNone,
  NumFlags = 21 }

  Buffer flags.

Public Member Functions

- Platform getPlatform () const

  buffer platform
- const char * **getPlatformName** () const
- uint32_t getIndex () const

  buffer device index
- void clear ()

  clear buffer
- bool isCreated () const

  check buffer
- void setName (const char *name)

  buffer name
- String **getName** () const
- bool create (Flags flags, size_t size, Format format=FormatUnknown)

  create buffer
- bool **isMapped** () const
- Flags getFlags () const

  buffer flags
- bool **hasFlag** (Flags flags) const
- bool **hasFlags** (Flags flags) const
- String **getFlagsName** () const
- Format getFormat () const

  buffer format
- const char * **getFormatName** () const

- uint32_t **getComponents** () const
- uint32_t **getPixelSize** () const
- size_t getSize ()

  buffer size
- size_t getPageSize ()

  sparse buffer page size
- String getDescription () const

  buffer description

5.23.1 Detailed Description

The Buffer class represents a GPU buffer resource that can be used for a wide range of data storage and access patterns. It supports creation with various flags that define its usage, such as read/write access, dynamic allocation, sharing, mapping, uniform or storage usage, vertex or index roles, and more. The class provides methods to create, clear, and query the buffer state, including platform type, format, size, and associated flags. The Buffer class is a versatile and foundational component for managing memory on the GPU.

5.24 Tellusim::BufferTable Class Reference

```
#include <platform/TellusimBuffer.h>
```

Public Member Functions

- Platform getPlatform () const

  table platform
- const char * **getPlatformName** () const
- uint32_t getIndex () const

  table device index
- void clear ()

  clear table
- bool isCreated () const

  check table
- void setName (const char *name)

  table name
- String **getName** () const
- bool create (uint32_t size)

  create table
- uint32_t getSize () const

  table buffers
- Buffer **get** (uint32_t index) const
- bool **isOwner** (uint32_t index) const
- size_t getMemory () const

  memory usage

5.24.1 Detailed Description

The BufferTable class provides a container for managing multiple buffers with support for bindless resource access. It enables more efficient rendering and compute operations by reducing the overhead associated with traditional binding.

5.25 Tellusim::Tracing::BuildIndirect Struct Reference

build indirect parameters

`#include <platform/TellusimTracing.h>`

Public Attributes

- uint32_t **num_primitives**
- uint32_t **base_primitive**
- uint32_t **base_vertex**
- uint32_t **base_transform**

5.25.1 Detailed Description

build indirect parameters

5.26 Tellusim::Canvas Class Reference

`#include <interface/TellusimCanvas.h>`

Public Types

- using CreateCallback = Function< bool(const Device device, Canvas canvas, uint32_t scale)>
- using PipelineCallback = Function< bool(Pipeline pipeline, Canvas canvas, CanvasElement element)>
- using BeginCallback = Function< bool(Command command, Canvas canvas)>
- using DrawCallback = Function< bool(Command command, Canvas canvas)>

Public Member Functions

- **Canvas** (Canvas *parent)
- void clear ()
  clear canvas
- bool isCreated () const
  check canvas
- uint32_t getScale (const Target &target, uint32_t scale=100) const
  canvas scale
- bool create (const Device &device, Format color, Format depth, uint32_t multisample=1, uint32_t scale=0)
  create canvas
- bool **create** (const Device &device, const Target &target, uint32_t scale=0)
- void setPipelineHash (uint32_t hash)
  pipeline hash
- uint32_t **getPipelineHash** () const
- Format getColorFormat () const
  canvas parameters
- Format **getDepthFormat** () const
- uint32_t **getMultisample** () const
- void setOrder (int32_t order)

canvas order

- int32_t **getOrder** () const
- void setEnabled (bool enabled)

canvas enabled flag

- bool **isEnabled** () const
- void setViewport (const Viewport &viewport)

canvas viewport

- void **setViewport** (uint32_t width, uint32_t height)
- void **setViewport** (float32_t width, float32_t height)
- const Viewport & **getViewport** () const
- float32_t **getWidth** () const
- float32_t **getHeight** () const
- void clearColor ()

canvas color

- void **setColor** (const Color &color)
- void **setColor** (float32_t r, float32_t g, float32_t b, float32_t a)
- const Color & **getColor** () const
- void clearScissor ()

canvas scissor

- void **setScissor** (const Rect &scissor)
- const Rect & **getScissor** () const
- void clearTransform ()

canvas transform

- void **setTransform** (const Matrix4x4f &transform)
- const Matrix4x4f & **getTransform** () const
- uint32_t setParent (Canvas &parent)

canvas parent

- const Canvas **getParent** () const
- Canvas **getParent** ()
- uint32_t addChild (Canvas &child)

canvas children

- bool **removeChild** (Canvas &child)
- bool **raiseChild** (Canvas &child)
- bool **lowerChild** (Canvas &child)
- void **releaseChildren** ()
- uint32_t **findChild** (const Canvas &child) const
- bool **isChild** (const Canvas &child) const
- uint32_t **getNumChildren** () const
- const Array< Canvas > **getChildren** () const
- Array< Canvas > **getChildren** ()
- const Canvas **getChild** (uint32_t index) const
- Canvas **getChild** (uint32_t index)
- uint32_t addElement (CanvasElement &element)

canvas elements

- bool **removeElement** (CanvasElement &element)
- bool **raiseElement** (CanvasElement &element)
- bool **lowerElement** (CanvasElement &element)
- uint32_t **findElement** (const CanvasElement &element) const
- bool **isElement** (const CanvasElement &element) const
- uint32_t **getNumElements** () const
- const Array< CanvasElement > **getElements** () const
- Array< CanvasElement > **getElements** ()
- const CanvasElement **getElement** (uint32_t index) const

- CanvasElement **getElement** (uint32_t index)
- bool isFont (const char ∗name) const

  canvas fonts
- bool **addFont** (const char ∗name, Stream &stream)
- bool **addFont** (const char ∗name, const uint8_t(∗blob)[256])
- void **removeFont** (const char ∗name)
- Font **getFont** (const char ∗name)
- bool isTexture (const char ∗name) const

  canvas textures
- bool **addTexture** (const char ∗name, Stream &stream)
- bool **addTexture** (const char ∗name, Texture &texture)
- bool **addTexture** (const char ∗name, const uint8_t(∗blob)[256])
- void **removeTexture** (const char ∗name)
- Texture **getTexture** (const char ∗name)
- void setDepthMask (Pipeline::DepthMask mask)

  depth parameters
- Pipeline::DepthMask **getDepthMask** () const
- void **setDepthFunc** (Pipeline::DepthFunc func)
- Pipeline::DepthFunc **getDepthFunc** () const
- void draw (Command &command, const Target &target)

  draw canvas
- void **draw** (Command &command)
- void **setCreateCallback** (const CreateCallback &func)
- CreateCallback **getCreateCallback** () const
- void **setPipelineCallback** (const PipelineCallback &func)
- PipelineCallback **getPipelineCallback** () const
- void **setBeginCallback** (const BeginCallback &func)
- BeginCallback **getBeginCallback** () const
- void **setDrawCallback** (const DrawCallback &func)
- DrawCallback **getDrawCallback** () const
- uint32_t getNumDrawPipelines () const

  draw statistics
- uint32_t **getNumDrawElements** () const
- uint32_t **getNumDrawCommands** () const
- Rect getRect () const

  canvas rectangle

5.26.1 Detailed Description

The Canvas class represents a graphical canvas in a rendering system, providing methods for managing its state, rendering, transformations, and hierarchy. It allows multiple canvases to be combined, offering significant flexibility in how canvases and their elements are composed and rendered together in a graphical system.

5.26.2 Member Typedef Documentation

5.26.2.1 CreateCallback

```
using Tellusim::Canvas::CreateCallback = Function<bool(const Device device, Canvas canvas,
uint32_t scale)>
```

create callback

Parameters

func	Function that runs on Canvas::create() call. Returning false prevents Canvas creation.

5.26.2.2 PipelineCallback

```
using Tellusim::Canvas::PipelineCallback = Function<bool(Pipeline pipeline, Canvas canvas,
CanvasElement element)>
```

pipeline callback

Parameters

func	Function that runs on CanvasElement::create() call. Returning false prevents Pipeline creation, so the Pipeline can be completely replaced.

5.26.2.3 BeginCallback

```
using Tellusim::Canvas::BeginCallback = Function<bool(Command command, Canvas canvas)>
```

begin callback

Parameters

func	Function that runs on Canvas::draw() call. Returning false prevents Canvas draw.

5.26.2.4 DrawCallback

```
using Tellusim::Canvas::DrawCallback = Function<bool(Command command, Canvas canvas)>
```

draw callback

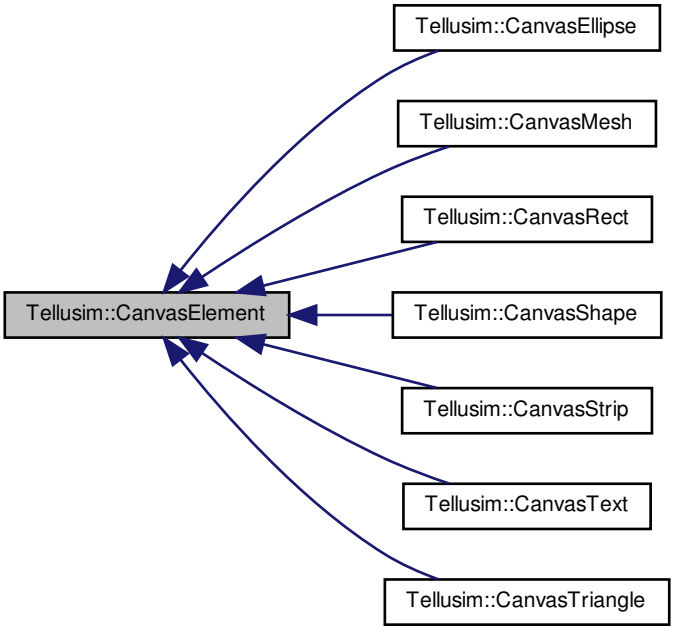
Parameters

func	Function that runs on each CanvasElement group draw call of Canvas::draw().

5.27 Tellusim::CanvasElement Class Reference

```
#include <interface/TellusimCanvas.h>
```

Inheritance diagram for Tellusim::CanvasElement:



Public Types

- enum Mode {
  ModeSolid = 0,
  ModeTexture,
  ModeTextureFetch,
  ModeTextureClamp,
  ModeTextureCubic,
  ModeTextureCubic3x3,
  ModeTextureCubic5x5,
  ModeTextureRed,
  ModeTextureGreen,
  ModeTextureBlue,
  ModeTextureAlpha,
  ModeGradient,
  NumModes }

  Element modes.
- enum Align {
  AlignNone = 0,
  AlignLeft = (1 << 0),
  AlignRight = (1 << 1),
  AlignBottom = (1 << 2),
  AlignTop = (1 << 3),
  AlignCenterX = (1 << 4),
  AlignCenterY = (1 << 5),
  AlignLeftBottom = (AlignLeft | AlignBottom),
  AlignLeftTop = (AlignLeft | AlignTop),

AlignRightBottom = (AlignRight | AlignBottom),
AlignRightTop = (AlignRight | AlignTop),
AlignCenter = (AlignCenterX | AlignCenterY),
NumAligns = 6 }

Element alignments.

- enum Stack {
StackNone = 0,
StackPush = (1 << 0),
StackPop = (1 << 1),
StackSet = (1 << 2),
StackMul = (1 << 3),
StackGet = (1 << 4) }

Element stack operations.

- using DrawCallback = Function< bool(Command command, CanvasElement element)>

Public Member Functions

- Type getType () const

element type

- const char * **getTypeName** () const
- bool **isText** () const
- bool **isMesh** () const
- bool **isRect** () const
- bool **isTriangle** () const
- bool **isEllipse** () const
- bool **isShape** () const
- bool **isStrip** () const
- void setCanvas (Canvas &canvas)

element canvas

- const Canvas **getCanvas** () const
- Canvas **getCanvas** ()
- void setMode (Mode mode)

element mode

- Mode **getMode** () const
- void setAlign (Align align)

element align

- Align **getAlign** () const
- bool **hasAlign** (Align align) const
- bool **hasAligns** (Align aligns) const
- void setOrder (int32_t order)

element order

- int32_t **getOrder** () const
- void setEnabled (bool enabled)

element enabled flag

- bool **isEnabled** () const
- void clearColor ()

element color

- void **setColor** (Stack op)
- void **setColor** (const Color &color, Stack op=StackNone)
- void **setColor** (float32_t r, float32_t g, float32_t b, float32_t a, Stack op=StackNone)
- const Color & **getColor** () const
- Stack **getColorOp** () const
- void clearTransform ()

element transform

- void **setTransform** (Stack op)
- void **setTransform** (const Matrix4x4f &transform, Stack op=StackNone)
- const Matrix4x4f & **getTransform** () const
- Stack **getTransformOp** () const
- void clearScissor ()

element scissor

- void **setScissor** (Stack op)
- void **setScissor** (const Rect &scissor, Stack op=StackNone)
- const Rect & **getScissor** () const
- Stack **getScissorOp** () const
- void setMipmap (float32_t mipmap)

element mipmap number

- float32_t **getMipmap** () const
- void setSampler (Sampler &sampler)

sampler pointer

- Sampler **getSampler** () const
- void setFilter (Sampler::Filter filter)

filter mode

- Sampler::Filter **getFilter** () const
- void **setAnisotropy** (uint32_t anisotropy)
- uint32_t **getAnisotropy** () const
- void setWrapMode (Sampler::WrapMode mode)

wrapping mode

- Sampler::WrapMode **getWrapMode** () const
- void setTexture (Texture &texture, bool linear=false)

texture pointer

- Texture **getTexture** () const
- bool **getTextureLinear** () const
- void setPipeline (Pipeline pipeline)

pipeline pointer

- Pipeline **getPipeline** () const
- void setPrimitive (Pipeline::Primitive primitive)

rasterization parameters

- Pipeline::Primitive **getPrimitive** () const
- void **setCullMode** (Pipeline::CullMode mode)
- Pipeline::CullMode **getCullMode** () const
- void **setFrontMode** (Pipeline::FrontMode mode)
- Pipeline::FrontMode **getFrontMode** () const
- void setBlend (Pipeline::BlendOp op, Pipeline::BlendFunc src, Pipeline::BlendFunc dest)

blending parameters

- Pipeline::BlendOp **getBlendOp** () const
- Pipeline::BlendFunc **getBlendSrcFunc** () const
- Pipeline::BlendFunc **getBlendDestFunc** () const
- void setColorMask (Pipeline::ColorMask mask)

color parameters

- Pipeline::ColorMask **getColorMask** () const
- void setDepthMask (Pipeline::DepthMask mask)

depth parameters

- Pipeline::DepthMask **getDepthMask** () const
- void **setDepthFunc** (Pipeline::DepthFunc func)
- Pipeline::DepthFunc **getDepthFunc** () const
- void setStencilRef (uint32_t ref)

stencil parameters

- void **setStencilFunc** (Pipeline::StencilFunc func, Pipeline::StencilOp fail_op, Pipeline::StencilOp dfail_op, Pipeline::StencilOp dpass_op)
- uint32_t **getStencilRef** () const
- Pipeline::StencilFunc **getStencilFunc** () const
- Pipeline::StencilOp **getStencilFailOp** () const
- Pipeline::StencilOp **getStencilDepthFailOp** () const
- Pipeline::StencilOp **getStencilDepthPassOp** () const
- void **setDrawCallback** (const DrawCallback &func)
- DrawCallback **getDrawCallback** () const
- const Rect & getRect ()

  element rectangle

Static Public Member Functions

- static const char * **getTypeName** (Type type)

Friends

- class **Canvas**

5.27.1 Detailed Description

The CanvasElement class represents a drawable element within a Canvas and serves as a base for various types such as text, shapes, and meshes, offering customizable rendering modes, alignments, and transformations. It provides fine-grained control over visual attributes including color, texture, depth, and blending, and supports stack-based operations for managing transform, color, and scissor states. Each element can be linked to a dedicated rendering pipeline, sampler, and draw callback, enabling flexible and precise control over its appearance and rendering behavior.

5.27.2 Member Typedef Documentation

5.27.2.1 DrawCallback

```
using Tellusim::CanvasElement::DrawCallback = Function<bool(Command command, CanvasElement element)>
```

draw callback

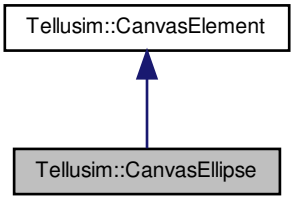
Parameters

func	Function that runs on Canvas::draw() call. Returning false prevents CanvasElement draw.

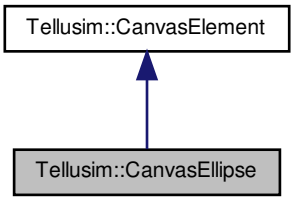
5.28 Tellusim::CanvasEllipse Class Reference

```
#include <interface/TellusimCanvas.h>
```

Inheritance diagram for Tellusim::CanvasEllipse:



Collaboration diagram for Tellusim::CanvasEllipse:



Public Member Functions

- **CanvasEllipse** (Canvas &canvas)
- **CanvasEllipse** (Canvas &canvas, float32_t radius)
- void setRadius (float32_t radius)

  ellipse radius
- float32_t **getRadius** () const
- void setTextureName (const char *name)

  texture name
- void **setTextureName** (const String &name)
- String **getTextureName** () const
- void setStrokeColor (const Color &color)

  stroke color
- const Color & **getStrokeColor** () const
- void setStrokeStyle (const StrokeStyle &style)

  stroke style
- const StrokeStyle & **getStrokeStyleConst** () const
- const StrokeStyle & **getStrokeStyle** () const
- StrokeStyle & **getStrokeStyle** ()
- void setGradientStyle (const GradientStyle &style)

  gradient style

- const GradientStyle & **getGradientStyleConst** () const
- const GradientStyle & **getGradientStyle** () const
- GradientStyle & **getGradientStyle** ()
- void setPosition (const Vector3f &position)

  ellipse positions
- void **setPosition0** (const Vector3f &position)
- void **setPosition1** (const Vector3f &position)
- void **setPosition** (float32_t x, float32_t y, float32_t z=0.0f)
- void **setPosition0** (float32_t x, float32_t y, float32_t z=0.0f)
- void **setPosition1** (float32_t x, float32_t y, float32_t z=0.0f)
- void **setPosition** (const Vector3f &position_0, const Vector3f &position_1)
- const Vector3f & **getPosition0** () const
- const Vector3f & **getPosition1** () const
- void setTexCoord (const Rect &texcoord)

  texture coordinates
- void **setTexCoord** (float32_t left, float32_t right, float32_t bottom, float32_t top)
- const Rect & **getTexCoord** () const

Additional Inherited Members

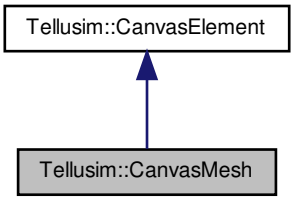
5.28.1 Detailed Description

The CanvasEllipse class defines an elliptical CanvasElement that can be positioned and rendered with adjustable visual properties such as texture, stroke color, stroke style, and gradient style. It allows specifying the ellipse through two bounding positions and an optional radius, enabling the creation of both circles and stretched ellipses. Texture coordinates can be mapped to the ellipse area for detailed control over appearance, making this element suitable for decorative shapes, highlights, and graphical effects in a canvas.

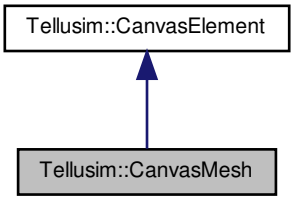
5.29 Tellusim::CanvasMesh Class Reference

```
#include <interface/TellusimCanvas.h>
```

Inheritance diagram for Tellusim::CanvasMesh:



Collaboration diagram for Tellusim::CanvasMesh:



Public Member Functions

- **CanvasMesh** (Canvas &canvas)
- **CanvasMesh** (Canvas &canvas, Mode mode)
- void setTextureName (const char *name)

  texture name
- void **setTextureName** (const String &name)
- String **getTextureName** () const
- void setGradientStyle (const GradientStyle &style)

  gradient style
- const GradientStyle & **getGradientStyleConst** () const
- const GradientStyle & **getGradientStyle** () const
- GradientStyle & **getGradientStyle** ()
- void clearVertices ()

  mesh vertices
- void **setNumVertices** (uint32_t num_vertices)
- void **reserveVertices** (uint32_t num_vertices)
- uint32_t **getNumVertices** () const
- void **setVertices** (const CanvasVertex *vertices, uint32_t num_vertices)
- void **addVertices** (const CanvasVertex *vertices, uint32_t num_vertices)
- const CanvasVertex * **getVertices** () const
- CanvasVertex * **getVertices** ()
- void **setVertex** (uint32_t index, const CanvasVertex &vertex)
- const CanvasVertex & **getVertex** (uint32_t index) const
- CanvasVertex & **getVertex** (uint32_t index)
- void setVertexPosition (uint32_t index, const Vector3f &position)

  vertex positions
- void **setVertexPosition** (uint32_t index, float32_t x, float32_t y, float32_t z=0.0f)
- Vector3f **getVertexPosition** (uint32_t index) const
- void setVertexTexCoord (uint32_t index, const Vector2f &texcoord)

  vertex texture coordinates
- void **setVertexTexCoord** (uint32_t index, float32_t s, float32_t t)
- Vector2f **getVertexTexCoord** (uint32_t index) const
- void setVertexColor (uint32_t index, const Color &color)

  vertex colors
- void **setVertexColor** (uint32_t index, uint32_t color)
- uint32_t **getVertexColor** (uint32_t index) const

- uint32_t addVertex (const Vector3f &position)

  add mesh vertex
- uint32_t **addVertex** (const Vector3f &position, uint32_t color)
- uint32_t **addVertex** (const Vector3f &position, const Vector2f &texcoord)
- uint32_t **addVertex** (const Vector3f &position, const Vector2f &texcoord, uint32_t color)
- uint32_t **addVertex** (float32_t x, float32_t y, float32_t z, float32_t s, float32_t t, uint32_t color=0xffffffffu)
- uint32_t **addVertex** (float32_t x, float32_t y, float32_t z, uint32_t color=0xffffffffu)
- uint32_t **addVertex** (float32_t x, float32_t y, uint32_t color=0xffffffffu)
- void clearIndices ()

  mesh indices
- void **setNumIndices** (uint32_t num_indices)
- void **reserveIndices** (uint32_t num_indices)
- uint32_t **getNumIndices** () const
- void **setIndices** (const uint32_t *indices, uint32_t num_indices)
- void **addIndices** (const uint32_t *indices, uint32_t num_indices)
- const uint32_t * **getIndices** () const
- uint32_t * **getIndices** ()
- void **setIndex** (uint32_t index, uint32_t value)
- uint32_t **getIndex** (uint32_t index) const
- void addIndex (uint32_t i0)

  add mesh indices
- void **addIndices** (uint32_t i0, uint32_t i1)
- void **addIndices** (uint32_t i0, uint32_t i1, uint32_t i2)
- void **addIndices** (uint32_t i0, uint32_t i1, uint32_t i2, uint32_t i3)
- void setRect (const Rect &rect)

  mesh rectangle

Additional Inherited Members

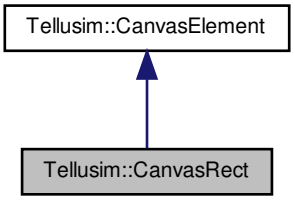
5.29.1 Detailed Description

The CanvasMesh class is a specialized CanvasElement that represents a custom drawable mesh composed of vertices and indices, allowing precise control over geometry, texture mapping, and color per vertex. It supports setting vertex positions, texture coordinates, and colors, as well as managing index buffers for defining primitive connectivity. The class enables advanced rendering through texture names and gradient styles, and provides methods for dynamic mesh construction, such as reserving space, adding vertices and indices, and modifying individual vertex attributes. This makes it suitable for rendering complex shapes and textured surfaces within a Canvas

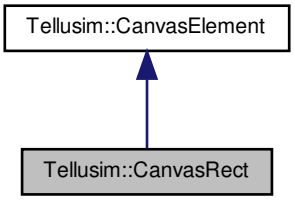
5.30 Tellusim::CanvasRect Class Reference

```
#include <interface/TellusimCanvas.h>
```

Inheritance diagram for Tellusim::CanvasRect:



Collaboration diagram for Tellusim::CanvasRect:



Public Member Functions

- **CanvasRect** (Canvas &canvas)
- **CanvasRect** (Canvas &canvas, float32_t radius)
- **CanvasRect** (Canvas &canvas, float32_t radius, const Vector2f &size)
- void setRadius (float32_t radius)

  rect radius
- float32_t **getRadius** () const
- void setTextureName (const char *name)

  texture name
- void **setTextureName** (const String &name)
- String **getTextureName** () const
- void setStrokeColor (const Color &color)

  stroke color
- const Color & **getStrokeColor** ()
- void setStrokeStyle (const StrokeStyle &style)

  stroke style
- const StrokeStyle & **getStrokeStyleConst** () const
- const StrokeStyle & **getStrokeStyle** () const
- StrokeStyle & **getStrokeStyle** ()
- void setGradientStyle (const GradientStyle &style)

gradient style

- const GradientStyle & **getGradientStyleConst** () const
- const GradientStyle & **getGradientStyle** () const
- GradientStyle & **getGradientStyle** ()
- void setSize (const Vector2f &size)

  rect size
- void **setSize** (float32_t width, float32_t height)
- const Vector2f & **getSize** () const
- float32_t **getWidth** () const
- float32_t **getHeight** () const
- void setPosition (const Vector3f &position)

  rect position
- void **setPosition** (float32_t x, float32_t y, float32_t z=0.0f)
- const Vector3f & **getPosition** () const
- void setTexCoord (const Rect &texcoord)

  texture coordinates
- void **setTexCoord** (float32_t left, float32_t right, float32_t bottom, float32_t top)
- const Rect & **getTexCoord** () const

Additional Inherited Members

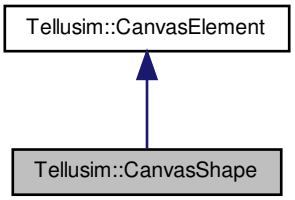
5.30.1 Detailed Description

The CanvasRect class is a rectangular CanvasElement that supports rendering with customizable size, position, corner radius, texture mapping, stroke, and gradient styles. It provides methods for setting dimensions, rounded corners, and 2D or 3D placement, as well as applying textures via name and adjusting texture coordinates. Additionally, it enables visual enhancements such as stroke color and style and gradient fills, making it suitable for rendering styled UI panels, buttons, or other rectangular graphics in a canvas-based rendering system.

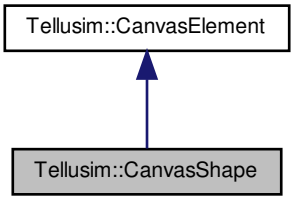
5.31 Tellusim::CanvasShape Class Reference

`#include <interface/TellusimCanvas.h>`

Inheritance diagram for Tellusim::CanvasShape:



Collaboration diagram for Tellusim::CanvasShape:



Public Member Functions

- **CanvasShape** (Canvas &canvas)
- **CanvasShape** (Canvas &canvas, bool cubic)
- void setCubic (bool cubic)

  cubic flag
- bool **isCubic** () const
- void setThreshold (float32_t threshold)

  cubic to quadratic threshold
- float32_t **getThreshold** () const
- void setStrokeColor (const Color &color)

  stroke color
- const Color & **getStrokeColor** () const
- void setStrokeStyle (const StrokeStyle &style)

  stroke style
- const StrokeStyle & **getStrokeStyleConst** () const
- const StrokeStyle & **getStrokeStyle** () const
- StrokeStyle & **getStrokeStyle** ()
- void setGradientStyle (const GradientStyle &style)

  gradient style
- const GradientStyle & **getGradientStyleConst** () const
- const GradientStyle & **getGradientStyle** () const
- GradientStyle & **getGradientStyle** ()
- bool createSVG (const char *src, float32_t scale=1.0f)

  create shape from SVG path
- void clearPositions ()

  shape positions
- void **setNumPositions** (uint32_t num_positions)
- void **reservePositions** (uint32_t num_positions)
- uint32_t **getNumPositions** () const
- void **setPositions** (const Vector3f *positions, uint32_t num_positions)
- void **addPositions** (const Vector3f *positions, uint32_t num_positions)
- const Vector3f * **getPositions** () const
- Vector3f * **getPositions** ()
- void **setPosition** (uint32_t index, const Vector3f &position)
- void **setPosition** (uint32_t index, float32_t x, float32_t y, float32_t z=0.0f)
- const Vector3f & **getPosition** (uint32_t index) const

- Vector3f & **getPosition** (uint32_t index)
- uint32_t addPosition (const Vector2f &position)

  add shape position
- uint32_t **addPosition** (const Vector3f &position)
- uint32_t **addPosition** (float32_t x, float32_t y, float32_t z=0.0f)
- void setTexCoord (const Rect &texcoord)

  texture coordinates
- void **setTexCoord** (float32_t left, float32_t right, float32_t bottom, float32_t top)
- const Rect & **getTexCoord** () const

Additional Inherited Members

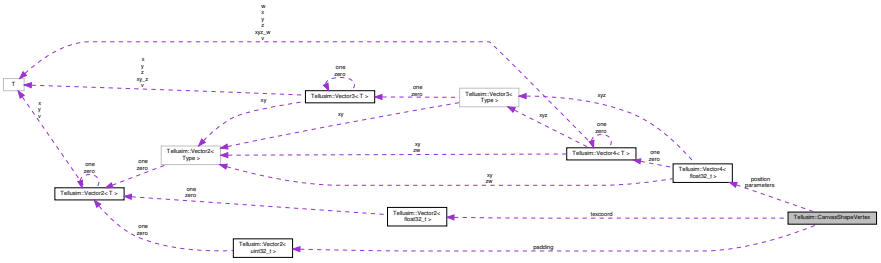
5.31.1 Detailed Description

The CanvasShape class defines a versatile shape element for a canvas that supports both cubic and quadratic curves, customizable stroke and gradient styles, and texture mapping. It enables creation of complex vector graphics by manually setting positions or importing SVG paths, with a threshold parameter controlling curve simplification. This class is ideal for rendering arbitrary outlines, icons, and path-based graphics with fine control over shape geometry and visual styling.

5.32 Tellusim::CanvasShapeVertex Struct Reference

```
#include <interface/TellusimCanvas.h>
```

Collaboration diagram for Tellusim::CanvasShapeVertex:



Public Attributes

- Vector4f **position**
- Vector4f **parameters**
- Vector2f **texcoord**
- Vector2u **padding**

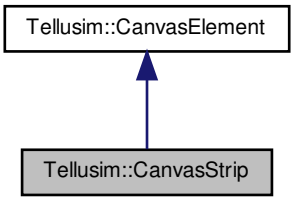
5.32.1 Detailed Description

The CanvasShapeVertex struct defines a vertex format specific to the CanvasShape class.

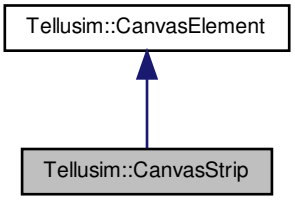
5.33 Tellusim::CanvasStrip Class Reference

`#include <interface/TellusimCanvas.h>`

Inheritance diagram for Tellusim::CanvasStrip:



Collaboration diagram for Tellusim::CanvasStrip:



Public Member Functions

- **CanvasStrip** (Canvas &canvas)
- **CanvasStrip** (Canvas &canvas, float32_t width)
- void setWidth (float32_t width)

  strip width
- float32_t **getWidth** () const
- void setOffset (float32_t offset)

  strip offset
- float32_t **getOffset** () const
- void setStrokeColor (const Color &color)

  stroke color
- const Color & **getStrokeColor** () const
- void setStrokeStyle (const StrokeStyle &style)

  stroke style
- const StrokeStyle & **getStrokeStyleConst** () const
- const StrokeStyle & **getStrokeStyle** () const

- StrokeStyle & **getStrokeStyle** ()
- void createQuadratic (const Vector2f &p0, const Vector2f &p1, const Vector2f &p2, float32_t threshold=1.0f)

  create quadratic spline
- void **createQuadratic** (const Vector3f &p0, const Vector3f &p1, const Vector3f &p2, float32_t threshold=1.0f)
- void createCubic (const Vector2f &p0, const Vector2f &p1, const Vector2f &p2, const Vector2f &p3, float32_t threshold=1.0f)

  create cubic spline
- void **createCubic** (const Vector3f &p0, const Vector3f &p1, const Vector3f &p2, const Vector3f &p3, float32_t threshold=1.0f)
- void clearPositions ()

  strip positions
- void **setNumPositions** (uint32_t num_positions)
- void **reservePositions** (uint32_t num_positions)
- uint32_t **getNumPositions** () const
- void **setPositions** (const Vector3f *positions, uint32_t num_positions)
- void **addPositions** (const Vector3f *positions, uint32_t num_positions)
- const Vector3f * **getPositions** () const
- Vector3f * **getPositions** ()
- void **setPosition** (uint32_t index, const Vector3f &position)
- void **setPosition** (uint32_t index, float32_t x, float32_t y, float32_t z=0.0f)
- const Vector3f & **getPosition** (uint32_t index) const
- Vector3f & **getPosition** (uint32_t index)
- uint32_t addPosition (const Vector2f &position)

  add strip position
- uint32_t **addPosition** (const Vector3f &position)
- uint32_t **addPosition** (float32_t x, float32_t y, float32_t z=0.0f)

Additional Inherited Members

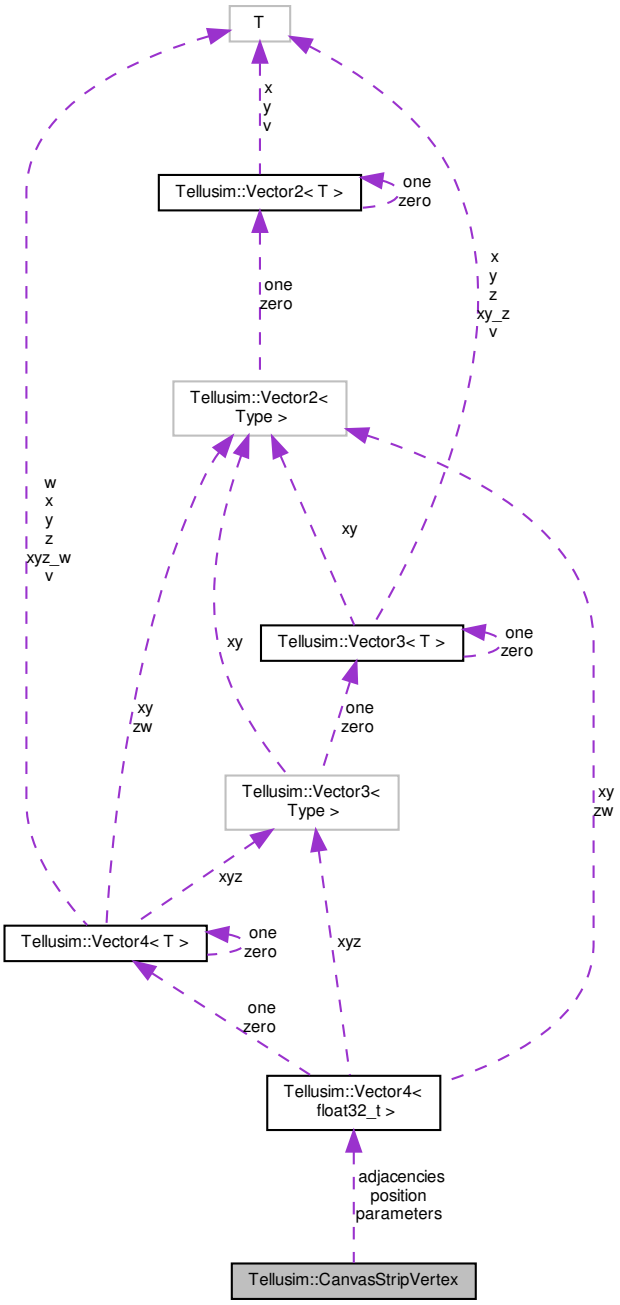
5.33.1 Detailed Description

The CanvasStrip class represents a polyline-based canvas element that renders stroked strips with variable width and offset, commonly used for drawing lines, splines, and path outlines. It supports both quadratic and cubic spline generation with adjustable approximation threshold, along with manual control over position data. Stroke color and style are fully customizable, enabling high-quality rendering of curves and outlines with precise geometry control.

5.34 Tellusim::CanvasStripVertex Struct Reference

```
#include <interface/TellusimCanvas.h>
```

Collaboration diagram for Tellusim::CanvasStripVertex:



Public Attributes

- Vector4f **position**
- Vector4f **parameters**
- Vector4f **adjacencies**

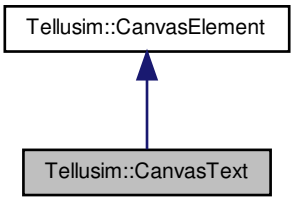
5.34.1 Detailed Description

The CanvasStripVertex struct defines a vertex format specific to the CanvasStrip class.

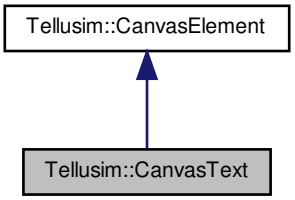
5.35 Tellusim::CanvasText Class Reference

```
#include <interface/TellusimCanvas.h>
```

Inheritance diagram for Tellusim::CanvasText:



Collaboration diagram for Tellusim::CanvasText:



Public Member Functions

- **CanvasText** (Canvas &canvas)
- **CanvasText** (Canvas &canvas, const char *text)
- **CanvasText** (Canvas &canvas, const String &text)
- void setFontName (const char *name)

  font name
- void **setFontName** (const String &name)
- String **getFontName** () const
- void setFontColor (const Color &color)

  font color

- const Color & **getFontColor** () const
- bool setFontSize (uint32_t scale)

  font style
- uint32_t **getFontSize** () const
- bool **setFontScale** (uint32_t scale)
- uint32_t **getFontScale** () const
- bool **setFontStyle** (const FontStyle &style)
- const FontStyle & **getFontStyleConst** () const
- const FontStyle & **getFontStyle** () const
- FontStyle & **getFontStyle** ()
- void setPosition (const Vector3f &position)

  font position
- void **setPosition** (float32_t x, float32_t y, float32_t z=0.0f)
- const Vector3f & **getPosition** () const
- void setText (const char *text)

  font text
- void **setText** (const String &text)
- String **getText** () const
- void clearBatches ()

  font batches
- void **setBatches** (const Array< FontBatch > &batches)
- void **setBatches** (const FontBatch *batches, uint32_t num_batches)

Additional Inherited Members

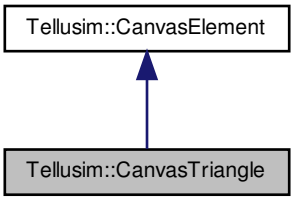
5.35.1 Detailed Description

The CanvasText class is a specialized CanvasElement used to render text within a Canvas, supporting configurable properties such as font name, size, scale, style, color, and position. It enables precise control over text layout and appearance, allowing dynamic text updates and styling through the associated FontStyle. Text content can be set directly as a string, and rendering can be optimized using font batches for efficient processing. This class is ideal for displaying scalable, styled, and positioned text in a 2D or 3D canvas context.

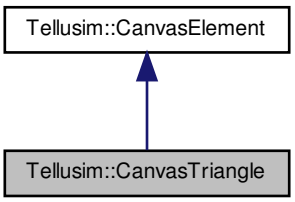
5.36 Tellusim::CanvasTriangle Class Reference

```
#include <interface/TellusimCanvas.h>
```

Inheritance diagram for Tellusim::CanvasTriangle:



Collaboration diagram for Tellusim::CanvasTriangle:



Public Member Functions

- **CanvasTriangle** (Canvas &canvas)
- **CanvasTriangle** (Canvas &canvas, float32_t radius)
- void setRadius (float32_t radius)

  triangle radius
- float32_t **getRadius** () const
- void setStrokeColor (const Color &color)

  stroke color
- const Color & **getStrokeColor** () const
- void setStrokeStyle (const StrokeStyle &style)

  stroke style
- const StrokeStyle & **getStrokeStyleConst** () const
- const StrokeStyle & **getStrokeStyle** () const
- StrokeStyle & **getStrokeStyle** ()
- void setGradientStyle (const GradientStyle &style)

  gradient style
- const GradientStyle & **getGradientStyleConst** () const
- const GradientStyle & **getGradientStyle** () const
- GradientStyle & **getGradientStyle** ()
- void setPosition0 (const Vector3f &position)

  triangle positions
- void **setPosition1** (const Vector3f &position)
- void **setPosition2** (const Vector3f &position)
- void **setPosition0** (float32_t x, float32_t y, float32_t z=0.0f)
- void **setPosition1** (float32_t x, float32_t y, float32_t z=0.0f)
- void **setPosition2** (float32_t x, float32_t y, float32_t z=0.0f)
- void **setPosition** (const Vector3f &position_0, const Vector3f &position_1, const Vector3f &position_2)
- const Vector3f & **getPosition0** () const
- const Vector3f & **getPosition1** () const
- const Vector3f & **getPosition2** () const

Additional Inherited Members

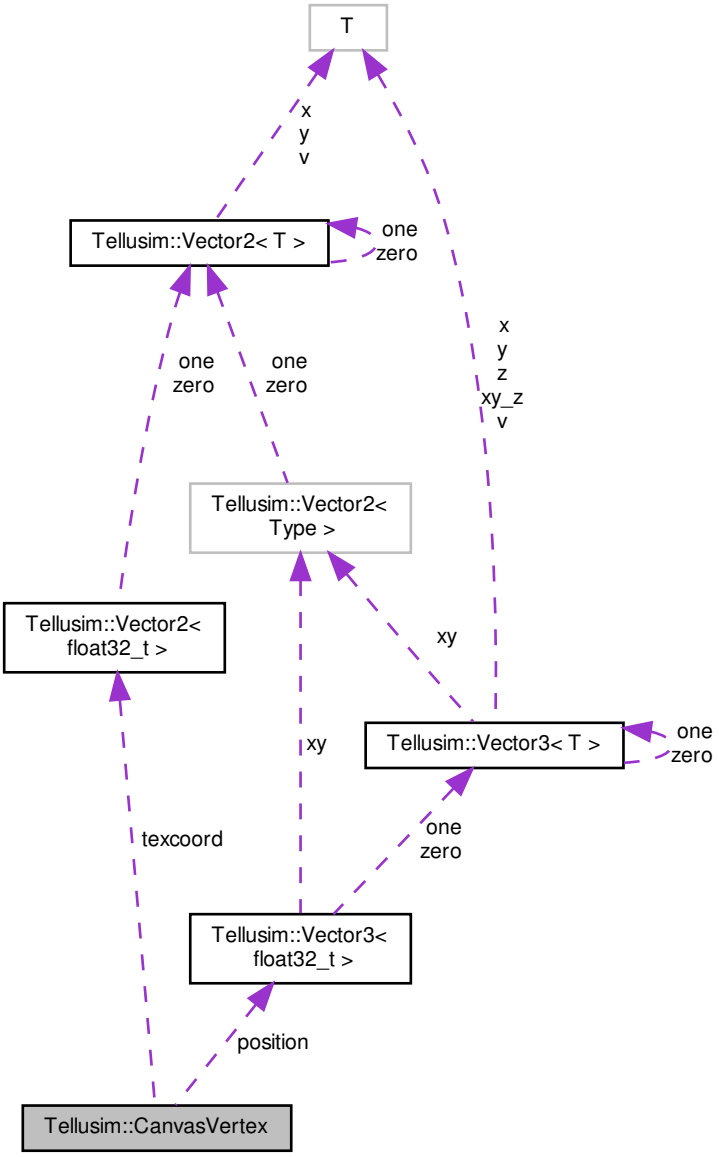
5.36.1 Detailed Description

The CanvasTriangle class represents a triangular CanvasElement that can be positioned and rendered with customizable visual attributes, including stroke color, stroke style, and gradient style. It allows setting each of the three vertex positions individually or all at once, and supports specifying a corner radius to control curvature at the triangle vertices. This class enables the rendering of filled or outlined triangles with advanced styling, useful for decorative elements, directional indicators, or UI accents within a canvas.CanvasTriangle class

5.37 Tellusim::CanvasVertex Struct Reference

```
#include <interface/TellusimCanvas.h>
```

Collaboration diagram for Tellusim::CanvasVertex:



Public Attributes

- Vector3f **position**
- Vector2f **texcoord**
- uint32_t **color**

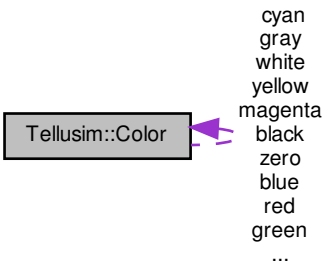
5.37.1 Detailed Description

The CanvasVertex struct defines a generic vertex format used by the Canvas and Font classes.

5.38 Tellusim::Color Struct Reference

`#include <math/TellusimColor.h>`

Collaboration diagram for Tellusim::Color:



Public Types

- enum { **Size** = 4 }

Public Member Functions

- **Color** (const char *src)
- **Color** (uint32_t color)
- **Color** (float32_t c)
- **Color** (float32_t l, float32_t a)
- **Color** (const Color &c, float32_t a)
- **Color** (const float32_t *c, uint32_t size=Size)
- **Color** (uint32_t r, uint32_t g, uint32_t b, uint32_t a=255)
- **Color** (float32_t r, float32_t g, float32_t b, float32_t a=1.0f)
- void set (const Color &c, float32_t A)

  update color data
- void **set** (float32_t R, float32_t G, float32_t B, float32_t A)
- void **set** (const float32_t *1 c, uint32_t size=Size)
- void **get** (float32_t *1 c, uint32_t size=Size) const
- bool isValid () const

  check color
- **operator bool** () const
- bool isBlack () const

  color parameters
- bool **isWhite** () const
- bool **isTransparent** () const
- bool **isOpaque** () const
- Color & operator*= (float32_t l)

  color operators
- Color & **operator/=** (float32_t l)

- Color & **operator+=** (float32_t l)
- Color & **operator-=** (float32_t l)
- Color & **operator*=** (const Color &c)
- Color & **operator/=** (const Color &c)
- Color & **operator+=** (const Color &c)
- Color & **operator-=** (const Color &c)
- void gammaToLinear (float32_t *1 v) const

  gamma format
- void **linearToGamma** (float32_t *1 v) const
- Color **gammaToLinear** () const
- Color **linearToGamma** () const
- void sRGBtoLinear (float32_t *1 v) const

  sRGB format
- void **linearToSRGB** (float32_t *1 v) const
- Color **sRGBtoLinear** () const
- Color **linearToSRGB** () const
- void setRGBAu8 (uint32_t R, uint32_t G, uint32_t B, uint32_t A)

  RGBAu8 color components.
- uint8_t **getRu8** () const
- uint8_t **getGu8** () const
- uint8_t **getBu8** () const
- uint8_t **getAu8** () const
- void setRGBAu8 (uint32_t color)

  RGBAu8 color format.
- void **setBGRAu8** (uint32_t color)
- void **setABGRu8** (uint32_t color)
- uint32_t **getRGBAu8** () const
- uint32_t **getBGRAu8** () const
- uint32_t **getABGRu8** () const
- bool setHTML (const char *src)

  HTML color format.
- bool setHSV (float32_t h, float32_t s, float32_t v)

  HSV color format.
- bool **getHSV** (float32_t &h, float32_t &s, float32_t &v) const
- void setTemperature (float32_t t)

  temperature in K
- const float32_t & operator[ ] (uint32_t index) const

  Color data.
- float32_t & **operator[ ]** (uint32_t index)

Static Public Member Functions

- static Color **html** (const char *src)
- static Color **hsv** (float32_t h, float32_t s, float32_t v)
- static Color **temperature** (float32_t t)

Public Attributes

- union {
  struct {
  float32_t **r**
  float32_t **g**
  float32_t **b**
  float32_t **a**
  }
  float32_t **c** [Size]
  };

Static Public Attributes

- static const Color zero
  default colors
- static const Color **black**
- static const Color **white**
- static const Color **gray**
- static const Color red
  rgb colors
- static const Color **yellow**
- static const Color **green**
- static const Color **cyan**
- static const Color **blue**
- static const Color **magenta**

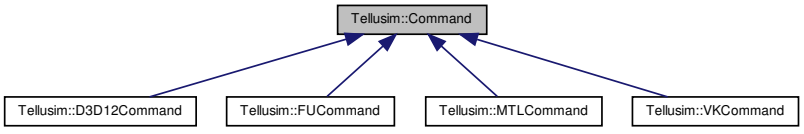
5.38.1 Detailed Description

The Color struct represents a four-component color with red, green, blue, and alpha channels, offering a flexible and efficient interface for defining, manipulating, and converting colors across different formats. It supports various constructors, including grayscale, HTML strings, RGBAu8 integers, HSV values, and temperature-based inputs. The struct provides utilities for setting and retrieving color data, checking color properties, and performing color arithmetic operations. It includes functions to convert between gamma and linear color spaces, as well as sRGB and linear formats, and supports multiple packed RGBA formats.

5.39 Tellusim::Command Class Reference

`#include <platform/TellusimCommand.h>`

Inheritance diagram for Tellusim::Command:



Classes

- struct DrawArraysIndirect
  draw arrays indirect parameters
- struct DrawElementsIndirect
  draw elements indirect parameters
- struct DrawMeshIndirect
  draw mesh indirect parameters

Public Member Functions

- Platform getPlatform () const
  command platform
- const char * **getPlatformName** () const
- uint32_t getIndex () const
  command device index
- void setPipeline (Pipeline &pipeline)
  set pipeline
- Pipeline **getPipeline** () const
- void setViewport (uint32_t index, const Viewport &viewport)
  set viewport parameters
- void **setViewports** (const Viewport *viewports, uint32_t num_viewports)
- void setScissor (uint32_t index, const Scissor &scissor)
  set scissor parameters
- void **setScissors** (const Scissor *scissors, uint32_t num_scissors)
- void setSampler (uint32_t index, Sampler &sampler)
  set samplers
- void **setSamplers** (uint32_t index, const Array< Sampler > &samplers)
- void **setSamplers** (uint32_t index, const InitializerList< Sampler > &samplers)
- void setTexture (uint32_t index, Texture &texture)
  set textures
- void **setTexture** (uint32_t index, Texture &texture, const Slice &slice)
- void **setTextures** (uint32_t index, const Array< Texture > &textures)
- void **setTextures** (uint32_t index, const InitializerList< Texture > &textures)
- void setSurfaceTexture (uint32_t index, Texture &texture)
  set surfaces
- void **setSurfaceTexture** (uint32_t index, Texture &texture, const Slice &slice, Format format=Format↩ Unknown)
- void **setSurfaceTextures** (uint32_t index, const Array< Texture > &textures)
- void **setSurfaceTextures** (uint32_t index, const InitializerList< Texture > &textures)
- void * getUniformData (uint32_t index, size_t size)
  set uniforms
- void **setUniformData** (uint32_t index, const void *src, size_t size)
- void **setUniformBuffer** (uint32_t index, Buffer &buffer, size_t offset=0, size_t size=0)
- void **setUniformOffset** (uint32_t index, size_t offset, bool relative=false)
- void **setUniformBuffers** (uint32_t index, const Array< Buffer > &buffers)
- void **setUniformBuffers** (uint32_t index, const Array< Buffer > &buffers, const Array< size_t > &offsets)
- void **setUniformBuffers** (uint32_t index, const InitializerList< Buffer > &buffers)
- void **setUniformBuffers** (uint32_t index, const InitializerList< Buffer > &buffers, const InitializerList< size↩ _t > &offsets)
- template<class Type >
  void **setUniform** (uint32_t index, const Array< Type > &data)

- template<class Type >
  void **setUniform** (uint32_t index, const Type &data)
- void * getStorageData (uint32_t index, size_t size)

  set storages
- void **setStorageData** (uint32_t index, const void *src, size_t size)
- void **setStorageBuffer** (uint32_t index, Buffer &buffer, size_t offset=0, size_t size=0)
- void **setStorageOffset** (uint32_t index, size_t offset, bool relative=false)
- void **setStorageBuffers** (uint32_t index, const Array< Buffer > &buffers)
- void **setStorageBuffers** (uint32_t index, const Array< Buffer > &buffers, const Array< size_t > &offsets)
- void **setStorageBuffers** (uint32_t index, const InitializerList< Buffer > &buffers)
- void **setStorageBuffers** (uint32_t index, const InitializerList< Buffer > &buffers, const InitializerList< size_t > &offsets)
- template<class Type >
  void **setStorage** (uint32_t index, const Array< Type > &data)
- template<class Type >
  void **setStorage** (uint32_t index, const Type &data)
- void setTracing (uint32_t index, Tracing &tracing)

  set tracings
- void **setTracings** (uint32_t index, const Array< Tracing > &tracings)
- void **setTracings** (uint32_t index, const InitializerList< Tracing > &tracings)
- void setTexelBuffer (uint32_t index, Buffer &buffer)

  set texel buffers
- void **setTexelBuffers** (uint32_t index, const Array< Buffer > &buffers)
- void **setTexelBuffers** (uint32_t index, const InitializerList< Buffer > &buffers)
- void setTextureTable (uint32_t index, TextureTable &table)

  set texture tables
- void **setTextureTables** (uint32_t index, const Array< TextureTable > &tables)
- void **setTextureTables** (uint32_t index, const InitializerList< TextureTable > &tables)
- void setStorageTable (uint32_t index, BufferTable &table)

  set storage tables
- void **setStorageTables** (uint32_t index, const Array< BufferTable > &tables)
- void **setStorageTables** (uint32_t index, const InitializerList< BufferTable > &tables)
- void * getVertexData (uint32_t index, size_t size)

  set vertices
- void **setVertexData** (uint32_t index, const void *src, size_t size)
- void **setVertexBuffer** (uint32_t index, Buffer &buffer, size_t offset=0)
- void **setVertexOffset** (uint32_t index, size_t offset, bool relative=false)
- void **setVertexBuffers** (uint32_t index, const Array< Buffer > &buffers)
- void **setVertexBuffers** (uint32_t index, const Array< Buffer > &buffers, const Array< size_t > &offsets)
- void **setVertexBuffers** (uint32_t index, const InitializerList< Buffer > &buffers)
- void **setVertexBuffers** (uint32_t index, const InitializerList< Buffer > &buffers, const InitializerList< size_t > &offsets)
- template<class Type >
  void **setVertices** (uint32_t index, const Array< Type > &vertices)
- template<class Type , size_t Size>
  void **setVertices** (uint32_t index, const Type(&vertices)[Size])
- template<class Type >
  void **setVertices** (uint32_t index, const InitializerList< Type > &vertices)
- void * getIndexData (Format format, size_t size)

  set indices
- void **setIndexData** (Format format, const void *src, size_t size)
- void **setIndexBuffer** (Format format, Buffer &buffer, size_t offset=0)
- void **setIndexOffset** (size_t offset, bool relative=false)

- template<class Type >
  void **setIndices** (Format format, const Array< Type > &indices)
- template<class Type , size_t Size>
  void **setIndices** (Format format, const Type(&indices)[Size])
- void **setIndices** (const InitializerList< uint16_t > &indices)
- void * getIndirectData (size_t size)

  set indirect
- void **setIndirectData** (const void *src, size_t size)
- void **setIndirectBuffer** (Buffer &buffer, size_t offset=0)
- void **setIndirectOffset** (size_t offset, bool relative=false)
- template<class Type >
  void **setIndirect** (const Type &data)
- void setBlendColor (const Color &color)

  blending parameters
- void **setBlendColor** (float32_t r, float32_t g, float32_t b, float32_t a)
- void setStencilRef (uint32_t ref)

  stencil parameters
- void drawArrays (uint32_t num_vertices, uint32_t base_vertex=0)

  draw arrays
- void **drawArraysInstanced** (uint32_t num_vertices, uint32_t base_vertex, uint32_t num_instances, uint32_t base_instance=0)
- void **drawArraysIndirect** (uint32_t num_draws, size_t stride=sizeof(DrawArraysIndirect))
- void **drawArraysIndirect** (Buffer &buffer, size_t offset, uint32_t num_draws, size_t stride=sizeof(DrawArraysIndirect))
- void drawElements (uint32_t num_indices, uint32_t base_index=0, int32_t base_vertex=0)

  draw elements
- void **drawElementsInstanced** (uint32_t num_indices, uint32_t base_index, uint32_t num_instances)
- void **drawElementsInstanced** (uint32_t num_indices, uint32_t base_index, int32_t base_vertex, uint32_t num_instances, uint32_t base_instance=0)
- void **drawElementsIndirect** (uint32_t num_draws, size_t stride=sizeof(DrawElementsIndirect))
- void **drawElementsIndirect** (Buffer &buffer, size_t offset, uint32_t num_draws, size_t stride=sizeof(DrawElementsIndirect))
- void drawMesh (uint32_t width, uint32_t height=1, uint32_t depth=1)

  draw mesh
- void **drawMeshIndirect** (uint32_t num_draws, size_t stride=sizeof(DrawMeshIndirect))
- void **drawMeshIndirect** (Buffer &buffer, size_t offset, uint32_t num_draws, size_t stride=sizeof(DrawMeshIndirect))
- void beginConditional (Buffer &buffer, size_t offset)

  begin/end conditional
- void **endConditional** ()
- bool beginQuery (Query &query)

  begin/end query
- void **endQuery** (Query &query)

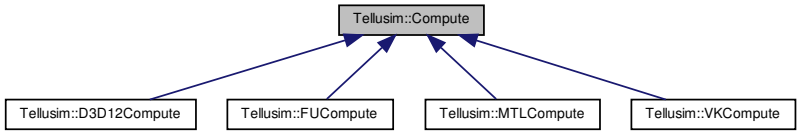
5.39.1 Detailed Description

The Command class is responsible for resource binding and the execution of rendering commands. It facilitates the setup of various graphics resources, such as pipelines, viewports, samplers, textures, buffers, and uniforms, all required for rendering operations. The class supports managing multiple render targets, handling various draw commands, and controlling query and conditional execution. Additionally, it provides capabilities for indirect drawing, resource barriers, and blending/stencil state configurations, ensuring efficient and flexible rendering command execution across different platforms.

5.40 Tellusim::Compute Class Reference

`#include <platform/TellusimCompute.h>`

Inheritance diagram for Tellusim::Compute:



Classes

- struct DispatchIndirect
  dispatch indirect parameters

Public Member Functions

- Platform getPlatform () const
  compute platform
- const char * **getPlatformName** () const
- uint32_t getIndex () const
  compute device index
- void setKernel (Kernel &kernel)
  set kernel
- Kernel **getKernel** () const
- void setTraversal (Traversal &traversal)
  set traversal
- Traversal **getTraversal** () const
- void setSampler (uint32_t index, Sampler &sampler)
  set samplers
- void **setSamplers** (uint32_t index, const Array< Sampler > &samplers)
- void **setSamplers** (uint32_t index, const InitializerList< Sampler > &samplers)
- void setTexture (uint32_t index, Texture &texture)
  set textures
- void **setTexture** (uint32_t index, Texture &texture, const Slice &slice)
- void **setTextures** (uint32_t index, const Array< Texture > &textures)
- void **setTextures** (uint32_t index, const InitializerList< Texture > &textures)
- void setSurfaceTexture (uint32_t index, Texture &texture)
  set surfaces
- void **setSurfaceTexture** (uint32_t index, Texture &texture, const Slice &slice, Format format=Format↩Unknown)
- void **setSurfaceTextures** (uint32_t index, const Array< Texture > &textures)
- void **setSurfaceTextures** (uint32_t index, const InitializerList< Texture > &textures)
- void * getUniformData (uint32_t index, size_t size)
  set uniforms
- void **setUniformData** (uint32_t index, const void *src, size_t size)

- void **setUniformBuffer** (uint32_t index, Buffer &buffer, size_t offset=0, size_t size=0)
- void **setUniformOffset** (uint32_t index, size_t offset, bool relative=false)
- void **setUniformBuffers** (uint32_t index, const Array< Buffer > &buffers)
- void **setUniformBuffers** (uint32_t index, const Array< Buffer > &buffers, const Array< size_t > &offsets)
- void **setUniformBuffers** (uint32_t index, const InitializerList< Buffer > &buffers)
- void **setUniformBuffers** (uint32_t index, const InitializerList< Buffer > &buffers, const InitializerList< size_t > &offsets)
- template<class Type >
  void **setUniform** (uint32_t index, const Array< Type > &data)
- template<class Type >
  void **setUniform** (uint32_t index, const Type &data)
- void * getStorageData (uint32_t index, size_t size)

  set storages
- void **setStorageData** (uint32_t index, const void *src, size_t size)
- void **setStorageBuffer** (uint32_t index, Buffer &buffer, size_t offset=0, size_t size=0)
- void **setStorageOffset** (uint32_t index, size_t offset, bool relative=false)
- void **setStorageBuffers** (uint32_t index, const InitializerList< Buffer > &buffers)
- void **setStorageBuffers** (uint32_t index, const InitializerList< Buffer > &buffers, const InitializerList< size_t > &offsets)
- void **setStorageBuffers** (uint32_t index, const Array< Buffer > &buffers)
- void **setStorageBuffers** (uint32_t index, const Array< Buffer > &buffers, const Array< size_t > &offsets)
- template<class Type >
  void **setStorage** (uint32_t index, const Array< Type > &data)
- template<class Type >
  void **setStorage** (uint32_t index, const Type &data)
- void setTracing (uint32_t index, Tracing &tracing)

  set tracings
- void **setTracings** (uint32_t index, const Array< Tracing > &tracings)
- void **setTracings** (uint32_t index, const InitializerList< Tracing > &tracings)
- void setTexelBuffer (uint32_t index, Buffer &buffer)

  set texel buffers
- void **setTexelBuffers** (uint32_t index, const Array< Buffer > &buffers)
- void **setTexelBuffers** (uint32_t index, const InitializerList< Buffer > &buffers)
- void setTextureTable (uint32_t index, TextureTable &table)

  set texture tables
- void **setTextureTables** (uint32_t index, const Array< TextureTable > &tables)
- void **setTextureTables** (uint32_t index, const InitializerList< TextureTable > &tables)
- void setStorageTable (uint32_t index, BufferTable &table)

  set storage tables
- void **setStorageTables** (uint32_t index, const Array< BufferTable > &tables)
- void **setStorageTables** (uint32_t index, const InitializerList< BufferTable > &tables)
- void * getIndirectData (size_t size)

  set indirect
- void **setIndirectData** (const void *src, size_t size)
- void **setIndirectBuffer** (Buffer &buffer, size_t offset=0)
- void **setIndirectOffset** (size_t offset, bool relative=false)
- template<class Type >
  void **setIndirect** (const Type &data)
- void dispatch (uint32_t width, uint32_t height=1, uint32_t depth=1)

  dispatch kernel
- void **dispatch** (const Texture &texture)
- void **dispatch** (const Size &size)
- void dispatchIndirect ()

  dispatch kernel indirect

- bool setBuffer (Buffer &buffer, size_t offset, const void *src, size_t size)

  set buffer data
- bool **setBuffer** (Buffer &buffer, const void *src, size_t size)
- bool **setBuffer** (Buffer &buffer, const void *src)
- bool copyBuffer (Buffer &buffer, size_t dest_offset, Buffer &src, size_t src_offset, size_t size)

  copy buffer data
- bool **copyBuffer** (Buffer &buffer, size_t dest_offset, Buffer &src, size_t size)
- bool **copyBuffer** (Buffer &buffer, Buffer &src, size_t size)
- bool **copyBuffer** (Buffer &buffer, Buffer &src)
- bool clearBuffer (Buffer &buffer, Format format, size_t offset, const void *src, size_t size)

  clear buffer data
- bool **clearBuffer** (Buffer &buffer, Format format, const void *src, size_t size)
- bool **clearBuffer** (Buffer &buffer, Format format, const void *src)
- bool **clearBuffer** (Buffer &buffer)
- bool setTexture (Texture &texture, const Origin &dest_origin, const Slice &dest_slice, const Image &image, const Slice &src_slice)

  set texture data
- bool **setTexture** (Texture &texture, const Origin &dest_origin, const Image &image)
- bool **setTexture** (Texture &texture, const Slice &dest_slice, const Image &image)
- bool **setTexture** (Texture &texture, const Image &image)
- bool copyTexture (Texture &texture, const Origin &dest_origin, const Slice &dest_slice, Texture &src, const Region &src_region, const Slice &src_slice)

  copy texture data
- bool **copyTexture** (Texture &texture, const Origin &dest_origin, Texture &src, const Region &src_region)
- bool **copyTexture** (Texture &texture, const Slice &dest_slice, Texture &src, const Slice &src_slice)
- bool **copyTexture** (Texture &texture, Texture &src)
- bool clearTexture (Texture &texture, const Region &region, const Slice &slice, const void *src)

  clear texture data
- bool **clearTexture** (Texture &texture, const Region &region, const void *src)
- bool **clearTexture** (Texture &texture, const Slice &slice, const void *src)
- bool **clearTexture** (Texture &texture, const void *src)
- void barrier (Texture &texture)

  resource barriers
- void **barrier** (Buffer &buffer)
- void **barrier** (const Array< Texture > &textures)
- void **barrier** (const Array< Buffer > &buffers)
- void **barrier** (const InitializerList< Texture > &textures)
- void **barrier** (const InitializerList< Buffer > &buffers)
- void beginConditional (Buffer &buffer, size_t offset)

  begin/end conditional
- void **endConditional** ()
- bool beginQuery (Query &query)

  begin/end query
- void **endQuery** (Query &query)

5.40.1 Detailed Description

The Compute class is responsible for resource binding and the execution of compute kernels. It allows setting up and binding various resources such as kernels, samplers, textures, buffers, and other data structures required for compute operations. The class manages resource bindings, executes compute tasks on the target platform, and supports dispatching kernels with different configurations. It also facilitates handling of indirect execution, resource barriers, conditional execution, and query management, ensuring efficient execution of compute workloads across various platforms.

5.41 Tellusim::RadixMap< Key, Type, Size >::ConstIterator Class Reference

Constant iterator.

`#include <core/TellusimRadix.h>`

Public Member Functions

- **ConstIterator** (const Iterator &it)
- void **clear** ()
- ConstIterator & **operator=** (const Iterator &it)
- ConstIterator & **operator=** (const ConstIterator &it)
- **operator bool** () const
- bool **operator==** (const ConstIterator &it) const
- bool **operator!=** (const ConstIterator &it) const
- ConstIterator & **operator++** ()
- ConstIterator & **operator--** ()
- ConstIterator **operator++** (int32_t)
- ConstIterator **operator--** (int32_t)
- ConstIterator **next** ()
- ConstIterator **prev** ()
- const Type & **operator*** () const
- const Type * **operator->** () const
- const Type & **get** () const

Friends

- class **RadixMap**

5.41.1 Detailed Description

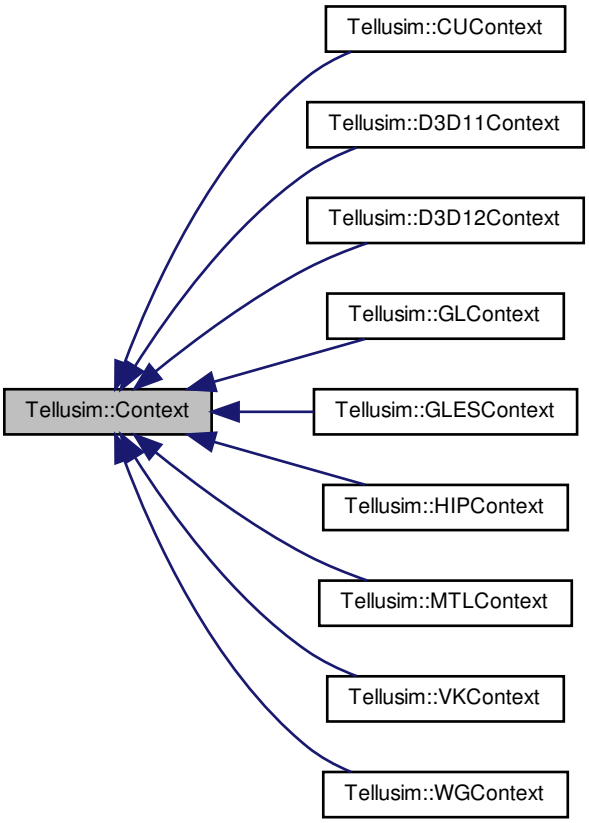
template<class Key, class Type, uint32_t Size = 32>
class Tellusim::RadixMap< Key, Type, Size >::ConstIterator

Constant iterator.

5.42 Tellusim::Context Class Reference

`#include <platform/TellusimContext.h>`

Inheritance diagram for Tellusim::Context:



Public Member Functions

- Context ()

  context constructor
- **Context** (Platform platform, uint32_t index=Maxu32)
- Platform getPlatform () const

  context platform
- const char * **getPlatformName** () const
- uint32_t getIndex () const

  context device index
- bool isCreated () const

  check context
- bool create ()

  create context
- bool flush ()

  flush context
- bool **finish** ()

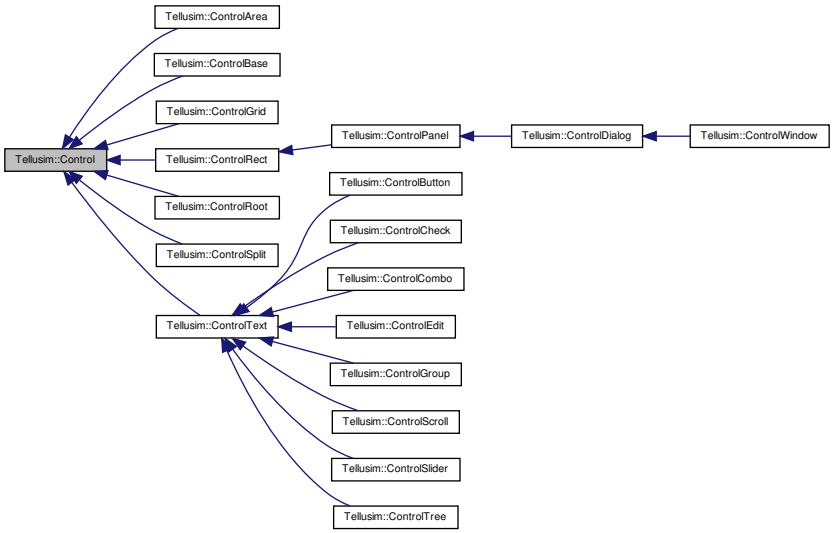
5.42.1 Detailed Description

The Context class initializes the target platform API in headless mode. It provides a simple way to create compute or off-screen rendering tasks. It can be constructed with a specific platform and an optional device index. For visualization, the separate Window class should be used.

5.43 Tellusim::Control Class Reference

`#include <interface/TellusimControls.h>`

Inheritance diagram for Tellusim::Control:



Public Types

- enum State {
  StateUnknown = 0,
  StateNormal,
  StateFocused,
  StatePressed,
  StateDisabled,
  NumStates }

  Control states.
- enum Mesh {
  MeshCheck = 0,
  MeshButton,
  MeshSliderLine,
  MeshHScrollLine,
  MeshVScrollLine,
  MeshSliderHandle,
  MeshHScrollHandle,
  MeshVScrollHandle,
  MeshBackground,
  MeshSelection,
  MeshBorder,
  MeshFrame,
  NumMeshes }

Control meshes.

- enum Align {
  AlignNone = 0,
  AlignLeft = (1 << 0),
  AlignRight = (1 << 1),
  AlignBottom = (1 << 2),
  AlignTop = (1 << 3),
  AlignCenterX = (1 << 4),
  AlignCenterY = (1 << 5),
  AlignExpandX = (1 << 6),
  AlignExpandY = (1 << 7),
  AlignOverlap = (1 << 8),
  AlignSpacer = (1 << 9),
  AlignAspect = (1 << 10),
  AlignLeftBottom = (AlignLeft | AlignBottom),
  AlignLeftTop = (AlignLeft | AlignTop),
  AlignRightBottom = (AlignRight | AlignBottom),
  AlignRightTop = (AlignRight | AlignTop),
  AlignCenter = (AlignCenterX | AlignCenterY),
  AlignExpand = (AlignExpandX | AlignExpandY),
  NumAligns = 11 }

  Control alignments.

- enum Button {
  ButtonNone = 0,
  ButtonLeft = (1 << 0),
  ButtonLeft2 = (1 << 1),
  ButtonRight = (1 << 2),
  ButtonRight2 = (1 << 3),
  ButtonMiddle = (1 << 4),
  ButtonMiddle2 = (1 << 5),
  NumButtons = 6 }

  Control buttons.

- enum Axis {
  AxisUnknown = 0,
  AxisX,
  AxisY,
  AxisZ,
  AxisW,
  NumAxes }

  Control axes.

- enum Key {
  KeyNone = 128,
  KeyTab,
  KeyBackspace,
  KeyDelete,
  KeyInsert,
  KeyReturn,
  KeyPrior,
  KeyNext,
  KeyEnd,
  KeyHome,
  KeyUp,
  KeyDown,
  KeyLeft,
  KeyRight,
  KeyShift,
  KeyCtrl,

KeyAlt,
KeyCmd,
NumKeys,
KeyOption = KeyCtrl }

Control keys.

Public Member Functions

- **Control** (Control *parent)
- **Control** (Control *parent, float32_t width, float32_t height=0.0f)
- Type getType () const

  control type
- const char * **getTypeName** () const
- bool **isUnknown** () const
- bool **isRoot** () const
- bool **isText** () const
- bool **isRect** () const
- bool **isGrid** () const
- bool **isGroup** () const
- bool **isPanel** () const
- bool **isDialog** () const
- bool **isWindow** () const
- bool **isCheck** () const
- bool **isCombo** () const
- bool **isButton** () const
- bool **isSlider** () const
- bool **isScroll** () const
- bool **isSplit** () const
- bool **isArea** () const
- bool **isTree** () const
- bool **isEdit** () const
- void setAlign (Align align)

  control alignment
- Align **getAlign** () const
- bool **hasAlign** (Align align) const
- bool **hasAligns** (Align aligns) const
- bool **isSpacer** () const
- void setCreated (bool created)

  control created
- bool **isCreated** () const
- void setEnabled (bool enabled)

  control enabled flag
- bool **isEnabled** () const
- bool **wasEnabled** () const
- bool **wasUpdated** () const
- void setDisabled (bool disabled)

  control disabled flag
- bool **isDisabled** () const
- Canvas getCanvas () const

  control canvas
- const ControlRoot getRoot () const

  control root
- ControlRoot **getRoot** ()

- const ControlPanel getPanel () const

  control panel
- ControlPanel **getPanel** ()
- uint32_t setParent (Control &parent)

  control parent
- const Control **getParent** () const
- Control **getParent** ()
- bool **isParentEnabled** () const
- bool **isParentDisabled** () const
- uint32_t addChild (Control &child)

  control children
- Control **setChild** (uint32_t index, Control &child)
- bool **raiseChild** (Control &child)
- bool **lowerChild** (Control &child)
- bool **removeChild** (Control &child)
- void **releaseChildren** ()
- uint32_t **findChild** (const Control &child) const
- bool **isChild** (const Control &child, bool hierarchy=false) const
- uint32_t **getNumChildren** () const
- const Array< Control > **getChildren** () const
- Array< Control > **getChildren** ()
- const Control **getChild** (uint32_t index) const
- Control **getChild** (uint32_t index)
- void setSize (const Vector2f &size)

  control size
- void **setSize** (float32_t width, float32_t height)
- const Vector2f & **getSize** () const
- float32_t **getWidth** () const
- float32_t **getHeight** () const
- void setPosition (const Vector3f &position)

  control position
- void **setPosition** (float32_t x, float32_t y, float32_t z=0.0f)
- const Vector3f & **getPosition** () const
- float32_t **getPositionX** () const
- float32_t **getPositionY** () const
- void setOffset (const Vector3f &offset)

  control offset
- void **setOffset** (float32_t x, float32_t y, float32_t z=0.0f)
- const Vector3f & **getOffset** () const
- float32_t **getOffsetX** () const
- float32_t **getOffsetY** () const
- void setMargin (float32_t value)

  control margin
- void **setMargin** (float32_t horizontal, float32_t vertical)
- void **setMargin** (float32_t left, float32_t right, float32_t bottom, float32_t top)
- void **setMargin** (const Rect &margin)
- const Rect & **getMargin** () const
- const Rect & getRect () const

  control rectangle
- State getState () const

  control state

Static Public Member Functions

- static const char * **getTypeName** (Type type)

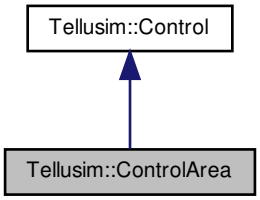
5.43.1 Detailed Description

The Control class is a base class for UI controls such as buttons, sliders, and panels, providing methods to define their type, state, and alignment. It supports flexible parent-child relationships, allowing for the addition, removal, and organization of controls in a hierarchical layout. The class also manages control size, position, margins, and offsets, with the ability to enable or disable controls based on user interaction. It handles various control states, such as focused, pressed, or disabled, and allows for querying and modifying these states. Additionally, it provides functionality for managing interaction events, such as mouse clicks and key presses, making it essential for creating dynamic and interactive UI elements.

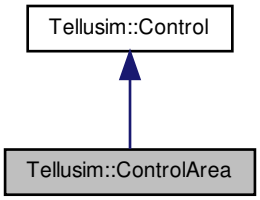
5.44 Tellusim::ControlArea Class Reference

```
#include <interface/TellusimControls.h>
```

Inheritance diagram for Tellusim::ControlArea:



Collaboration diagram for Tellusim::ControlArea:



Public Member Functions

- **ControlArea** (Control *parent)
- **ControlArea** (Control *parent, uint32_t columns)
- **ControlArea** (Control *parent, bool horizontal, bool vertical)
- **ControlArea** (Control *parent, uint32_t columns, float32_t x, float32_t y)
- void setAbsolute (bool absolute)

  absolute flag
- bool **isAbsolute** () const
- void setScalable (bool scalable)

  scalable flag
- bool **isScalable** () const
- void setScrollable (bool scrollable)

  scrollable flag
- bool **isScrollable** () const
- void setScale (float32_t scale)

  area scale
- float32_t **getScale** () const
- void setScaleRange (float32_t min, float32_t max)

  scale range
- float32_t **getMinScale** () const
- float32_t **getMaxScale** () const
- void setHorizontalStep (float64_t step)

  area step
- void **setVerticalStep** (float64_t step)
- void **setStep** (float64_t horizontal, float64_t vertical)
- float64_t **getHorizontalStep** () const
- float64_t **getVerticalStep** () const
- void setHorizontalValue (float64_t value)

  area value
- void **setVerticalValue** (float64_t value)
- void **setValue** (float64_t horizontal, float64_t vertical)
- float64_t **getHorizontalValue** () const
- float64_t **getVerticalValue** () const
- void setFrameAlign (Align align)

  frame alignment
- Align **getFrameAlign** () const
- float64_t getHorizontalFrame () const

  area frame
- float64_t **getVerticalFrame** () const
- float64_t getHorizontalRange () const

  area range
- float64_t **getVerticalRange** () const
- void setHorizontalEnabled (bool enabled, bool dynamic=false)

  horizontal scroll
- bool **isHorizontalEnabled** () const
- bool **isHorizontalDynamic** () const
- bool **isHorizontalHidden** () const
- const ControlScroll **getHorizontalScroll** () const
- ControlScroll **getHorizontalScroll** ()
- void setVerticalEnabled (bool enabled, bool dynamic=false)

  vertical scroll
- bool **isVerticalEnabled** () const

- bool **isVerticalDynamic** () const
- bool **isVerticalHidden** () const
- const ControlScroll **getVerticalScroll** () const
- ControlScroll **getVerticalScroll** ()
- bool setFontSize (uint32_t size)

  font style
- uint32_t **getFontSize** () const
- bool **setFontStyle** (const FontStyle &style)
- const FontStyle & **getFontStyleConst** () const
- const FontStyle & **getFontStyle** () const
- FontStyle & **getFontStyle** ()
- void setColumns (uint32_t columns)

  number of columns
- uint32_t **getColumns** () const
- void setSpacing (const Vector2f &spacing)

  grid spacing
- void **setSpacing** (float32_t x, float32_t y)
- const Vector2f & **getSpacing** () const
- void setColumnRatio (uint32_t index, float32_t ratio)

  grid column ratio
- float32_t **getColumnRatio** (uint32_t index) const
- const Vector2f & getControlsSize () const

  controls size
- const Vector2f & getControlsOffset () const

  controls offset
- const Rect & getViewRect () const

  view rectangle

Additional Inherited Members

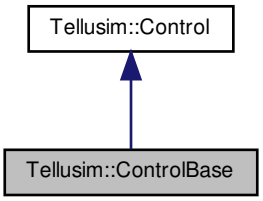
5.44.1 Detailed Description

The ControlArea class represents a flexible area control within a user interface, capable of managing multiple controls in a grid-like layout. It allows for both horizontal and vertical scrolling, configurable scaling, and offers a range of alignment, size, and step options. The class supports dynamic scrolling, with adjustable values and steps for both horizontal and vertical axes. This control is ideal for creating organized, scrollable, and scalable grid-based layouts within a user interface.

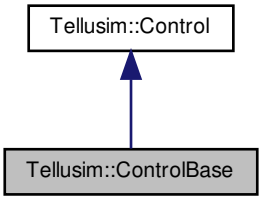
5.45 Tellusim::ControlBase Class Reference

```
#include <interface/TellusimControls.h>
```

Inheritance diagram for Tellusim::ControlBase:



Collaboration diagram for Tellusim::ControlBase:



Protected Member Functions

- **ControlBase** (Control *parent)
- virtual void clear ()

  clear control
- void create ()

  create control
- CanvasText create_text ()

  create canvas
- CanvasMesh **create_mesh** ()
- virtual bool is_batch () const

  control batch
- bool get_disabled (ControlRoot &root) const

  disabled control
- Vector3f get_position (const Rect &region, uint32_t scale) const

  control position
- virtual void update_enabled (bool enabled)

  update control
- virtual void **update_disabled** (bool disabled)
- virtual void **update_style** (const FontStyle &style)

- virtual void **update_expand** (ControlRoot &root, const Rect &region)
- virtual void **update_position** (ControlRoot &root, const Vector2f &offset)
- virtual void **update_mouse** (ControlRoot &root, Axis axis, float32_t delta)
- virtual bool **update_keyboard** (ControlRoot &root, uint32_t key, uint32_t code)
- virtual void **update_rectangle** (ControlRoot &root, int32_t &order, uint32_t scale)
- virtual bool **update** (ControlRoot &root, const Rect &region, const Rect &view, uint32_t scale)
- void update_text (ControlRoot &root, CanvasText &canvas_text, const Color &color, State state, const Vector3f &position) const

  update canvas
- void **update_mesh** (ControlRoot &root, CanvasMesh &canvas_mesh, Mesh mesh, State state, const Rect &rect, uint32_t scale, bool clear=true) const
- State set_state (ControlRoot &root, State state)

  set control state
- void set_rect (const Rect &rect)

  set control rect

Additional Inherited Members

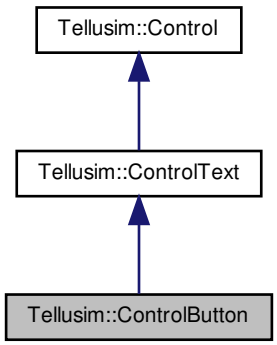
5.45.1 Detailed Description

The ControlBase class serves as the foundation for creating various user interface controls. It provides essential functionalities for control creation, position management, and updates to control states, styles, and interactions. The class allows for control initialization and clearing, creation of canvas elements, and provides functionality to manage control visibility and state changes. It includes methods for updating control properties like position, style, mouse input, keyboard input, and control expansion. It also includes utilities for managing control states and rendering through canvases, making it suitable as a base class for more specific control implementations.

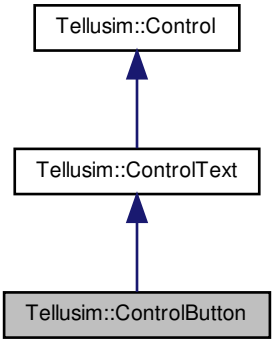
5.46 Tellusim::ControlButton Class Reference

```
#include <interface/TellusimControls.h>
```

Inheritance diagram for Tellusim::ControlButton:



Collaboration diagram for Tellusim::ControlButton:



Public Types

- using PressedCallback = Function< void(ControlButton, float32_t x, float32_t y)>
  pressed callback
- using ReleasedCallback = Function< void(ControlButton, float32_t x, float32_t y)>
  released callback
- using ClickedCallback = Function< void(ControlButton)>
  clicked callback

Public Member Functions

- **ControlButton** (Control *parent)
- **ControlButton** (Control *parent, const char *text)
- **ControlButton** (Control *parent, const String &text)
- void setBackground (bool background)
  background flag
- bool **getBackground** () const
- void setButtonMode (CanvasElement::Mode mode)
  control mode
- CanvasElement::Mode **getButtonMode** () const
- void setButtonRadius (float32_t radius)
  button radius
- float32_t **getButtonRadius** () const
- void setButtonColor (const Color &color)
  button color
- const Color & **getButtonColor** () const
- void setStrokeStyle (const StrokeStyle &style)
  stroke style
- const StrokeStyle & **getStrokeStyleConst** () const
- const StrokeStyle & **getStrokeStyle** () const
- StrokeStyle & **getStrokeStyle** ()
- void setGradientStyle (const GradientStyle &style)

gradient style

- const GradientStyle & **getGradientStyleConst** () const
- const GradientStyle & **getGradientStyle** () const
- GradientStyle & **getGradientStyle** ()
- void **setPressedCallback** (const PressedCallback &func)
- PressedCallback **getPressedCallback** () const
- bool **isPressed** ()
- void **setReleasedCallback** (const ReleasedCallback &func)
- ReleasedCallback **getReleasedCallback** () const
- bool **isReleased** ()
- void **setClickedCallback** (const ClickedCallback &func)
- ClickedCallback **getClickedCallback** () const
- bool **isClicked** ()
- CanvasRect getCanvasRect ()

canvas elements

- CanvasMesh **getCanvasMesh** ()

Additional Inherited Members

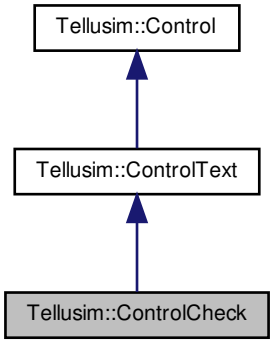
5.46.1 Detailed Description

The ControlButton class represents a button control that can be customized with various visual properties such as background, radius, color, stroke style, and gradient style. The class allows for defining callbacks for button interactions, including when the button is pressed, released, or clicked, enabling responsive behavior based on user input.

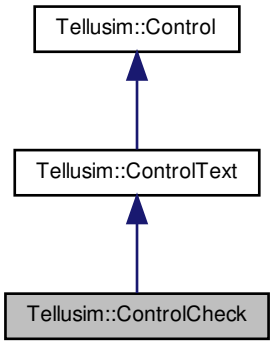
5.47 Tellusim::ControlCheck Class Reference

```
#include <interface/TellusimControls.h>
```

Inheritance diagram for Tellusim::ControlCheck:



Collaboration diagram for Tellusim::ControlCheck:



Public Types

- using ClickedCallback = Function< void(ControlCheck)>

  clicked callback

Public Member Functions

- **ControlCheck** (Control *parent)
- **ControlCheck** (Control *parent, const char *text)
- **ControlCheck** (Control *parent, const String &text)
- **ControlCheck** (Control *parent, const char *text, bool checked)
- **ControlCheck** (Control *parent, const String &text, bool checked)
- void setCheckText (const char *text)

  check text
- void **setCheckText** (const String &text)
- String **getCheckText** () const
- void setCheckColor (const Color &color)

  check color
- const Color & **getCheckColor** () const
- void setCheckedColor (const Color &color)

  checked color
- const Color & **getCheckedColor** () const
- bool switchChecked (bool callback=false)

  checked state
- void **setChecked** (bool checked, bool callback=false)
- bool **isChecked** () const
- void **setClickedCallback** (const ClickedCallback &func)
- ClickedCallback **getClickedCallback** () const
- bool **isClicked** ()
- CanvasMesh getCanvasMesh ()

  canvas elements

Additional Inherited Members

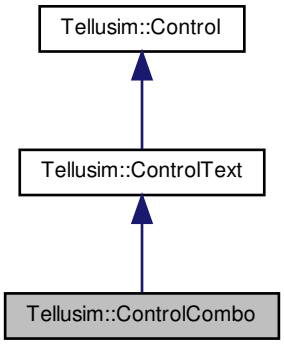
5.47.1 Detailed Description

The ControlCheck class provides a UI control for a checkable text element, typically rendered as a checkbox with an associated label. It allows customization of check and checked colors, handles checked state toggling, and supports a ClickedCallback to respond to user interaction. It inherits from ControlText, enabling full styling and text layout capabilities, and uses CanvasMesh for rendering the checkbox graphic.

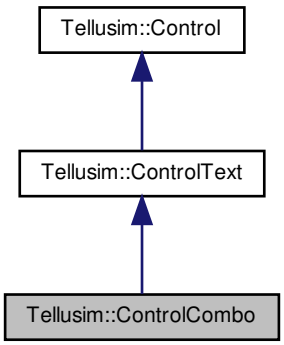
5.48 Tellusim::ControlCombo Class Reference

```
#include <interface/TellusimControls.h>
```

Inheritance diagram for Tellusim::ControlCombo:



Collaboration diagram for Tellusim::ControlCombo:



Public Types

- using ClickedCallback = Function< void(ControlCombo)>

  clicked callback
- using ChangedCallback = Function< void(ControlCombo)>

  changed callback

Public Member Functions

- **ControlCombo** (Control *parent)
- **ControlCombo** (Control *parent, const InitializerList< const char *> &items)
- **ControlCombo** (Control *parent, const InitializerList< const char *> &items, uint32_t index)
- void setTextEnabled (bool enabled)

  text enabled flag
- bool **isTextEnabled** () const
- void setMultiSelection (bool multi_selection)

  multi-selection flag
- bool **isMultiSelection** () const
- void setComboText (const char *text)

  combo text
- void **setComboText** (const String &text)
- String **getComboText** () const
- void setComboColor (const Color &color)

  combo color
- const Color & **getComboColor** () const
- void setItemsSpacing (float32_t spacing)

  items spacing
- float32_t **getItemsSpacing** () const
- void clearItems ()

  combo items
- uint32_t **addItem** (const char *text)
- uint32_t **addItem** (const String &text)
- void **addItem** (uint32_t index, const char *text)
- void **addItem** (uint32_t index, const String &text)
- void **addItems** (const InitializerList< const char *> &items)
- void **removeItem** (uint32_t index)
- uint32_t **getNumItems** () const
- bool switchItemSelected (uint32_t index)

  item selected flag
- void **setItemSelected** (uint32_t index, bool selected)
- bool **isItemSelected** (uint32_t index) const
- void setItemText (uint32_t index, const char *text)

  item text
- void **setItemText** (uint32_t index, const String &text)
- String **getItemText** (uint32_t index) const
- uint32_t **findItemText** (const char *text) const
- uint32_t **findItemText** (const String &text) const
- void setItemColor (uint32_t index, const Color &color)

  item color
- const Color & **getItemColor** (uint32_t index) const
- void setCurrentIndex (uint32_t index, bool callback=false)

  current item

- bool **setCurrentText** (const char *text, bool callback=false)
- bool **setCurrentText** (const String &text, bool callback=false)
- uint32_t **getCurrentIndex** () const
- String **getCurrentText** () const
- void **setClickedCallback** (const ClickedCallback &func)
- ClickedCallback **getClickedCallback** () const
- bool **isClicked** ()
- void **setChangedCallback** (const ChangedCallback &func)
- ChangedCallback **getChangedCallback** () const
- bool **isChanged** ()
- CanvasMesh getCanvasMesh ()

  canvas elements

Additional Inherited Members

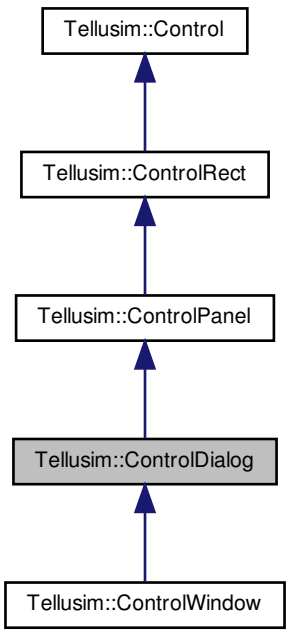
5.48.1 Detailed Description

The ControlCombo class represents a combo box control that allows users to select from a list of items. It provides functionality to enable or disable text input, as well as enable multi-selection mode. The combo box text and color can be set and retrieved, and item spacing can be customized.

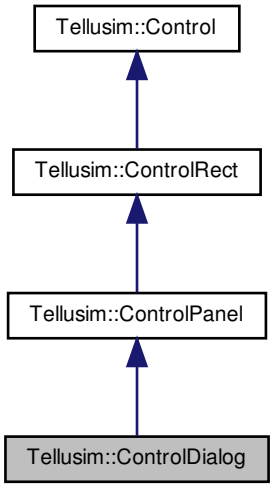
5.49 Tellusim::ControlDialog Class Reference

```
#include <interface/TellusimControls.h>
```

Inheritance diagram for Tellusim::ControlDialog:



Collaboration diagram for Tellusim::ControlDialog:



Public Types

- using UpdatedCallback = Function< void(ControlDialog)>

  updated callback

Public Member Functions

- **ControlDialog** (Control *parent)
- **ControlDialog** (Control *parent, uint32_t columns)
- **ControlDialog** (Control *parent, uint32_t columns, float32_t x, float32_t y)
- void setConstrained (bool constrained)

  constrained flag
- bool **isConstrained** () const
- void setResizable (bool resizable)

  resizable flag
- bool **isResizable** () const
- void setMoveable (bool moveable)

  moveable flag
- bool **isMoveable** () const
- void setResizeArea (float32_t area)

  resize area
- float32_t **getResizeArea** () const
- Align getResizeAlign () const

  resize alignment
- bool **hasResizeAlign** (Align align) const
- bool **hasResizeAligns** (Align aligns) const
- void setMousePosition (const Vector2f &position)

  mouse position
- const Vector2f & **getMousePosition** () const
- void **setUpdatedCallback** (const UpdatedCallback &func)
- UpdatedCallback **getUpdatedCallback** () const
- bool **isUpdated** ()

Additional Inherited Members

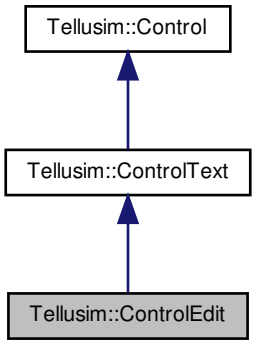
5.49.1 Detailed Description

The ControlDialog class is a specialized UI container derived from ControlPanel, designed to represent movable and resizable dialog windows. It supports features like constraint within parent bounds, resizability with customizable resize area, and movability through mouse interaction. These capabilities make it ideal for creating flexible, interactive dialogs in a GUI environment, such as tool windows or popup panels.

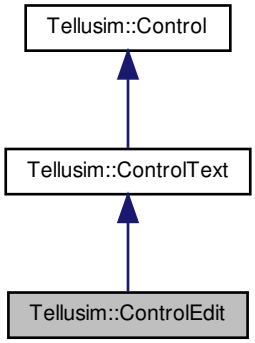
5.50 Tellusim::ControlEdit Class Reference

```
#include <interface/TellusimControls.h>
```

Inheritance diagram for Tellusim::ControlEdit:



Collaboration diagram for Tellusim::ControlEdit:



Public Types

- enum EditMode {
  EditModeText,
  EditModePassword,
  EditModeNumber,
  EditModeSigned,
  EditModeUnsigned,
  EditModeHexadecimal }

  edit mode
- using ClickedCallback = Function< void(ControlEdit)>

  clicked callback
- using ChangedCallback = Function< void(ControlEdit)>

  changed callback
- using ReturnedCallback = Function< void(ControlEdit)>

  returned callback

Public Member Functions

- **ControlEdit** (Control *parent)
- **ControlEdit** (Control *parent, const char *text)
- **ControlEdit** (Control *parent, const String &text)
- void setFrame (bool frame)

  frame flag
- bool **getFrame** () const
- void setBackground (bool background)

  background flag
- bool **getBackground** () const
- void setEditColor (const Color &color)

  edit color
- const Color & **getEditColor** () const
- void **setEditMode** (EditMode mode)
- EditMode **getEditMode** () const
- void setPasswordCode (uint32_t code)

  password code
- uint32_t **getPasswordCode** () const
- uint32_t getNumCodes () const

  edit codes
- const uint32_t * **getCodes** () const
- void setCurrentIndex (uint32_t index, uint32_t selection_index=Maxu32)

  current index
- uint32_t **getCurrentIndex** () const
- uint32_t **getSelectionIndex** () const
- void setSelection (bool current=false, bool changed=false)

  selected text
- void **clearSelection** ()
- String **getSelectedText** () const
- bool updateKeyboard (ControlRoot &root, uint32_t key, uint32_t code)

  update control
- void **setClickedCallback** (const ClickedCallback &func)
- ClickedCallback **getClickedCallback** () const
- bool **isClicked** ()

- void **setChangedCallback** (const ChangedCallback &func)
- ChangedCallback **getChangedCallback** () const
- bool **isChanged** ()
- void **setReturnedCallback** (const ReturnedCallback &func)
- ReturnedCallback **getReturnedCallback** () const
- bool **isReturned** ()
- CanvasMesh getCanvasMesh ()

  canvas elements

Additional Inherited Members

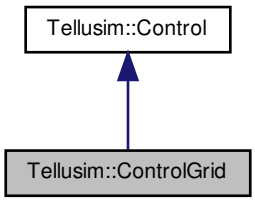
5.50.1 Detailed Description

The ControlEdit class provides a user interface control for text input, supporting various modes like plain text, password, numeric, and hexadecimal input. It allows for customizable features like frame and background visibility, edit color, and password code. The class supports multiple text selection and manipulation functionalities, including setting and getting the current index, selection, and selected text. It also allows interaction through callbacks for events such as clicks, changes, and text returns. Additionally, the control can be updated based on keyboard input, enabling real-time interaction in user interfaces. Ideal for scenarios that require dynamic text input and modification, such as forms or password fields.

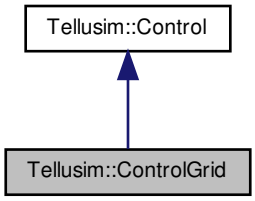
5.51 Tellusim::ControlGrid Class Reference

`#include <interface/TellusimControls.h>`

Inheritance diagram for Tellusim::ControlGrid:



Collaboration diagram for Tellusim::ControlGrid:



Public Member Functions

- **ControlGrid** (Control *parent)
- **ControlGrid** (Control *parent, uint32_t columns)
- **ControlGrid** (Control *parent, uint32_t columns, float32_t x, float32_t y=0.0f)
- void setColumns (uint32_t columns)

  number of columns
- uint32_t **getColumns** () const
- void setSpacing (const Vector2f &spacing)

  grid spacing
- void **setSpacing** (float32_t x, float32_t y)
- const Vector2f & **getSpacing** () const
- void setColumnRatio (uint32_t index, float32_t ratio)

  grid column ratio
- float32_t **getColumnRatio** (uint32_t index) const
- const Vector2f & getControlsSize () const

  controls size

Additional Inherited Members

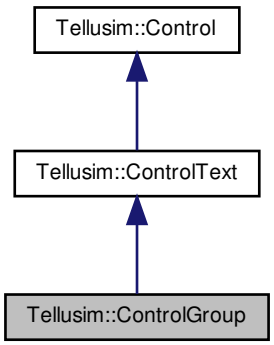
5.51.1 Detailed Description

The ControlGrid class is a layout control derived from Control that arranges child controls in a grid structure with configurable columns and spacing. It allows dynamic adjustment of the number of columns, spacing between items, and individual column width ratios to create flexible UI layouts. The class provides methods to retrieve the size occupied by all child controls, making it useful for auto-layout and responsive interface design.

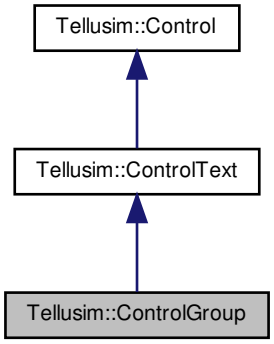
5.52 Tellusim::ControlGroup Class Reference

`#include <interface/TellusimControls.h>`

Inheritance diagram for Tellusim::ControlGroup:



Collaboration diagram for Tellusim::ControlGroup:



Public Types

- using ClickedCallback = Function< void(ControlGroup)>
  clicked callback

Public Member Functions

- **ControlGroup** (Control *parent, bool above=false)
- **ControlGroup** (Control *parent, const char *text, bool above=false)
- **ControlGroup** (Control *parent, const String &text, bool above=false)
- **ControlGroup** (Control *parent, const char *text, uint32_t columns, bool above=false)
- **ControlGroup** (Control *parent, const char *text, uint32_t columns, float32_t x, float32_t y, bool above=false)
- void setAbove (bool above, bool text=true)
  above flag
- bool **isAbove** () const
- bool **isBelow** () const
- void setFoldable (bool foldable)
  foldable flag
- bool **isFoldable** () const
- void setExpanded (bool expanded)
  expanded flag
- bool **isExpanded** () const
- void setBackground (bool background)
  background flag
- bool **getBackground** () const
- void setGroupRadius (float32_t radius)
  group radius
- float32_t **getGroupRadius** () const
- void setGroupColor (const Color &color)
  group color
- const Color & **getGroupColor** () const
- void setStrokeStyle (const StrokeStyle &style)

stroke style

- const StrokeStyle & **getStrokeStyleConst** () const
- const StrokeStyle & **getStrokeStyle** () const
- StrokeStyle & **getStrokeStyle** ()
- void setGradientStyle (const GradientStyle &style)

  gradient style
- const GradientStyle & **getGradientStyleConst** () const
- const GradientStyle & **getGradientStyle** () const
- GradientStyle & **getGradientStyle** ()
- void setFoldedText (const char *text)

  folded text
- void **setFoldedText** (const String &text)
- String **getFoldedText** () const
- void setExpandedText (const char *text)

  expanded text
- void **setExpandedText** (const String &text)
- String **getExpandedText** () const
- void setColumns (uint32_t columns)

  number of columns
- uint32_t **getColumns** () const
- void setSpacing (const Vector2f &spacing)

  grid spacing
- void **setSpacing** (float32_t x, float32_t y)
- const Vector2f & **getSpacing** () const
- void setColumnRatio (uint32_t index, float32_t ratio)

  grid column ratio
- float32_t **getColumnRatio** (uint32_t index) const
- const Vector2f & getControlsSize () const

  controls size
- void **setClickedCallback** (const ClickedCallback &func)
- ClickedCallback **getClickedCallback** () const
- bool **isClicked** ()
- CanvasRect getCanvasRect ()

  canvas elements

Additional Inherited Members

5.52.1 Detailed Description

The ControlGroup class is a composite UI control derived from ControlText that groups multiple controls together with a labeled header, optional folding behavior, and customizable styling. It supports layout features similar to a grid, including configurable columns, spacing, and per-column ratios. A ControlGroup can be rendered either above or below its children, optionally with a background and rounded borders.

5.53 Tellusim::Controller Class Reference

```
#include <system/TellusimController.h>
```

Public Types

- enum Stick {
  StickLeft = 0,
  StickRight,
  NumSticks }

  Controller sticks.
- enum Axis {
  AxisX = 0,
  AxisY,
  AxisZ,
  AxisRX,
  AxisRY,
  AxisRZ,
  NumAxes = 16 }

  Controller axes.
- enum Button {
  ButtonLeft = 0,
  ButtonRight,
  ButtonDown,
  ButtonUp,
  ButtonHome,
  ButtonShoulderLeft,
  ButtonShoulderRight,
  ButtonTriggerLeft,
  ButtonTriggerRight,
  ButtonStickLeft,
  ButtonStickRight,
  ButtonA,
  ButtonB,
  ButtonX,
  ButtonY,
  ButtonView,
  ButtonMenu,
  ButtonL1 = ButtonShoulderLeft,
  ButtonR1,
  ButtonL2,
  ButtonR2,
  ButtonL3,
  ButtonR3,
  ButtonCross,
  ButtonCircle,
  ButtonSquare,
  ButtonTriangle,
  ButtonShare,
  ButtonOptions,
  NumButtons = 32 }

  Controller buttons.
- enum Motor {
  MotorLow = 0,
  MotorHigh,
  NumMotors }

  Controller motors.
- using ButtonPressedCallback = Function< void(Controller controller, Button button)>

  button pressed callback
- using ButtonReleasedCallback = Function< void(Controller controller, Button button)>

  button released callback

- using ConnectedCallback = Function< void(Controller controller)>

  connected callback
- using DisconnectedCallback = Function< void(Controller controller)>

  disconnected callback

Public Member Functions

- **Controller** (uint32_t index)
- **Controller** (Type type, uint32_t index=Maxu32)
- void setType (Type type)

  controller type
- Type **getType** () const
- const char * **getTypeName** () const
- bool **isUnknown** () const
- bool **isJoystick** () const
- bool **isGamePad** () const
- bool **isWheel** () const
- void setIndex (uint32_t index)

  controller index
- uint32_t **getIndex** () const
- void setName (const char *name)

  controller name
- String **getName** () const
- void setModel (const char *model)

  controller model
- String **getModel** () const
- bool isConnected () const

  check controller
- bool **wasConnected** () const
- bool connect (const char *name=nullptr)

  connect controller
- void **release** ()
- void setStickName (Stick stick, const char *name)

  stick name
- String **getStickName** (Stick stick) const
- Stick **findStick** (const char *name) const
- void setStick (Stick stick, float32_t x, float32_t y)

  stick value
- float32_t **getStickX** (Stick stick) const
- float32_t **getStickY** (Stick stick) const
- void setAxisName (Axis axis, const char *name)

  axis name
- String **getAxisName** (Axis axis) const
- Axis **findAxis** (const char *name) const
- void setAxis (Axis axis, float32_t value)

  axis value
- float32_t **getAxis** (Axis axis) const
- void setButtonName (Button button, const char *name)

  button name
- String **getButtonName** (Button button) const
- Button **findButton** (const char *name) const
- void setButton (Button button, bool value)

button state

- bool **getButton** (Button button, bool clear=false) const
- void setButtonValue (Button button, float32_t value)

  button value
- float32_t **getButtonValue** (Button button) const
- void setMotorName (Motor motor, const char *name)

  motor name
- String **getMotorName** (Motor motor) const
- Motor **findMotor** (const char *name) const
- void setMotor (Motor motor, float32_t value)

  motor state
- float32_t **getMotor** (Motor motor) const
- void **setButtonPressedCallback** (const ButtonPressedCallback &func)
- ButtonPressedCallback **getButtonPressedCallback** () const
- void **setButtonReleasedCallback** (const ButtonReleasedCallback &func)
- ButtonReleasedCallback **getButtonReleasedCallback** () const
- void **setConnectedCallback** (const ConnectedCallback &func)
- ConnectedCallback **getConnectedCallback** () const
- void **setDisconnectedCallback** (const DisconnectedCallback &func)
- DisconnectedCallback **getDisconnectedCallback** () const

Static Public Member Functions

- static uint32_t getNumControllers ()

  all controllers
- static uint32_t **findController** (const char *name)
- static Controller **getController** (uint32_t index)
- static void update ()

  update controllers
- static const char * **getTypeName** (Type type)

Static Public Attributes

- static const char * NameXbox

  controller names
- static const char * **NamePlayStation**
- static const char * **NameNintendo**

5.53.1 Detailed Description

The Controller class provides cross-platform methods to interact with various types of controllers such as joysticks, gamepads, and steering wheels. It supports querying and setting controller properties such as type, name, model, buttons, sticks, axes, and motors. It also allows you to check the connection status of the controller and register callbacks for button presses, releases, and connection events. The class allows for managing multiple controllers, retrieving controller details by index, and interacting with buttons, axes, and motors by name or value. It also supports the use of custom callbacks for button events and connection changes. The class includes static methods for querying and managing all connected controllers.

5.53.2 Member Enumeration Documentation

5.53.2.1 Button

`enum Tellusim::Controller::Button`

Controller buttons.

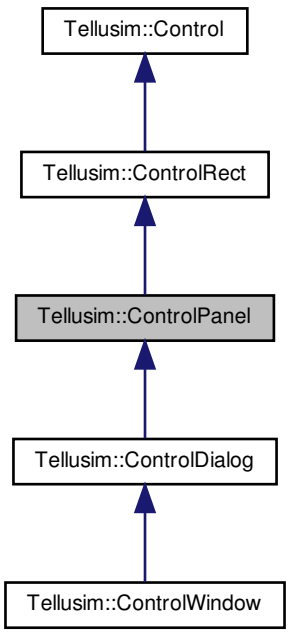
Enumerator

ButtonLeft	D-Pad.
ButtonHome	Common buttons.
ButtonShoulderLeft	Xbox buttons.
ButtonL1	PlayStation buttons.

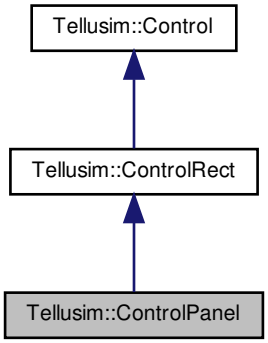
5.54 Tellusim::ControlPanel Class Reference

```
#include <interface/TellusimControls.h>
```

Inheritance diagram for Tellusim::ControlPanel:



Collaboration diagram for Tellusim::ControlPanel:



Public Member Functions

- **ControlPanel** (Control *parent)
- **ControlPanel** (Control *parent, uint32_t columns)
- **ControlPanel** (Control *parent, uint32_t columns, float32_t x, float32_t y)
- void setColumns (uint32_t columns)

  number of columns
- uint32_t **getColumns** () const
- void setSpacing (const Vector2f &spacing)

  grid spacing
- void **setSpacing** (float32_t x, float32_t y)
- const Vector2f & **getSpacing** () const
- void setColumnRatio (uint32_t index, float32_t ratio)

  grid column ratio
- float32_t **getColumnRatio** (uint32_t index) const
- const Vector2f & getControlsSize () const

  controls size

Additional Inherited Members

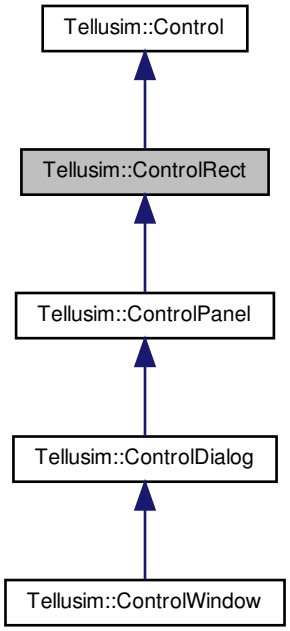
5.54.1 Detailed Description

The ControlPanel class is a container UI control derived from ControlRect that arranges child controls in a structured, grid-based layout. It allows customization of the number of columns, spacing between elements, and the relative width of each column using column ratios. As a rectangular control, it inherits visual styling options such as background color, radius, and texture support, making it suitable for organizing and grouping controls within a visually distinct section of a user interface.

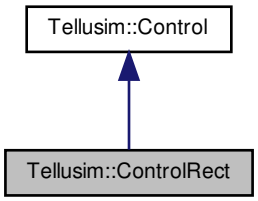
5.55 Tellusim::ControlRect Class Reference

`#include <interface/TellusimControls.h>`

Inheritance diagram for Tellusim::ControlRect:



Collaboration diagram for Tellusim::ControlRect:



Public Types

- using InsideCallback = Function< bool(ControlRect, float32_t x, float32_t y)>
  inside callback

- using PressedCallback = Function< void(ControlRect, float32_t x, float32_t y)>

  pressed callback
- using ReleasedCallback = Function< void(ControlRect, float32_t x, float32_t y)>

  released callback
- using ClickedCallback = Function< void(ControlRect)>

  clicked callback

Public Member Functions

- **ControlRect** (Control *parent)
- **ControlRect** (Control *parent, Texture &texture)
- **ControlRect** (Control *parent, const char *name)
- **ControlRect** (Control *parent, CanvasElement::Mode mode)
- void setCallback (bool callback)

  callback flag
- bool **getCallback** () const
- void setFullscreen (bool fullscreen)

  fullscreen flag
- bool **isFullscreen** () const
- void setMode (CanvasElement::Mode mode)

  control mode
- CanvasElement::Mode **getMode** () const
- void setPipeline (Pipeline &pipeline)

  control pipeline
- void **setPipeline** (Pipeline &pipeline, const CanvasElement::DrawCallback &func)
- Pipeline **getPipeline** () const
- void setRadius (float32_t radius)

  control radius
- float32_t **getRadius** () const
- void setColor (const Color &color)

  control color
- void **setColor** (float32_t r, float32_t g, float32_t b, float32_t a)
- const Color & **getColor** () const
- void setStrokeStyle (const StrokeStyle &style)

  stroke style
- const StrokeStyle & **getStrokeStyleConst** () const
- const StrokeStyle & **getStrokeStyle** () const
- StrokeStyle & **getStrokeStyle** ()
- void setGradientStyle (const GradientStyle &style)

  gradient style
- const GradientStyle & **getGradientStyleConst** () const
- const GradientStyle & **getGradientStyle** () const
- GradientStyle & **getGradientStyle** ()
- void setMipmap (float32_t mipmap)

  control mipmap
- float32_t **getMipmap** () const
- void setFilter (Sampler::Filter filter)

  filter mode
- Sampler::Filter **getFilter** () const
- void **setAnisotropy** (uint32_t anisotropy)
- uint32_t **getAnisotropy** () const
- void setWrapMode (Sampler::WrapMode mode)

wrapping mode

- Sampler::WrapMode **getWrapMode** () const
- void setBlend (Pipeline::BlendOp op, Pipeline::BlendFunc src, Pipeline::BlendFunc dest)

blending parameters

- Pipeline::BlendOp **getBlendOp** () const
- Pipeline::BlendFunc **getBlendSrcFunc** () const
- Pipeline::BlendFunc **getBlendDestFunc** () const
- void setTexture (Texture &texture, bool linear=false)

texture pointer

- Texture **getTexture** () const
- bool **getTextureLinear** () const
- void setTextureName (const char *name)

texture name

- void **setTextureName** (const String &name)
- String **getTextureName** () const
- void setTextureScale (float32_t scale_x, float32_t scale_y)

texture scale

- float32_t **getTextureScaleX** () const
- float32_t **getTextureScaleY** () const
- void setTextureFlip (bool flip_x, bool flip_y)

texture orientation

- bool **getTextureFlipX** () const
- bool **getTextureFlipY** () const
- void setTextureProj (bool projection)

texture projection

- bool **getTextureProj** () const
- void setTexCoord (const Rect &texcoord)

texture coordinates

- void **setTexCoord** (float32_t left, float32_t right, float32_t bottom, float32_t top)
- const Rect & **getTexCoord** () const
- void **setInsideCallback** (const InsideCallback &func)
- InsideCallback **getInsideCallback** () const
- void **setPressedCallback** (const PressedCallback &func)
- PressedCallback **getPressedCallback** () const
- void **setReleasedCallback** (const ReleasedCallback &func)
- ReleasedCallback **getReleasedCallback** () const
- void **setClickedCallback** (const ClickedCallback &func)
- void **setClicked2Callback** (const ClickedCallback &func)
- void **setClickedRightCallback** (const ClickedCallback &func)
- ClickedCallback **getClickedCallback** () const
- ClickedCallback **getClicked2Callback** () const
- ClickedCallback **getClickedRightCallback** () const
- CanvasRect getCanvasRect ()

canvas elements

- CanvasMesh **getCanvasMesh** ()

Additional Inherited Members

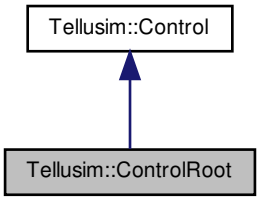
5.55.1 Detailed Description

The ControlRect class extends the Control class and is designed for managing rectangular UI elements with customizable properties such as color, texture, stroke style, and gradients. It provides methods for setting control modes, pipelines, filters, and wrapping modes, as well as controlling mipmaps and anisotropy for texture handling. The class supports texture properties like scale, flip, and projection, and allows for defining texture coordinates and names. Additionally, it includes various callbacks for interaction handling, such as inside, pressed, released, and clicked events, along with support for fullscreen mode, radius, and blending parameters. The ControlRect class also integrates with canvas elements, enabling the rendering of rectangular shapes and meshes with flexible visual configurations.

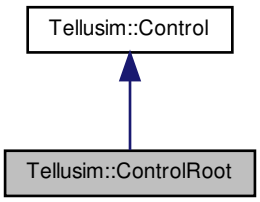
5.56 Tellusim::ControlRoot Class Reference

`#include <interface/TellusimControls.h>`

Inheritance diagram for Tellusim::ControlRoot:



Collaboration diagram for Tellusim::ControlRoot:



Public Types

- using CopyCallback = Function< void(ControlRoot, const char *text)>
  copy callback
- using PasteCallback = Function< String(ControlRoot)>
  paste callback

Public Member Functions

- **ControlRoot** (Canvas &canvas, bool blob=false)
- void setViewport (const Viewport &viewport)

  root viewport
- void **setViewport** (uint32_t width, uint32_t height)
- void **setViewport** (float32_t width, float32_t height)
- const Viewport & **getViewport** () const
- String getFontName () const

  font name
- void **setFontName** (const char *name)
- void **setFontName** (const String &name)
- bool **setFontBlob** (const uint8_t(*blob)[256], const char *name=nullptr)
- bool setFontSize (uint32_t size, bool update=false)

  font style
- uint32_t **getFontSize** () const
- bool **setFontScale** (uint32_t scale, bool update=false)
- uint32_t **getFontScale** () const
- bool **setFontStyle** (const FontStyle &style, bool update=false)
- const FontStyle & **getFontStyle** () const
- FontStyle & **getFontStyle** ()
- String getTextureName () const

  texture parameters
- bool **setTextureName** (const char *name, uint32_t width=0, uint32_t height=0, float32_t border=0.0f)
- bool **setTextureName** (const String &name, uint32_t width=0, uint32_t height=0, float32_t border=0.0f)
- bool **setTextureBlob** (const uint8_t(*blob)[256], const char *name=nullptr, uint32_t width=0, uint32_t height=0, float32_t border=0.0f)
- float32_t **getTextureWidth** () const
- float32_t **getTextureHeight** () const
- void setTextColor (Type type, State state, const Color &color)

  text parameters
- void **setTextOffset** (Type type, State state, const Vector3f &offset)
- const Color & **getTextColor** (Type type, State state) const
- const Vector3f & **getTextOffset** (Type type, State state) const
- void setMeshColor (Mesh mesh, State state, const Color &color)

  mesh parameters
- void **setMeshRegion** (Mesh mesh, const Rect &grid, const Rect &region, const Vector2f &border)
- void **setMeshRegions** (Mesh mesh, const Rect &grid, const Rect &regions, const Vector2f &border)
- uint32_t **getMeshColor** (Mesh mesh, State state) const
- const Rect & **getMeshGrid** (Mesh mesh) const
- const Rect & **getMeshMargin** (Mesh mesh) const
- const Rect & **getMeshRegion** (Mesh mesh, State state) const
- void setGroupRadius (float32_t radius)

  panel parameters
- void **setGroupColor** (const Color &color)
- float32_t **getGroupRadius** () const
- const Color & **getGroupColor** () const
- void setPanelRadius (float32_t radius)

  panel parameters
- void **setPanelColor** (const Color &color)
- float32_t **getPanelRadius** () const
- const Color & **getPanelColor** () const
- void setCheckedColor (const Color &color)

check parameters

- const Color & **getCheckedColor** () const
- void setSplitSize (float32_t size)

split parameters

- float32_t **getSplitSize** () const
- void setMouse (int32_t x, int32_t y, Button buttons)

mouse button

- void **setMouse** (float32_t x, float32_t y, Button buttons)
- const Vector2f & **getMouse** () const
- Button **getMouseButtons** () const
- float32_t **getMouseX** () const
- float32_t **getMouseY** () const
- void setMouseOffset (const Vector2f &offset)

mouse offset

- const Vector2f & **getMouseOffset** () const
- void setMouseAxis (Axis axis, float32_t delta)

mouse axes

- void setMouseAlign (Align align, bool clear=true)

mouse alignment

- Align **getMouseAlign** () const
- bool **hasMouseAlign** (Align align) const
- bool **hasMouseAligns** (Align aligns) const
- bool setKeyboardKey (uint32_t key, uint32_t code, bool value)

keyboard keys

- bool **getKeyboardKey** (uint32_t key, bool clear=false)
- void clearCurrentControl ()

current control

- void **setCurrentControl** (Control control, bool grab=false)
- Control **getCurrentControl** () const
- bool **isCurrentControl** () const
- bool **getControlGrab** () const
- void clearFocusedControl ()

focused control

- void **setFocusedControl** (Control control)
- Control **getFocusedControl** () const
- bool **isFocusedControl** () const
- void clearMouseControl ()

mouse control

- void **setMouseControl** (Control control)
- Control **getMouseControl** () const
- bool **isMouseControl** () const
- void clearInputControl ()

input control

- void **setInputControl** (Control control)
- Control **getInputControl** () const
- bool **isInputControl** () const
- void clearModalControl ()

modal control

- void **setModalControl** (Control control, bool disabled=true)
- Control **getModalControl** () const
- bool **isModalDisabled** () const
- bool **isModalControl** () const
- void setOverlayOrder (int32_t order)

overlay order

- int32_t **getOverlayOrder** () const
- bool update (uint32_t scale=0, int32_t order=0)

  update controls
- void setCopyText (const char *text)

  copy/paste buffer
- void **setCopyText** (const String &text)
- String **getPasteText** ()
- void **setCopyCallback** (const CopyCallback &func)
- CopyCallback **getCopyCallback** () const
- void **setPasteCallback** (const PasteCallback &func)
- PasteCallback **getPasteCallback** () const

Additional Inherited Members

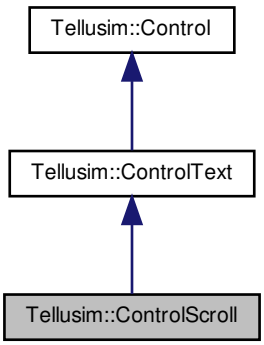
5.56.1 Detailed Description

The ControlRoot class extends the Control class and manages the root of a UI hierarchy, providing functions for setting viewport dimensions, font, texture, and mesh properties. It allows control over text color and offsets, as well as panel and group properties like radius and color. The class supports advanced input handling, including mouse position and button tracking, keyboard key events, and control focus management. It offers functionalities for managing the current, focused, modal, and mouse-controlled elements in the UI. Additionally, it includes copy and paste functionality, with user-defined callback functions for these operations.

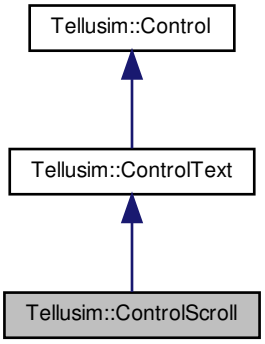
5.57 Tellusim::ControlScroll Class Reference

```
#include <interface/TellusimControls.h>
```

Inheritance diagram for Tellusim::ControlScroll:



Collaboration diagram for Tellusim::ControlScroll:



Public Types

- using ClickedCallback = Function< void(ControlScroll)>
  clicked callback
- using ChangedCallback = Function< void(ControlScroll)>
  changed callback

Public Member Functions

- **ControlScroll** (Control *parent, bool vertical=false)
- **ControlScroll** (Control *parent, float64_t value, bool vertical=false)
- **ControlScroll** (Control *parent, float64_t value, float64_t frame, float64_t range, bool vertical=false)
- **ControlScroll** (Control *parent, uint32_t value, uint32_t frame, uint32_t range, bool vertical=false)
- **ControlScroll** (Control *parent, int32_t value, int32_t frame, int32_t range, bool vertical=false)
- void setVertical (bool vertical, bool text=true)
  vertical flag
- bool **isHorizontal** () const
- bool **isVertical** () const
- void setPrevText (const char *text)
  scroll previous text
- void **setPrevText** (const String &text)
- String **getPrevText** () const
- void setNextText (const char *text)
  scroll next text
- void **setNextText** (const String &text)
- String **getNextText** () const
- void setScrollColor (const Color &color)
  scroll color
- const Color & **getScrollColor** () const
- void setStep (float64_t step)
  scroll step
- float64_t **getStep** () const

- void setValue (float64_t value, bool callback=false)

  scroll value
- float64_t **getValue** () const
- void setFrame (float64_t frame)

  scroll frame
- float64_t **getFrame** () const
- void setRange (float64_t range)

  scroll range
- float64_t **getRange** () const
- void setFrameAlign (Align align)

  frame alignment
- Align **getFrameAlign** () const
- bool **hasFrameAlign** (Align align) const
- bool **hasFrameAligns** (Align aligns) const
- void **setClickedCallback** (const ClickedCallback &func)
- ClickedCallback **getClickedCallback** () const
- bool **isClicked** ()
- void **setChangedCallback** (const ChangedCallback &func)
- ChangedCallback **getChangedCallback** () const
- bool **isChanged** (bool clear=true)
- CanvasMesh getCanvasMesh ()

  canvas elements

Additional Inherited Members

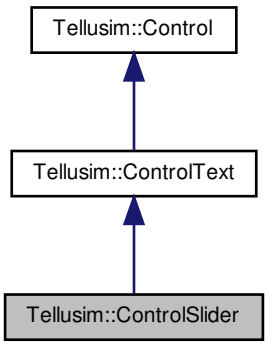
5.57.1 Detailed Description

The ControlScroll class represents a scroll control, typically used for scrolling content or adjusting values within a specified range. It can be configured to be either vertical or horizontal and supports customization of scroll step, color, range, and frame size.

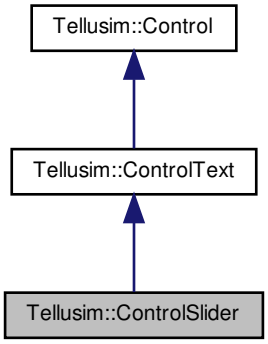
5.58 Tellusim::ControlSlider Class Reference

```
#include <interface/TellusimControls.h>
```

Inheritance diagram for Tellusim::ControlSlider:



Collaboration diagram for Tellusim::ControlSlider:



Public Types

- using FormatCallback = Function< String(ControlSlider)>
  format callback
- using PressedCallback = Function< void(ControlSlider)>
  pressed callback
- using ReleasedCallback = Function< void(ControlSlider)>
  released callback
- using ClickedCallback = Function< void(ControlSlider)>
  clicked callback
- using ChangedCallback = Function< void(ControlSlider)>
  changed callback

Public Member Functions

- **ControlSlider** (Control *parent)
- **ControlSlider** (Control *parent, const char *text)
- **ControlSlider** (Control *parent, const String &text)
- **ControlSlider** (Control *parent, const char *text, uint32_t digits)
- **ControlSlider** (Control *parent, const char *text, uint32_t digits, float64_t value)
- **ControlSlider** (Control *parent, const char *text, uint32_t digits, float64_t value, float64_t min, float64_t max)
- **ControlSlider** (Control *parent, const char *text, float64_t value, float64_t min, float64_t max)
- **ControlSlider** (Control *parent, const char *text, uint32_t value, uint32_t min, uint32_t max)
- **ControlSlider** (Control *parent, const char *text, int32_t value, int32_t min, int32_t max)
- void setConstrained (bool constrained)
  constrained flag
- bool **isConstrained** () const
- void setTextEnabled (bool enabled)
  text enabled flag
- bool **isTextEnabled** () const
- void setSliderColor (const Color &color)

slider color

- const Color & **getSliderColor** () const
- void setDigits (uint32_t digits)

slider digits

- uint32_t **getDigits** () const
- void setStep (float64_t step)

slider step

- float64_t **getStep** () const
- void setBase (float64_t base)

exponent base

- float64_t **getBase** () const
- void setFormat (const char *format)

slider format

- void **setFormat** (const String &format)
- String **getFormat** () const
- void setValue (float64_t value, bool callback=false, bool exponent=false)
- float64_t **getValue** (bool exponent=false) const
- float32_t **getValuef32** (bool exponent=false) const
- uint32_t **getValueu32** (bool exponent=false) const
- int32_t **getValuei32** (bool exponent=false) const
- void setRange (float64_t min, float64_t max, bool exponent=false)
- float64_t **getMinRange** (bool exponent=false) const
- float64_t **getMaxRange** (bool exponent=false) const
- void setHandleSize (float32_t size)

handle size

- float32_t **getHandleSize** () const
- void **setFormatCallback** (const FormatCallback &func)
- FormatCallback **getFormatCallback** () const
- void **setPressedCallback** (const PressedCallback &func)
- PressedCallback **getPressedCallback** () const
- bool **isPressed** ()
- void **setReleasedCallback** (const ReleasedCallback &func)
- ReleasedCallback **getReleasedCallback** () const
- bool **isReleased** ()
- void **setClickedCallback** (const ClickedCallback &func)
- void **setClicked2Callback** (const ClickedCallback &func)
- void **setClickedRightCallback** (const ClickedCallback &func)
- ClickedCallback **getClickedCallback** () const
- ClickedCallback **getClicked2Callback** () const
- ClickedCallback **getClickedRightCallback** () const
- bool **isClicked** ()
- void **setChangedCallback** (const ChangedCallback &func)
- ChangedCallback **getChangedCallback** () const
- bool **isChanged** (bool clear=true)
- CanvasMesh getCanvasMesh ()

canvas elements

Additional Inherited Members

5.58.1 Detailed Description

The ControlSlider class represents a slider control that allows the user to select a value within a specified range. It offers customization options such as setting the slider color, digits, step size, and base for exponential conversion. The class supports constraints, text enablement, and custom formatting for value display.

5.58.2 Member Function Documentation

5.58.2.1 setValue()

```
void Tellusim::ControlSlider::setValue (
            float64_t value,
            bool callback = false,
            bool exponent = false )
```

slider value

Parameters

callback	Run changed callback on value change.
exponent	perform exponential conversion.

5.58.2.2 setRange()

```
void Tellusim::ControlSlider::setRange (
            float64_t min,
            float64_t max,
            bool exponent = false )
```

slider range

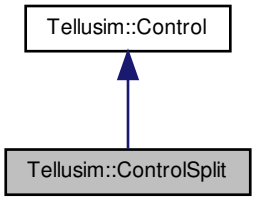
Parameters

exponent	perform exponential conversion.

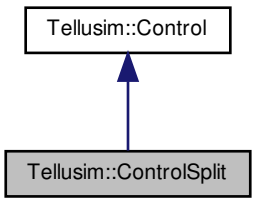
5.59 Tellusim::ControlSplit Class Reference

`#include <interface/TellusimControls.h>`

Inheritance diagram for Tellusim::ControlSplit:



Collaboration diagram for Tellusim::ControlSplit:



Public Member Functions

- **ControlSplit** (Control *parent, bool vertical=false)
- **ControlSplit** (Control *parent, float32_t value, bool vertical=false)
- void setAbsolute (bool absolute)

  absolute flag
- bool **isAbsolute** () const
- void setVertical (bool vertical)

  vertical flag
- bool **isHorizontal** () const
- bool **isVertical** () const
- void setValue (float32_t value)

  split value
- float32_t **getValue** () const
- void setHandleSize (float32_t size)

  handle size
- float32_t **getHandleSize** () const
- const Vector2f & getControlsSize () const

  controls size

Additional Inherited Members

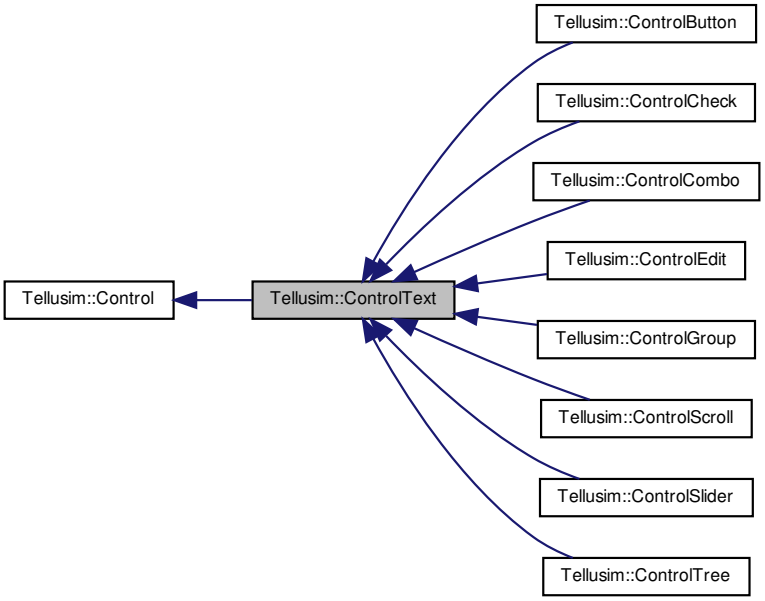
5.59.1 Detailed Description

The ControlSplit class represents a split control, often used for creating resizable panels or dividers within a user interface. It supports both vertical and horizontal orientations, with the ability to configure whether the split is absolute or relative. This control is useful for creating adjustable layouts in applications with multiple sections or panels.

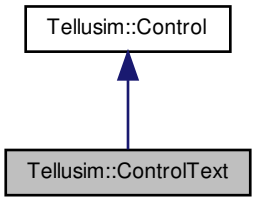
5.60 Tellusim::ControlText Class Reference

`#include <interface/TellusimControls.h>`

Inheritance diagram for Tellusim::ControlText:



Collaboration diagram for Tellusim::ControlText:



Public Member Functions

- **ControlText** (Control *parent)
- **ControlText** (Control *parent, const char *text)
- **ControlText** (Control *parent, const String &text)
- void setMode (CanvasElement::Mode mode)

control mode

- CanvasElement::Mode **getMode** () const
- void setPipeline (Pipeline &pipeline)

control pipeline

- void **setPipeline** (Pipeline &pipeline, const CanvasElement::DrawCallback &func)
- Pipeline **getPipeline** () const
- void setColor (const Color &color)

control color

- void **setColor** (float32_t r, float32_t g, float32_t b, float32_t a)
- const Color & **getColor** () const
- void setFilter (Sampler::Filter filter)

filter mode

- Sampler::Filter **getFilter** () const
- void **setAnisotropy** (uint32_t anisotropy)
- uint32_t **getAnisotropy** () const
- void setBlend (Pipeline::BlendOp op, Pipeline::BlendFunc src, Pipeline::BlendFunc dest)

blending parameters

- Pipeline::BlendOp **getBlendOp** () const
- Pipeline::BlendFunc **getBlendSrcFunc** () const
- Pipeline::BlendFunc **getBlendDestFunc** () const
- void setFontName (const char *name)

font name

- void **setFontName** (const String &name)
- String **getFontName** () const
- void setFontColor (const Color &color)

font color

- const Color & **getFontColor** () const
- bool setFontSize (uint32_t size)

font style

- uint32_t **getFontSize** () const
- bool **setFontStyle** (const FontStyle &style)
- const FontStyle & **getFontStyleConst** () const
- const FontStyle & **getFontStyle** () const
- FontStyle & **getFontStyle** ()
- void setFontAlign (Align align)

font alignment

- Align **getFontAlign** () const
- bool **hasFontAlign** (Align align) const
- bool **hasFontAligns** (Align aligns) const
- void setText (const char *text)

control text

- void **setText** (const String &text)
- String **getText** () const
- CanvasText getCanvasText ()

canvas elements

Additional Inherited Members

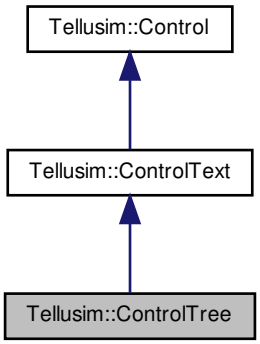
5.60.1 Detailed Description

The ControlText class extends the Control class and is responsible for managing text-based UI elements. It provides methods for setting and retrieving text, font properties such as size, style, color, and alignment, as well as control color and blending settings. The class allows for configuring control mode, pipeline settings, and filtering, including anisotropy and blending operations. Through its integration with canvas elements, the ControlText class enables precise control over the rendering of text and related visual attributes in the UI.

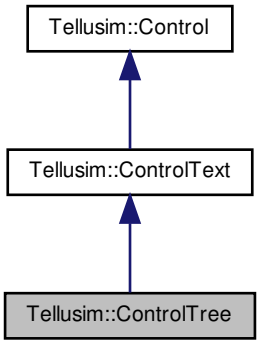
5.61 Tellusim::ControlTree Class Reference

`#include <interface/TellusimControls.h>`

Inheritance diagram for Tellusim::ControlTree:



Collaboration diagram for Tellusim::ControlTree:



Public Types

- using ChangedCallback = Function< void(ControlTree, uint32_t item)>
  changed callback
- using DraggedCallback = Function< bool(ControlTree, uint32_t item)>
  dragged callback
- using DroppedCallback = Function< void(ControlTree, uint32_t item)>
  dropped callback

- using ClickedCallback = Function< void(ControlTree, uint32_t item)>

  clicked callback
- using ExpandedCallback = Function< void(ControlTree, uint32_t item)>

  expanded callback
- using SelectedCallback = Function< void(ControlTree)>

  selected callback

Public Member Functions

- **ControlTree** (Control *parent)
- void setSelectable (bool selectable)

  selectable flag
- bool **isSelectable** () const
- void setMultiSelection (bool multi_selection)

  multi-selection flag
- bool **isMultiSelection** () const
- void setFoldedText (const char *text)

  folded text
- void **setFoldedText** (const String &text)
- String **getFoldedText** () const
- void setExpandedText (const char *text)

  expanded text
- void **setExpandedText** (const String &text)
- String **getExpandedText** () const
- void setTexture (Texture &texture, uint32_t rows=1, uint32_t columns=1)

  texture pointer
- Texture **getTexture** () const
- void setTextureName (const char *name, uint32_t rows=1, uint32_t columns=1)

  texture name
- void **setTextureName** (const String &name, uint32_t rows=1, uint32_t columns=1)
- String **getTextureName** () const
- void setTextureGrid (uint32_t rows, uint32_t columns)

  texture layout
- uint32_t **getTextureRows** () const
- uint32_t **getTextureColumns** () const
- void clearItems ()

  tree items
- uint32_t **addItem** (const char *text, uint32_t parent=Maxu32, bool expanded=true)
- uint32_t **addItem** (const String &text, uint32_t parent=Maxu32, bool expanded=true)
- void **addItems** (const InitializerList< const char *> &items, uint32_t parent=Maxu32)
- void **removeItem** (uint32_t item, bool children=false)
- void **viewItem** (uint32_t item)
- uint32_t **getNumItems** () const
- uint32_t **getItem** (uint32_t index) const
- bool switchItemHidden (uint32_t item, bool children=false)

  item hidden flag
- void **setItemHidden** (uint32_t item, bool hidden, bool children=false)
- bool **isItemHidden** (uint32_t item) const
- bool switchItemExpanded (uint32_t item, bool children=false)

  item expanded flag
- void **setItemExpanded** (uint32_t item, bool expanded, bool children=false)
- bool **isItemExpanded** (uint32_t item) const

- bool switchItemSelected (uint32_t item, bool children=false)

  item selected flag
- void **setItemSelected** (uint32_t item, bool selected, bool children=false)
- bool **isItemSelected** (uint32_t item) const
- void setItemParent (uint32_t item, uint32_t parent)

  item parent
- uint32_t **getItemParent** (uint32_t item) const
- bool **isItemParent** (uint32_t item, uint32_t parent, bool hierarchy=false) const
- void addItemChild (uint32_t item, uint32_t child)

  item children
- void **removeItemChild** (uint32_t item, uint32_t child)
- void **addItemChildren** (uint32_t item, const Array< uint32_t > &children)
- void **removeItemChildren** (uint32_t item, const Array< uint32_t > &children)
- uint32_t **findItemChild** (uint32_t item, uint32_t child) const
- bool **isItemChild** (uint32_t item, uint32_t child) const
- uint32_t **getNumItemChildren** (uint32_t item) const
- uint32_t **getItemChild** (uint32_t item, uint32_t index) const
- Array< uint32_t > **getItemChildren** (uint32_t item) const
- void setItemText (uint32_t item, const char *text)

  item text
- void **setItemText** (uint32_t item, const String &text)
- String **getItemText** (uint32_t item) const
- uint32_t **findItemText** (const char *text) const
- uint32_t **findItemText** (const String &text) const
- void setItemColor (uint32_t item, const Color &color)

  item color
- const Color & **getItemColor** (uint32_t item) const
- void setItemTexture (uint32_t item, uint32_t row, uint32_t column=0)

  item icon
- uint32_t **getItemTextureRow** (uint32_t item) const
- uint32_t **getItemTextureColumn** (uint32_t item) const
- void setItemData (uint32_t item, void *data)

  item data
- void * **getItemData** (uint32_t item) const
- uint32_t getFocusedItem () const

  focused item
- void setCurrentItem (uint32_t item, bool select=false, bool view=false, bool callback=false)

  current item
- uint32_t **getCurrentItem** () const
- String **getCurrentText** () const
- void setSelection ()

  selected items
- void **clearSelection** ()
- void **inverseSelection** ()
- uint32_t **getNumSelectedItems** () const
- uint32_t **getSelectedItem** (uint32_t index) const
- Array< uint32_t > **getSelectedItems** () const
- void **setChangedCallback** (const ChangedCallback &func)
- ChangedCallback **getChangedCallback** () const
- void **setDraggedCallback** (const DraggedCallback &func)
- DraggedCallback **getDraggedCallback** () const
- void **setDroppedCallback** (const DroppedCallback &func)
- DroppedCallback **getDroppedCallback** () const

- void **setClickedCallback** (const ClickedCallback &func)
- void **setClicked2Callback** (const ClickedCallback &func)
- void **setClickedRightCallback** (const ClickedCallback &func)
- ClickedCallback **getClickedCallback** () const
- ClickedCallback **getClicked2Callback** () const
- ClickedCallback **getClickedRightCallback** () const
- void **setExpandedCallback** (const ExpandedCallback &func)
- ExpandedCallback **getExpandedCallback** () const
- void **setSelectedCallback** (const SelectedCallback &func)
- SelectedCallback **getSelectedCallback** () const

Additional Inherited Members

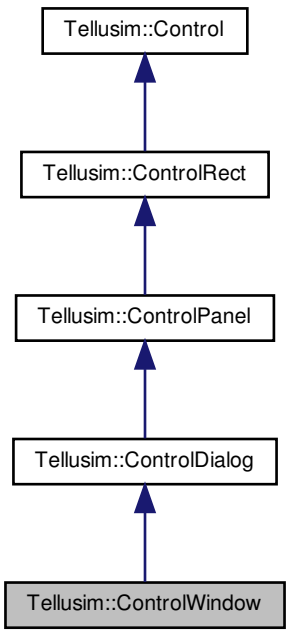
5.61.1 Detailed Description

The ControlTree class manages a hierarchical list or tree structure of items in a user interface, supporting various interaction features like selection, multi-selection, and item visibility. It allows items to be organized in a parent-child relationship, with functions to manipulate their text, color, icon, and custom data. The class also enables the folding and expansion of tree items, providing both a folded and expanded text representation. It offers texture management with configurable grid layouts and callbacks for user interactions, such as item selection, expansion, and dragging. This control is ideal for representing tree-like or linear structures, such as file explorers, menus, or categorized lists, with rich interaction capabilities.

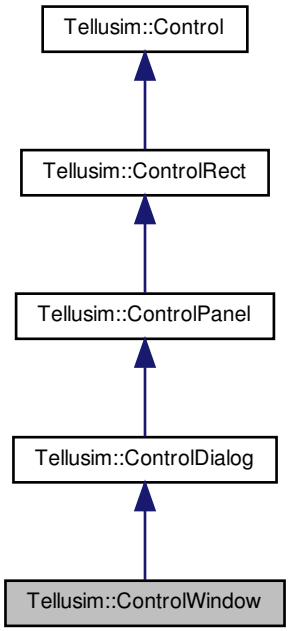
5.62 Tellusim::ControlWindow Class Reference

```
#include <interface/TellusimControls.h>
```

Inheritance diagram for Tellusim::ControlWindow:



Collaboration diagram for Tellusim::ControlWindow:



Public Member Functions

- **ControlWindow** (ControlRoot *root, Window &parent, Window &window)
- **ControlWindow** (ControlRoot *root, Window &parent, Window &window, uint32_t columns)
- **ControlWindow** (ControlRoot *root, Window &parent, Window &window, uint32_t columns, float32_t x, float32_t y)
- Window getParentWindow () const

  control windows
- Window **getDialogWindow** () const

Additional Inherited Members

5.62.1 Detailed Description

The ControlWindow class extends ControlDialog to integrate with native windowing, allowing UI controls to be embedded within or associated with system-level windows. Designed for complex UI applications, it combines dialog interaction features like resizing and movement with platform window management through a ControlRoot.

5.63 Tellusim::CubeFilter Class Reference

```
#include <graphics/TellusimCubeFilter.h>
```

Public Types

- enum Mode {
  ModeCube = 0,
  ModePanorama,
  NumModes }
  Filter modes.
- enum Flags {
  FlagCube = (1 << ModeCube),
  FlagPanorama = (1 << ModePanorama),
  FlagsAll = (FlagCube | FlagPanorama) }
  Filter flags.

Public Member Functions

- void clear ()
  clear filter
- bool isCreated (Mode mode) const
  check filter
- uint32_t getGroupSize () const
  filter parameters
- uint32_t **getMaxOrder** () const
- uint32_t **getMaxSize** () const
- uint32_t **getHarmonics** () const
- bool create (const Device &device, Mode mode, uint32_t order=3, uint32_t size=1024, uint32_t groups=256)
- bool **create** (const Device &device, Flags flags, uint32_t order=3, uint32_t size=1024, uint32_t groups=256)
- bool dispatch (Compute &compute, Buffer &buffer, uint32_t offset, Texture &texture, const Slice &slice) const
- bool **dispatch** (Compute &compute, Buffer &buffer, uint32_t offset, Texture &texture) const
- bool dispatch (Compute &compute, Texture &texture, const Slice &slice, Buffer &buffer, uint32_t offset) const
- bool **dispatch** (Compute &compute, Texture &texture, Buffer &buffer, uint32_t offset) const

5.63.1 Detailed Description

The CubeFilter class provides GPU-based filtering operations for cube and panorama textures, enabling the generation and reconstruction of spherical harmonics. Filters can be created with specific parameters and dispatched using compute shaders, either to generate coefficient buffers from cube textures or to reconstruct filtered textures from coefficients.

5.63.2 Member Function Documentation

5.63.2.1 create()

```
bool Tellusim::CubeFilter::create (
            const Device & device,
            Mode mode,
            uint32_t order = 3,
            uint32_t size = 1024,
            uint32_t groups = 256 )
```

create filter

Parameters

order	Maximum filter order (2, 3, 5).
size	Maximum filter size.
groups	Reduction group size.

5.63.2.2 dispatch() [1/2]

```
bool Tellusim::CubeFilter::dispatch (
             Compute & compute,
             Buffer & buffer,
             uint32_t offset,
             Texture & texture,
             const Slice & slice ) const
```

dispatch cube filter

Parameters

buffer	Destination coefficients buffer.
offset	Destination buffer offset.
texture	Source cube texture.
slice	Source texture slice.

5.63.2.3 dispatch() [2/2]

```
bool Tellusim::CubeFilter::dispatch (
             Compute & compute,
             Texture & texture,
             const Slice & slice,
             Buffer & buffer,
             uint32_t offset ) const
```

dispatch cube render

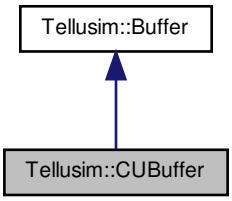
Parameters

texture	Destination cube texture (RGBAf16).
slice	Destination texture slice.
buffer	Source coefficients buffer.
offset	Source buffer offset.

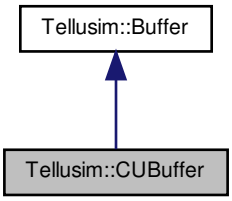
5.64 Tellusim::CUBuffer Class Reference

```
#include <platform/TellusimBuffer.h>
```

Inheritance diagram for Tellusim::CUBuffer:



Collaboration diagram for Tellusim::CUBuffer:



Public Member Functions

- size_t **getBufferPtr** () const
- uint8_t * **getBufferData** () const
- CUevent **getBufferEvent** () const
- uint32_t **getArrayFormat** () const
- uint32_t **getArrayChannels** () const
- CUexternalMemory **getSharedMemory** () const

Additional Inherited Members

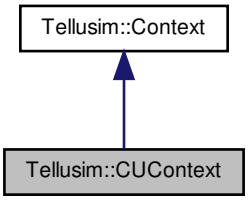
5.64.1 Detailed Description

The CUBuffer class is a CUDA-specific implementation of the Buffer class, providing access to CUDA buffer resources and memory management. It offers functionality to retrieve the buffer pointer, data, and event, as well as information about its array format and channels. This class also includes a method to access the shared memory for CUDA interoperation, enabling seamless integration with CUDA-based applications.

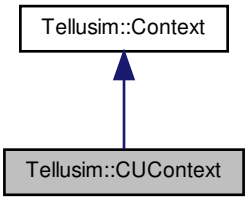
5.65 Tellusim::CUContext Class Reference

`#include <platform/TellusimContext.h>`

Inheritance diagram for Tellusim::CUContext:



Collaboration diagram for Tellusim::CUContext:



Public Member Functions

- int32_t getDevice () const

  current device
- CUcontext **getCUContext** () const
- CUstream **getStream** () const

Static Public Member Functions

- static void * getProcAddress (const char *name)

  Cuda functions.
- static bool error (uint32_t result)

  check Cuda errors

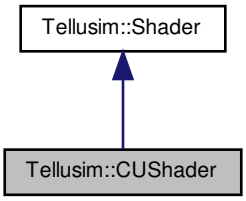
5.65.1 Detailed Description

The CUContext class is a CUDA-specific implementation of the Context class. It provides functionality to manage and interact with a CUDA context, stream, and device. The class allows retrieving the current CUDA device, context, and stream.

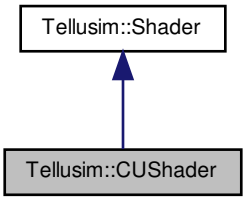
5.66 Tellusim::CUShader Class Reference

`#include <platform/TellusimShader.h>`

Inheritance diagram for Tellusim::CUShader:



Collaboration diagram for Tellusim::CUShader:



Public Member Functions

- CUmodule **getModule** () const
- CUfunction **getFunction** () const

Additional Inherited Members

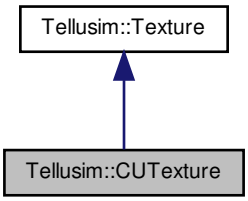
5.66.1 Detailed Description

The CUShader class extends the Shader class to specialize in managing shaders for CUDA. It provides methods to retrieve the underlying CUDA module and function, enabling integration with CUDA.

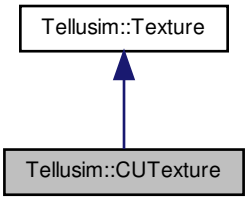
5.67 Tellusim::CUTexture Class Reference

`#include <platform/TellusimTexture.h>`

Inheritance diagram for Tellusim::CUTexture:



Collaboration diagram for Tellusim::CUTexture:



Public Member Functions

- CUmipmappedArray **getTextureArray** () const
- CUarray **getTextureLevel** (uint32_t index) const
- uint32_t **getArrayFormat** () const
- uint32_t **getArrayChannels** () const
- CUexternalMemory **getSharedMemory** () const

Additional Inherited Members

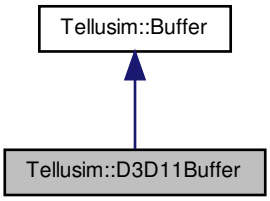
5.67.1 Detailed Description

The CUTexture class is a CUDA-specific implementation of the Texture class, providing access to CUDA texture resources and memory management. It allows retrieval of the texture array and texture levels, along with details about its array format and channels. Additionally, the class provides access to shared memory for efficient interoperation with CUDA, enabling advanced texture handling in GPU-based applications.

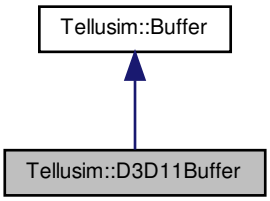
5.68 Tellusim::D3D11Buffer Class Reference

`#include <platform/TellusimBuffer.h>`

Inheritance diagram for Tellusim::D3D11Buffer:



Collaboration diagram for Tellusim::D3D11Buffer:



Public Member Functions

- bool create (Flags flags, ID3D11Buffer *buffer)
  create external buffer
- ID3D11Buffer * **getD3D11Buffer** () const
- ID3D11UnorderedAccessView * **getUnorderedAccessView** () const
- ID3D11ShaderResourceView * **getShaderResourceView** () const
- void * **getInteropHandle** () const

Additional Inherited Members

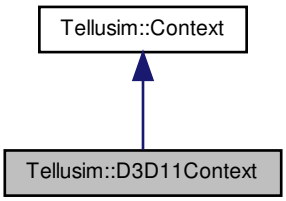
5.68.1 Detailed Description

The D3D11Buffer class is a Direct3D11-specific implementation of the Buffer class, providing access to internal resources and views. It enables the creation of external buffers from ID3D11Buffer objects and provides methods to retrieve unordered access and shader resource views. This class also includes support for interop handles to facilitate advanced resource management in Direct3D11-based applications.

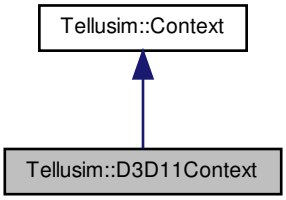
5.69 Tellusim::D3D11Context Class Reference

`#include <platform/TellusimContext.h>`

Inheritance diagram for Tellusim::D3D11Context:



Collaboration diagram for Tellusim::D3D11Context:



Public Member Functions

- bool create (ID3D11Device *device)
  create context
- IDXGIFactory * getFactory () const
  current device
- ID3D11Device * **getDevice** () const
- ID3D11DeviceContext * **getD3D11Context** () const

Static Public Member Functions

- static void * getProcAddress (const char *name)
  Direct3D11 functions.
- static bool error (uint32_t result)
  check Direct3D11 errors

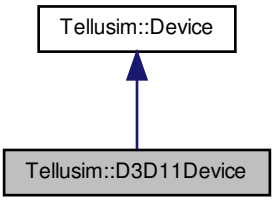
5.69.1 Detailed Description

The D3D11Context class is a Direct3D11-specific implementation of the Context class. It allows initializing the rendering context using an externally provided ID3D11Device. The class also provides access to the DXGI factory, device, and device context.

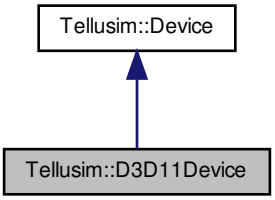
5.70 Tellusim::D3D11Device Class Reference

`#include <platform/TellusimDevice.h>`

Inheritance diagram for Tellusim::D3D11Device:



Collaboration diagram for Tellusim::D3D11Device:



Public Member Functions

- **D3D11Device** (Context &context)
- **D3D11Device** (Surface &surface)
- **D3D11Device** (Window &window)
- ID3D11Device * getD3D11Device () const

  command context
- ID3D11DeviceContext * **getCommand** () const

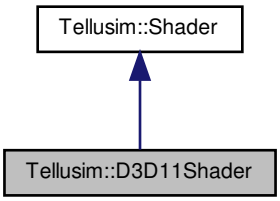
5.70.1 Detailed Description

The D3D11Device class extends the Device class to provide Direct3D11-specific functionality for managing a rendering device. It includes methods for obtaining access to the underlying Direct3D11 device and command context, enabling low-level rendering operations.

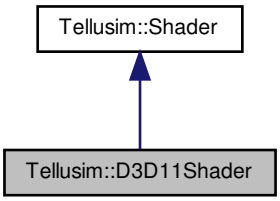
5.71 Tellusim::D3D11Shader Class Reference

`#include <platform/TellusimShader.h>`

Inheritance diagram for Tellusim::D3D11Shader:



Collaboration diagram for Tellusim::D3D11Shader:



Public Member Functions

- void * **getD3D11Shader** () const
- ID3DBlob * **getShaderBlob** () const

Additional Inherited Members

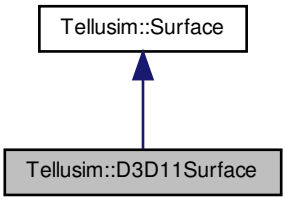
5.71.1 Detailed Description

The D3D11Shader class extends the Shader class to specialize in managing shaders for Direct3D11. It provides methods to retrieve the underlying Direct3D11 shader object and the shader blob, enabling integration with Direct3↩D11.

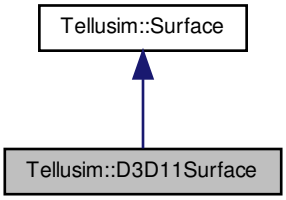
5.72 Tellusim::D3D11Surface Class Reference

`#include <platform/TellusimSurface.h>`

Inheritance diagram for Tellusim::D3D11Surface:



Collaboration diagram for Tellusim::D3D11Surface:



Public Member Functions

- **D3D11Surface** (D3D11Context &context)
- IDXGIFactory * getFactory () const

  current device
- ID3D11Device * **getDevice** () const
- ID3D11DeviceContext * **getContext** () const
- void setSwapChain (IDXGISwapChain *swap_chain)

  swap chain
- IDXGISwapChain * **getSwapChain** () const
- void setRenderTarget (ID3D11Texture2D *render_target)

  render targets
- void **setDepthStencil** (ID3D11Texture2D *depth_stencil)
- ID3D11Texture2D * **getRenderTarget** () const
- ID3D11Texture2D * **getDepthStencil** () const
- void setRenderTargetView (ID3D11RenderTargetView *render_target_view)

  render target views

- void **setDepthStencilView** (ID3D11DepthStencilView *depth_stencil_view)
- ID3D11RenderTargetView * **getRenderTargetView** () const
- ID3D11DepthStencilView * **getDepthStencilView** () const
- uint32_t getColorDXGIFormat () const

  surface formats
- uint32_t **getDepthDXGIFormat** () const

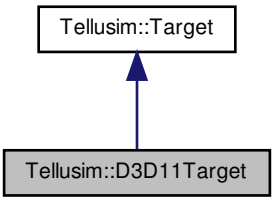
5.72.1 Detailed Description

The D3D11Surface class extends the Surface class to provide Direct3D11-specific functionality for managing a rendering surface. It includes methods for interacting with Direct3D11 devices, device contexts, and swap chains, enabling rendering operations in the context of Direct3D11. The class supports managing render targets, depth-stencil buffers, and render target views, which are essential for rendering operations.

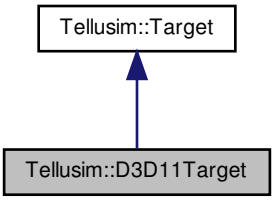
5.73 Tellusim::D3D11Target Class Reference

`#include <platform/TellusimTarget.h>`

Inheritance diagram for Tellusim::D3D11Target:



Collaboration diagram for Tellusim::D3D11Target:



Public Member Functions

- ID3D11RenderTargetView ∗∗ **getRenderTargetViews** () const
- ID3D11DepthStencilView ∗ **getDepthStencilView** () const

Additional Inherited Members

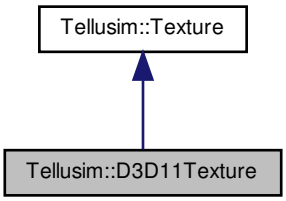
5.73.1 Detailed Description

The D3D11Target class is a Direct3D11-specific implementation of the Target class, providing methods for interacting with render and depth-stencil targets in a Direct3D11 context. It allows access to an array of render target views and a single depth-stencil view, which are essential for rendering operations in Direct3D11.

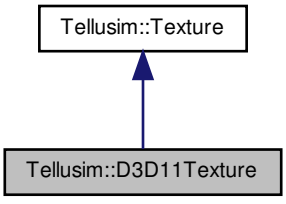
5.74 Tellusim::D3D11Texture Class Reference

```
#include <platform/TellusimTexture.h>
```

Inheritance diagram for Tellusim::D3D11Texture:



Collaboration diagram for Tellusim::D3D11Texture:



Public Member Functions

- bool create (Type type, ID3D11Texture2D *texture, Flags flags=DefaultFlags, Format format=Format↩ Unknown)

  create external texture
- uint32_t **getDXGIFormat** () const
- ID3D11Texture2D * **getD3D11Texture** () const
- ID3D11ShaderResourceView * **getShaderResourceView** () const
- ID3D11RenderTargetView * **getRenderTargetView** () const
- ID3D11DepthStencilView * **getDepthStencilView** () const
- ID3D11UnorderedAccessView * **getUnorderedAccessView** () const
- void * **getInteropHandle** () const

Additional Inherited Members

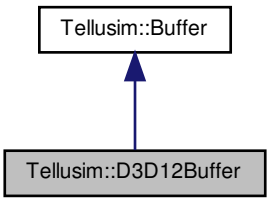
5.74.1 Detailed Description

The D3D11Texture class is a Direct3D11-specific implementation of the Texture class, providing access to internal resources and views. It supports the creation of external textures from ID3D11Texture2D objects and provides methods for accessing various views, such as shader resource, render target, depth stencil, and unordered access views. Additionally, the class includes interop handle support for integrating with other APIs or systems.

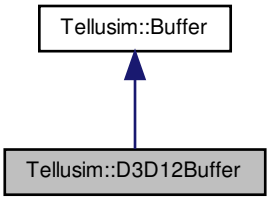
5.75 Tellusim::D3D12Buffer Class Reference

`#include <platform/TellusimBuffer.h>`

Inheritance diagram for Tellusim::D3D12Buffer:



Collaboration diagram for Tellusim::D3D12Buffer:



Public Member Functions

- bool create (Flags flags, ID3D12Resource *buffer, uint32_t state)
  create external buffer
- ID3D12Resource * **getD3D12Buffer** () const
- size_t **getUnorderedAccessView** () const
- size_t **getShaderResourceView** () const
- uint64_t **getBufferAddress** () const
- void **setBufferState** (uint32_t state)
- uint32_t **getBufferState** () const
- void * **getSharedHandle** () const
- void * **getInteropHandle** () const

Additional Inherited Members

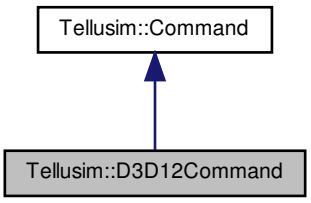
5.75.1 Detailed Description

The D3D12Buffer class is a Direct3D12-specific implementation of the Buffer class, providing access to internal resources and views. It allows the creation of external buffers from ID3D12Resource objects and provides methods to retrieve unordered access and shader resource views, along with the buffer memory address. This class also facilitates managing buffer state transitions and offers shared and interop handles for advanced resource handling.

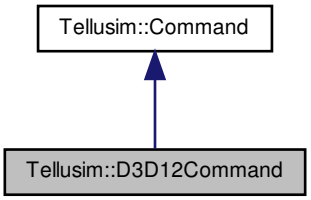
5.76 Tellusim::D3D12Command Class Reference

`#include <platform/TellusimCommand.h>`

Inheritance diagram for Tellusim::D3D12Command:



Collaboration diagram for Tellusim::D3D12Command:



Public Member Functions

- ID3D12GraphicsCommandList * getD3D12Command () const

  command context
- void update ()

  update resources

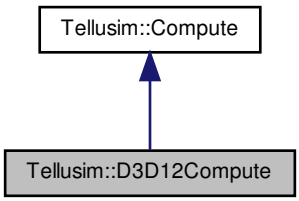
5.76.1 Detailed Description

The D3D12Command class is a Direct3D12-specific implementation of the Command class, providing access to the command list.

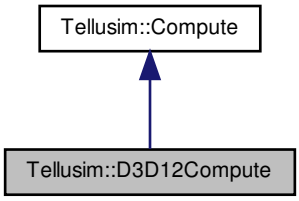
5.77 Tellusim::D3D12Compute Class Reference

```
#include <platform/TellusimCompute.h>
```

Inheritance diagram for Tellusim::D3D12Compute:



Collaboration diagram for Tellusim::D3D12Compute:



Public Member Functions

- ID3D12GraphicsCommandList * getCommand () const

  command context
- void update ()

  update resources

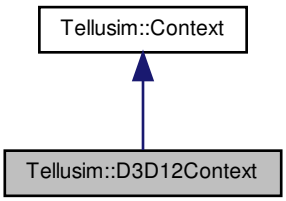
5.77.1 Detailed Description

The D3D12Compute class is a Direct3D12-specific implementation of the Compute class, providing access to the command list.

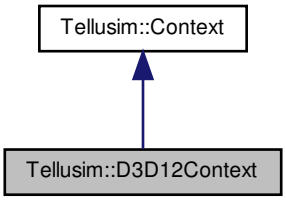
5.78 Tellusim::D3D12Context Class Reference

```
#include <platform/TellusimContext.h>
```

Inheritance diagram for Tellusim::D3D12Context:



Collaboration diagram for Tellusim::D3D12Context:



Public Member Functions

- bool create (ID3D12Device *device, ID3D12CommandQueue *queue)

  create context
- IDXGIFactory4 * getFactory () const

  current device
- ID3D12Device * **getDevice** () const
- ID3D12CommandQueue * **getQueue** () const
- ID3D12GraphicsCommandList * **getCommand** () const

Static Public Member Functions

- static void * getProcAddress (const char *name)

  Direct3D12 functions.
- static bool error (uint32_t result)

  check Direct3D12 errors

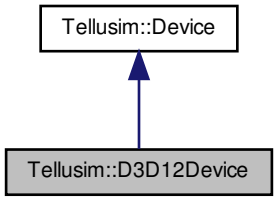
5.78.1 Detailed Description

The D3D12Context class is a Direct3D12-specific implementation of the Context class. It allows initialization of the rendering context using an externally provided ID3D12Device and command queue. The class also provides access to the underlying DXGI factory, device, command queue, and command list.

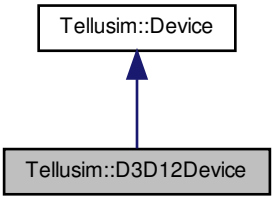
5.79 Tellusim::D3D12Device Class Reference

```
#include <platform/TellusimDevice.h>
```

Inheritance diagram for Tellusim::D3D12Device:



Collaboration diagram for Tellusim::D3D12Device:



Public Member Functions

- **D3D12Device** (Context &context)
- **D3D12Device** (Surface &surface)
- **D3D12Device** (Window &window)
- void setBufferState (Buffer &buffer, uint32_t state)

  buffer state
- void setTextureState (Texture &texture, uint32_t state)

  texture state
- ID3D12Device * getD3D12Device () const

  command context
- ID3D12CommandQueue * **getQueue** () const
- ID3D12GraphicsCommandList * **getCommand** () const

5.79.1 Detailed Description

The D3D12Device class extends the Device class to provide Direct3D12-specific functionality for managing a rendering device. It includes methods for interacting with Direct3D12 buffers and textures, such as setting their state, and provides access to key Direct3D12 components like the device, command queue, and graphics command list.

5.80 Tellusim::D3D12Tracing::D3D12Instance Struct Reference

tracing instance

`#include <platform/TellusimTracing.h>`

Public Attributes

- float32_t **transform** [12]
- uint32_t **data**: 24
- uint32_t **mask**: 8
- uint32_t **offset**: 24
- uint32_t **flags**: 8
- uint64_t **address**

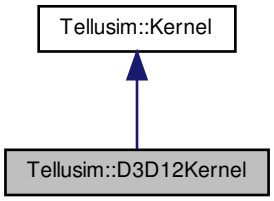
5.80.1 Detailed Description

tracing instance

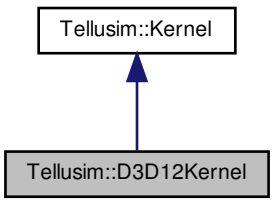
5.81 Tellusim::D3D12Kernel Class Reference

`#include <platform/TellusimKernel.h>`

Inheritance diagram for Tellusim::D3D12Kernel:



Collaboration diagram for Tellusim::D3D12Kernel:



Public Member Functions

- ID3D12RootSignature * **getRootSignature** () const

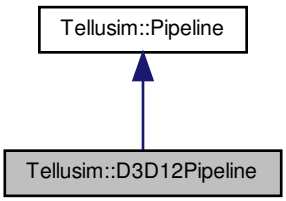
5.81.1 Detailed Description

The D3D12Kernel class is a specialized compute kernel for Direct3D12, inheriting from the Kernel class. It provides a method to retrieve the root signature, which is a critical component for binding resources and defining pipeline states in Direct3D12.

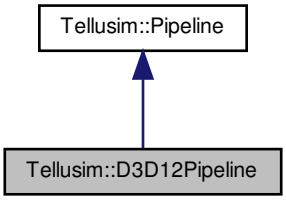
5.82 Tellusim::D3D12Pipeline Class Reference

`#include <platform/TellusimPipeline.h>`

Inheritance diagram for Tellusim::D3D12Pipeline:



Collaboration diagram for Tellusim::D3D12Pipeline:



Public Member Functions

- ID3D12RootSignature * **getRootSignature** () const

Additional Inherited Members

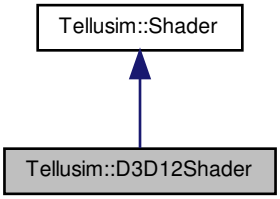
5.82.1 Detailed Description

The D3D12Pipeline class is a specialized graphics pipeline for Direct3D12, derived from the Pipeline class. It provides access to the native Direct3D12 root signature, enabling precise control over resource bindings and pipeline state in Direct3D12 rendering workflows.

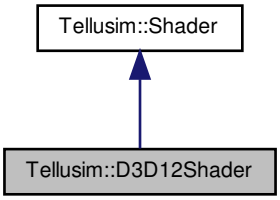
5.83 Tellusim::D3D12Shader Class Reference

`#include <platform/TellusimShader.h>`

Inheritance diagram for Tellusim::D3D12Shader:



Collaboration diagram for Tellusim::D3D12Shader:



Public Member Functions

- ID3DBlob * **getShaderBlob** () const

Additional Inherited Members

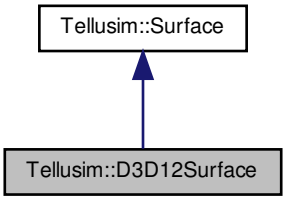
5.83.1 Detailed Description

The D3D12Shader class extends the Shader class to specialize in managing shaders for Direct3D12. It provides a method to retrieve the shader blob, which contains the compiled binary data of the shader, enabling integration with Direct3D12.

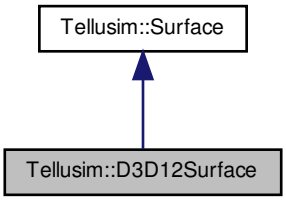
5.84 Tellusim::D3D12Surface Class Reference

`#include <platform/TellusimSurface.h>`

Inheritance diagram for Tellusim::D3D12Surface:



Collaboration diagram for Tellusim::D3D12Surface:



Public Member Functions

- **D3D12Surface** (D3D12Context &context)
- IDXGIFactory4 * getFactory () const

  current device
- ID3D12Device * **getDevice** () const
- ID3D12CommandQueue * **getQueue** () const
- ID3D12GraphicsCommandList * **getCommand** () const
- void setSwapChain (IDXGISwapChain *swap_chain)

  swap chain
- IDXGISwapChain * **getSwapChain** () const
- void setRenderTarget (ID3D12Resource *render_target)

  render targets
- void **setDepthStencil** (ID3D12Resource *depth_stencil)
- ID3D12Resource * **getRenderTarget** () const
- ID3D12Resource * **getDepthStencil** () const
- void setRenderTargetView (size_t render_target_view)

render target views

- void **setDepthStencilView** (size_t depth_stencil_view)
- size_t **getRenderTargetView** () const
- size_t **getDepthStencilView** () const
- uint32_t getColorDXGIFormat () const

surface formats

- uint32_t **getDepthDXGIFormat** () const

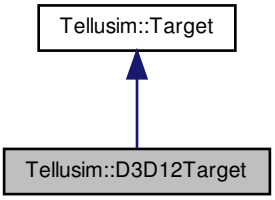
5.84.1 Detailed Description

The D3D12Surface class extends the Surface class to provide Direct3D12-specific functionality for managing a rendering surface. It includes methods for interacting with Direct3D12 devices, command queues, and command lists, enabling rendering operations in the context of Direct3D12. The class supports managing swap chains, render targets, and depth-stencil buffers, which are essential for rendering operations.

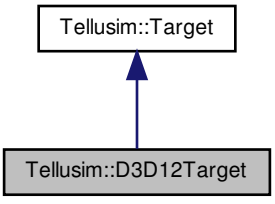
5.85 Tellusim::D3D12Target Class Reference

```
#include <platform/TellusimTarget.h>
```

Inheritance diagram for Tellusim::D3D12Target:



Collaboration diagram for Tellusim::D3D12Target:



Public Member Functions

- size_t * **getRenderTargetViews** () const
- size_t **getDepthStencilView** () const

Additional Inherited Members

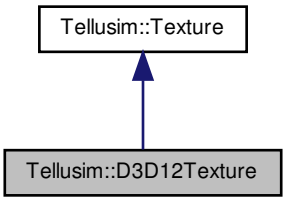
5.85.1 Detailed Description

The D3D12Target class is a Direct3D12-specific implementation of the Target class, providing methods for interacting with render and depth-stencil targets in a Direct3D12 context. It allows access to an array of render target views and a single depth-stencil view, which are essential for rendering operations in Direct3D12.

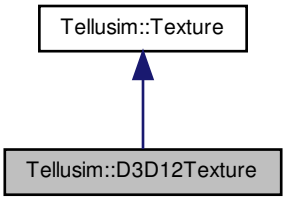
5.86 Tellusim::D3D12Texture Class Reference

`#include <platform/TellusimTexture.h>`

Inheritance diagram for Tellusim::D3D12Texture:



Collaboration diagram for Tellusim::D3D12Texture:



Public Member Functions

- bool create (Type type, ID3D12Resource *texture, uint32_t state, Flags flags=DefaultFlags, Format format=FormatUnknown)

  create external texture
- uint32_t **getDXGIFormat** () const
- ID3D12Resource * **getD3D12Texture** () const
- size_t **getShaderResourceView** () const
- size_t **getRenderTargetView** () const
- size_t **getDepthStencilView** () const
- size_t **getUnorderedAccessView** () const
- void **setTextureState** (uint32_t state)
- uint32_t **getTextureState** () const
- void * **getSharedHandle** () const
- void * **getInteropHandle** () const

Additional Inherited Members

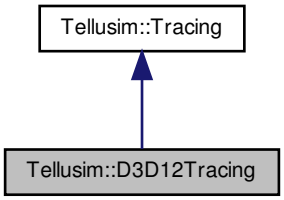
5.86.1 Detailed Description

The D3D12Texture class is a Direct3D12-specific implementation of the Texture class, providing access to internal resources and views. It supports creation from external ID3D12Resource objects and exposes methods to retrieve shader, render target, depth stencil, and unordered access views. This class also manages resource state transitions and provides access to shared and interop handles for advanced usage scenarios.

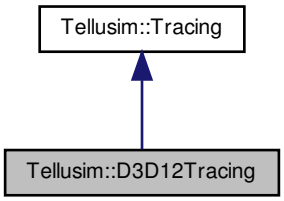
5.87 Tellusim::D3D12Tracing Class Reference

```
#include <platform/TellusimTracing.h>
```

Inheritance diagram for Tellusim::D3D12Tracing:



Collaboration diagram for Tellusim::D3D12Tracing:



Classes

- struct D3D12Instance
  tracing instance

Public Member Functions

- void * **getGeometryDesc** (uint32_t index) const
- void * **getBuildInputs** () const
- void * **getPrebuildInfo** () const
- void * **getBuildDesc** () const
- Buffer **getTracingBuffer** () const
- size_t **getShaderResourceView** () const

Additional Inherited Members

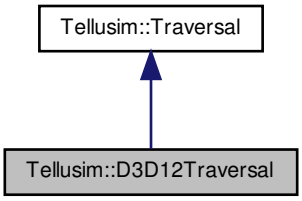
5.87.1 Detailed Description

The D3D12Tracing class is a Direct3D12-specific implementation of the Tracing class. It provides methods and structures for managing ray-tracing acceleration structures within the Direct3D12 API.

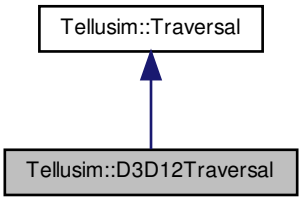
5.88 Tellusim::D3D12Traversal Class Reference

`#include <platform/TellusimTraversal.h>`

Inheritance diagram for Tellusim::D3D12Traversal:



Collaboration diagram for Tellusim::D3D12Traversal:



Public Member Functions

- ID3D12RootSignature * **getRootSignature** () const

5.88.1 Detailed Description

The D3D12Traversal class is a specialized ray-tracing pipeline for Direct3D12, derived from the Traversal class. It provides access to the native Direct3D12 root signature, enabling precise control over resource bindings and pipeline state in Direct3D12 ray-tracing workflows.

5.89 Tellusim::Date Class Reference

`#include <core/TellusimTime.h>`

Public Member Functions

- **Date** (int64_t time, bool local=true)
- **Date** (const char *str, const char *format=nullptr)
- void clear ()

  clear date
- void setTime (int64_t time, bool local=true)

  time in seconds since 1970-01-01 00:00:00
- int64_t **getTime** (bool local=true) const
- bool setString (const char *str, const char *format=nullptr)
- String **getString** (const char *format=nullptr) const
- void setYear (uint32_t year)

  current date
- void **setMonth** (uint32_t month)
- void **setDate** (uint32_t date)
- void **setDay** (uint32_t day)
- void **setHours** (uint32_t hours)
- void **setMinutes** (uint32_t minutes)
- void **setSeconds** (uint32_t seconds)
- uint32_t **getYear** () const
- uint32_t **getMonth** () const
- uint32_t **getDate** () const
- uint32_t **getDay** () const
- uint32_t **getHours** () const
- uint32_t **getMinutes** () const
- uint32_t **getSeconds** () const

Static Public Member Functions

- static int32_t getTimeZone ()

  local timezone in seconds

5.89.1 Detailed Description

The Date class provides a flexible way to handle date and time operations. It allows for the creation of date objects using a specific timestamp or a formatted string, with options for local or UTC time. The class offers methods to set and retrieve individual date and time components, such as year, month, day, hours, minutes, and seconds. Additionally, it supports converting time to and from a string representation, with customizable date formats. This class enables comprehensive date and time management with various formatting and conversion capabilities.

5.89.2 Member Function Documentation

5.89.2.1 setString()

```
bool Tellusim::Date::setString (
            const char * str,
            const char * format = nullptr )
```

string time value

Parameters

format	Default time format is yyyy-MM-dd HH:mm:ss

5.90 Tellusim::DecoderJPEG Class Reference

```
#include <graphics/TellusimDecoderJPEG.h>
```

Public Types

- enum Mode {
  ModeR = 0,
  ModeRG,
  ModeRGBA,
  ModeYUV444,
  ModeYUV422H,
  ModeYUV422V,
  ModeYUV420,
  NumModes }

  Decoder modes.

- enum Flags {
  FlagNone = 0,
  FlagR = (1 << ModeR),
  FlagRG = (1 << ModeRG),
  FlagRGBA = (1 << ModeRGBA),
  FlagYUV444 = (1 << ModeYUV444),
  FlagYUV422H = (1 << ModeYUV422H),
  FlagYUV422V = (1 << ModeYUV422V),
  FlagYUV420 = (1 << ModeYUV420),
  FlagsAll = (FlagR | FlagRG | FlagRGBA | FlagYUV444 | FlagYUV422H | FlagYUV422V | FlagYUV420) }

  Decoder flags.

Public Member Functions

- void clear ()

  clear decoder
- bool isCreated (Mode mode) const

  check decoder
- bool create (const Device &device, Mode mode)

  create decoder
- bool **create** (const Device &device, Flags flags)
- Texture loadTexture (const Device &device, const char *name, Texture::Flags flags=Texture::DefaultFlags) const
- Texture **loadTexture** (const Device &device, Stream &stream, Texture::Flags flags=Texture::DefaultFlags) const
- bool dispatch (Compute &compute, Mode mode, Texture &dest, Texture &src, const Slice &dest_slice, const Slice &src_slice) const
- bool **dispatch** (Compute &compute, Mode mode, Texture &dest, Texture &src, const Slice &src_slice) const
- bool **dispatch** (Compute &compute, Mode mode, Texture &dest, Texture &src) const
- bool dispatchYUV (Compute &compute, Mode mode, Texture &dest, Texture &src, const Slice &dest_slice, const Slice &src_slice) const
- bool **dispatchYUV** (Compute &compute, Mode mode, Texture &dest, Texture &src, const Slice &src_slice) const
- bool **dispatchYUV** (Compute &compute, Mode mode, Texture &dest, Texture &src) const

Static Public Member Functions

- static bool isYUV (Mode mode)

  YUV444 mode performs inplace YUVtoRGB conversion.
- static bool load (const char *name, Image &image, Mode &mode, Size &size)
- static bool **load** (Stream &stream, Image &image, Mode &mode, Size &size)

5.90.1 Detailed Description

The DecoderJPEG class provides a GPU-accelerated JPEG decoding interface capable of converting JPEG images into textures.

5.90.2 Member Function Documentation

5.90.2.1 load()

```
static bool Tellusim::DecoderJPEG::load (
            const char * name,
            Image & image,
            Mode & mode,
            Size & size )  [static]
```

load decoder image

Parameters

name	Image name
image	Decoder image.
mode	Decoding mode.
size	Decoding size.

5.90.2.2 loadTexture()

```
Texture Tellusim::DecoderJPEG::loadTexture (
            const Device & device,
            const char * name,
            Texture::Flags flags = Texture::DefaultFlags ) const
```

load texture from image

Parameters

device	Device pointer.
name	Image name.
flags	Texture flags.

5.90.2.3 dispatch()

```
bool Tellusim::DecoderJPEG::dispatch (
            Compute & compute,
            Mode mode,
            Texture & dest,
            Texture & src,
            const Slice & dest_slice,
            const Slice & src_slice ) const
```

dispatch decoder

Parameters

mode	Decoder mode.
dest	Destination surface.
src	Source FFT surface.
dest_slice	Destination texture slice.
src_slice	Source texture slice.

5.90.2.4 dispatchYUV()

```
bool Tellusim::DecoderJPEG::dispatchYUV (
            Compute & compute,
            Mode mode,
            Texture & dest,
            Texture & src,
            const Slice & dest_slice,
            const Slice & src_slice ) const
```

dispatch YUV converter

Parameters

mode	Decoder mode.
dest	Destination surface.
src	Source YUV surface.
dest_slice	Destination texture slice.
src_slice	Source texture slice.

5.91 Tellusim::DefaultDestructor< Type > Struct Template Reference

```
#include <core/TellusimPointer.h>
```

Static Public Member Functions

- static void destructor (Type *ptr)

  delete pointer

5.91.1 Detailed Description

template<class Type>
struct Tellusim::DefaultDestructor< Type >

Default destructor

5.92 Tellusim::Desktop Class Reference

```
#include <system/TellusimDesktop.h>
```

Public Member Functions

- bool update ()

  update configuration
- uint32_t getWidth () const

  desktop resolution
- uint32_t **getHeight** () const
- int32_t **getPositionX** () const
- int32_t **getPositionY** () const
- float32_t **getScale** () const
- uint32_t getNumScreens () const

  screen configuration
- String **getScreenName** (uint32_t index) const
- String **getScreenDevice** (uint32_t index) const
- uint32_t **getScreenWidth** (uint32_t index) const
- uint32_t **getScreenHeight** (uint32_t index) const
- int32_t **getScreenPositionX** (uint32_t index) const
- int32_t **getScreenPositionY** (uint32_t index) const
- uint32_t **getScreenFrequency** (uint32_t index) const
- uint32_t getNumModes (uint32_t index) const

  screen video modes
- uint32_t **getModeWidth** (uint32_t index, uint32_t mode) const
- uint32_t **getModeHeight** (uint32_t index, uint32_t mode) const
- uint32_t getModeIndex (uint32_t index, uint32_t width, uint32_t height) const

  change screen resolution
- bool **setMode** (uint32_t index, uint32_t width, uint32_t height)
- bool **restoreMode** (uint32_t index)
- uint32_t getWidth (uint32_t index) const

  current screen configuration
- uint32_t **getHeight** (uint32_t index) const
- int32_t **getPositionX** (uint32_t index) const
- int32_t **getPositionY** (uint32_t index) const
- uint32_t getScreenIndex (int32_t x, int32_t y) const

  get current screen index
- bool setMouse (int32_t x, int32_t y) const

  mouse position
- bool **getMouse** (int32_t &x, int32_t &y) const

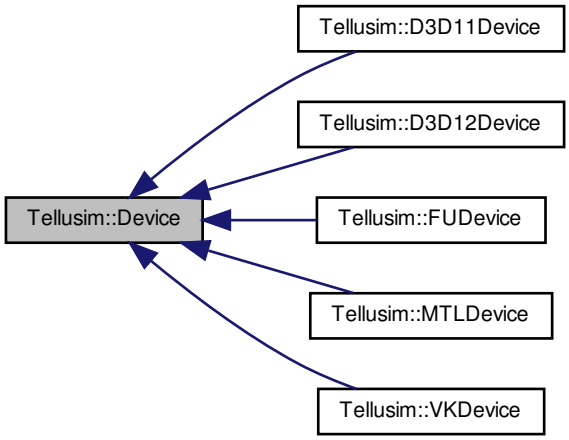
5.92.1 Detailed Description

The Desktop class provides methods to interact with the desktop environment and screen configurations on Windows, Linux, and macOS. It allows for adjusting screen resolution and position, querying screen details, and managing multiple screens and their settings. Additionally, it provides functionality for setting and retrieving the mouse position.

5.93 Tellusim::Device Class Reference

`#include <platform/TellusimDevice.h>`

Inheritance diagram for Tellusim::Device:



Classes

- struct Features
  device features

Public Member Functions

- **Device** (Context &context)
- **Device** (Surface &surface)
- **Device** (Window &window)
- Platform getPlatform () const
  device platform
- const char * **getPlatformName** () const
- uint32_t getIndex () const
  device index
- String getName () const
  device info
- String **getVendor** () const
- String **getVersion** () const
- const Features & **getFeatures** () const
- bool hasQuery (Query::Type type) const
  device types
- bool **hasShader** (Shader::Type type) const
- bool hasTarget (Format format) const
  device formats

- bool **hasTexture** (Format format) const
- bool **hasSurface** (Format format) const
- Device createDevice (uint32_t index, uint32_t frames=3) const

  create device
- Device **createCommandDevice** (uint32_t frames=3) const
- Device **createComputeDevice** (uint32_t frames=3) const
- Device **createCopyDevice** (uint32_t frames=3) const
- Query createQuery () const

  create query
- Query **createQuery** (Query::Type type) const
- Fence createFence () const

  create fence
- Fence **createFence** (Fence &shared) const
- Fence **createFence** (Fence::Flags flags) const
- Buffer createBuffer () const

  create buffer
- Buffer **createBuffer** (Buffer &shared) const
- Buffer **createBuffer** (Buffer::Flags flags, size_t size, Format format=FormatUnknown) const
- Buffer **createBuffer** (Buffer::Flags flags, const void *src, size_t size, Format format=FormatUnknown) const
- Sampler createSampler () const

  create sampler
- Sampler **createSampler** (const Sampler &sampler) const
- Sampler **createSampler** (Sampler::Filter filter, Sampler::WrapMode mode=Sampler::WrapModeRepeat, uint32_t anisotropy=Sampler::MaxAnisotropy) const
- Texture createTexture () const

  create texture
- Texture **createTexture** (Texture &shared) const
- Texture **createTexture** (Texture::Type type, Format format, const Size &size, uint32_t layers, Texture::Flags flags=Texture::DefaultFlags) const
- Texture **createTexture** (Texture::Type type, Format format, const Size &size, Texture::Flags flags=Texture::DefaultFlags) const
- Texture **createTexture** (const Image &image, Texture::Flags flags=Texture::DefaultFlags, Async *async=nullptr) const
- Texture **createTexture2D** (Format format, uint32_t size, Texture::Flags flags=Texture::DefaultFlags) const
- Texture **createTexture3D** (Format format, uint32_t size, Texture::Flags flags=Texture::DefaultFlags) const
- Texture **createTextureCube** (Format format, uint32_t size, Texture::Flags flags=Texture::DefaultFlags) const
- Texture **createTexture2D** (Format format, uint32_t width, uint32_t height, Texture::Flags flags=Texture::DefaultFlags) const
- Texture **createTexture3D** (Format format, uint32_t width, uint32_t height, uint32_t depth, Texture::Flags flags=Texture::DefaultFlags) const
- Texture **createTexture2D** (Format format, uint32_t width, uint32_t height, uint32_t layers, Texture::Flags flags=Texture::DefaultFlags) const
- Texture **createTextureCube** (Format format, uint32_t size, uint32_t layers, Texture::Flags flags=Texture::DefaultFlags) const
- Texture **loadTexture** (const char *name, Texture::Flags flags=Texture::DefaultFlags, Image::Flags image_flags=Image::FlagNone, uint32_t offset=0, Async *async=nullptr) const
- Texture **loadTexture** (const String &name, Texture::Flags flags=Texture::DefaultFlags, Image::Flags image_flags=Image::FlagNone, uint32_t offset=0, Async *async=nullptr) const
- Texture **loadTexture** (Stream &stream, Texture::Flags flags=Texture::DefaultFlags, Image::Flags image_flags=Image::FlagNone, uint32_t offset=0, Async *async=nullptr) const
- Tracing createTracing () const

  create tracing
- Tracing **createTracing** (const Tracing &tracing) const

- Tracing **createTracing** (uint32_t num_instances, Buffer instance_buffer=Buffer::null, size_t instance_offset=0, Tracing::Flags flags=Tracing::DefaultFlags) const
- Tracing **createTracing** (uint32_t num_vertices, Format vertex_format, size_t vertex_stride, uint32_t num_indices, Format index_format, Tracing::Flags flags=Tracing::DefaultFlags) const
- Tracing **createTracing** (uint32_t num_bounds, size_t bound_stride, Buffer bound_buffer=Buffer::null, size_t bound_offset=0, Tracing::Flags flags=Tracing::DefaultFlags) const
- BufferTable createBufferTable () const

  create buffer table
- BufferTable **createBufferTable** (uint32_t size) const
- BufferTable **createBufferTable** (const Array< Buffer > &buffers, bool owner=false) const
- TextureTable createTextureTable () const

  create texture table
- TextureTable **createTextureTable** (Texture::Type type, uint32_t size) const
- TextureTable **createTextureTable** (const Array< Texture > &textures, bool owner=false) const
- Shader createShader () const

  create shader
- Shader **loadShader** (Shader::Type type, const char *name, const char *format,...) const 1(4
- Shader Shader **loadShaderGLSL** (Shader::Type type, const char *name, const char *format,...) const 1(4
- Shader Shader Shader **loadShader** (Shader::Type type, const char *name, const String &macros=String::null) const
- Shader **loadShaderGLSL** (Shader::Type type, const char *name, const String &macros=String::null) const
- Shader **loadShaderSPIRV** (Shader::Type type, const char *name) const
- Shader **createShader** (Shader::Type type, const char *src, const char *format,...) const 1(4
- Shader Shader **createShaderGLSL** (Shader::Type type, const char *src, const char *format,...) const 1(4
- Shader Shader Shader **createShader** (Shader::Type type, const char *src, const String &macros=String::null) const
- Shader **createShaderGLSL** (Shader::Type type, const char *src, const String &macros=String::null) const
- Shader **createShaderSPIRV** (Shader::Type type, const Array< uint32_t > &data) const
- Kernel createKernel () const

  create kernel
- Kernel **createKernel** (const Kernel &kernel) const
- void releaseKernel (Kernel &kernel) const

  release kernel
- Pipeline createPipeline () const

  create pipeline
- Pipeline **createPipeline** (const Pipeline &pipeline) const
- void releasePipeline (Pipeline &pipeline) const

  release pipeline
- Traversal createTraversal () const

  create traversal
- Traversal **createTraversal** (const Traversal &traversal) const
- void releaseTraversal (Traversal &traversal) const

  release traversal
- Target createTarget () const

  create target
- Target **createTarget** (Surface &surface) const
- Target **createTarget** (Window &window) const
- Target **createTarget** (const InitializerList< Texture > &textures, Target::Operation op=Target::OpDefault) const
- Compute createCompute () const

  create compute
- Command createCommand () const

  create command

- Command **createCommand** (Target &target) const
- bool setBuffer (Buffer &buffer, size_t offset, const void *src, size_t size) const

  set buffer data
- bool **setBuffer** (Buffer &buffer, const void *src, size_t size) const
- bool **setBuffer** (Buffer &buffer, const void *src) const
- bool getBuffer (Buffer &buffer, size_t offset, void *dest, size_t size) const

  get buffer data
- bool **getBuffer** (Buffer &buffer, void *dest, size_t size) const
- bool **getBuffer** (Buffer &buffer, void *dest) const
- void * mapBuffer (Buffer &buffer, size_t offset, size_t size) const

  map buffer data
- void * **mapBuffer** (Buffer &buffer, size_t size) const
- void * **mapBuffer** (Buffer &buffer) const
- bool **unmapBuffer** (Buffer &buffer) const
- bool copyBuffer (Buffer &buffer, size_t dest_offset, Buffer &src, size_t src_offset, size_t size) const

  copy buffer data
- bool **copyBuffer** (Buffer &buffer, size_t dest_offset, Buffer &src, size_t size) const
- bool **copyBuffer** (Buffer &buffer, Buffer &src, size_t size) const
- bool **copyBuffer** (Buffer &buffer, Buffer &src) const
- bool clearBuffer (Buffer &buffer, Format format, size_t offset, const void *src, size_t size) const

  clear buffer data
- bool **clearBuffer** (Buffer &buffer, Format format, const void *src, size_t size) const
- bool **clearBuffer** (Buffer &buffer, Format format, const void *src) const
- bool **clearBuffer** (Buffer &buffer) const
- bool bindBuffer (Buffer &buffer, const Array< size_t > &offsets, const Array< size_t > &sizes, bool commit, Fence &fence) const

  bind buffer memory
- bool **bindBuffer** (Buffer &buffer, const Array< size_t > &offsets, const Array< size_t > &sizes, bool commit) const
- bool **bindBuffer** (Buffer &buffer, size_t offset, size_t size, bool commit, Fence &fence) const
- bool **bindBuffer** (Buffer &buffer, size_t offset, size_t size, bool commit) const
- bool flushBuffer (Buffer &buffer, Buffer::Flags flags=Buffer::FlagNone) const

  flush buffer data
- bool **flushBuffers** (const Array< Buffer > &buffers, Buffer::Flags flags=Buffer::FlagNone) const
- bool **flushBuffers** (const InitializerList< Buffer > &buffers, Buffer::Flags flags=Buffer::FlagNone) const
- void releaseBuffer (Buffer &buffer) const

  release buffer
- void releaseSampler (Sampler &sampler) const

  release sampler
- bool setTexture (Texture &texture, const Origin &dest_origin, const Slice &dest_slice, const Image &image, const Slice &src_slice) const

  set texture data
- bool **setTexture** (Texture &texture, const Origin &dest_origin, const Image &image) const
- bool **setTexture** (Texture &texture, const Slice &dest_slice, const Image &image) const
- bool **setTexture** (Texture &texture, const Image &image) const
- bool getTexture (Texture &texture, const Slice &src_slice, Image &image, const Slice &dest_slice) const

  get texture data
- bool **getTexture** (Texture &texture, Image &image, const Slice &dest_slice) const
- bool **getTexture** (Texture &texture, Image &image) const
- bool copyTexture (Texture &texture, const Origin &dest_origin, const Slice &dest_slice, Texture &src, const Region &src_region, const Slice &src_slice) const

  copy texture data

- bool **copyTexture** (Texture &texture, const Origin &dest_origin, Texture &src, const Region &src_region) const
- bool **copyTexture** (Texture &texture, const Slice &dest_slice, Texture &src, const Slice &src_slice) const
- bool **copyTexture** (Texture &texture, Texture &src) const
- bool clearTexture (Texture &texture, const Region &region, const Slice &slice, const void *src) const

  clear texture data
- bool **clearTexture** (Texture &texture, const Region &region, const void *src) const
- bool **clearTexture** (Texture &texture, const Slice &slice, const void *src) const
- bool **clearTexture** (Texture &texture, const void *src) const
- bool bindTexture (Texture &texture, const Region *regions, uint32_t num_regions, const Slice *slices, uint32_t num_slices, bool commit, Fence &fence) const

  bind texture memory
- bool **bindTexture** (Texture &texture, const Region *regions, uint32_t num_regions, const Slice *slices, uint32_t num_slices, bool commit) const
- bool **bindTexture** (Texture &texture, const Region &region, const Slice &slice, bool commit, Fence &fence) const
- bool **bindTexture** (Texture &texture, const Region &region, const Slice &slice, bool commit) const
- bool createMipmaps (Texture &texture, const Slice &slice) const

  create texture mipmaps
- bool **createMipmaps** (Texture &texture) const
- bool flushTexture (Texture &texture, Texture::Flags flags=Texture::FlagNone) const

  flush texture data
- bool **flushTexture** (Texture &texture, const Slice &slice, Texture::Flags flags=Texture::FlagNone) const
- bool **flushTextures** (const Array< Texture > &textures, Texture::Flags flags=Texture::FlagNone) const
- bool **flushTextures** (const InitializerList< Texture > &textures, Texture::Flags flags=Texture::FlagNone) const
- void releaseTexture (Texture &texture) const

  release texture
- virtual bool setTracing (Tracing &tracing, const Tracing::Instance *instances, uint32_t num_instances) const

  set tracing data
- bool buildTracing (Tracing &tracing, Buffer &buffer, Tracing::Flags flags=Tracing::FlagNone) const

  build tracing
- bool **buildTracing** (Tracing &tracing, Buffer &buffer, size_t offset, Tracing::Flags flags=Tracing::FlagNone) const
- bool **buildTracings** (const Array< Tracing > &tracings, Buffer &buffer, Tracing::Flags flags=Tracing::Flag↩None) const
- bool **buildTracings** (const Array< Tracing > &tracings, Buffer &buffer, size_t offset, Tracing::Flags flags=Tracing::FlagNone) const
- bool copyTracing (Tracing &tracing, Buffer &buffer, size_t offset=0) const

  copy tracing address
- bool **copyTracings** (const Array< Tracing > &tracings, Buffer &buffer, size_t offset, size_t stride=0) const
- bool flushTracing (Tracing &tracing) const

  flush tracing
- bool **flushTracings** (const Array< Tracing > &tracings) const
- void releaseTracing (Tracing &tracing) const

  release tracing
- bool setBufferTable (BufferTable &table, uint32_t index, Buffer &buffer, bool owner=false) const

  set table buffer
- bool **setBufferTable** (BufferTable &table, uint32_t index, const Array< Buffer > &buffers, bool owner=false) const
- void releaseBufferTable (BufferTable &table) const

  release buffer table
- bool setTextureTable (TextureTable &table, uint32_t index, Texture &texture, bool owner=false) const

set table texture

- bool **setTextureTable** (TextureTable &table, uint32_t index, const Array< Texture > &textures, bool owner=false) const
- void releaseTextureTable (TextureTable &table) const

  release texture table
- bool beginQuery (Query &query) const

  begin/end query
- void **endQuery** (Query &query) const
- bool copyQuery (Query &query, Buffer &buffer, size_t offset=0) const

  copy query data
- bool **copyQueries** (const Array< Query > &queries, Buffer &buffer, size_t offset=0, size_t stride=0) const
- bool **copyQueries** (const InitializerList< Query > &queries, Buffer &buffer, size_t offset=0, size_t stride=0) const
- bool waitFence (Fence &fence) const

  fence synchronization
- bool **signalFence** (Fence &fence) const
- bool execute (Device &device) const

  execute context
- bool flip (Fence &fence) const

  flip context
- bool **flip** () const
- bool flush () const

  flush context
- bool finish () const

  finish context
- bool check () const

  check errors

5.93.1 Detailed Description

The Device class represents a GPU device abstraction and serves as the primary interface for creating and managing GPU resources, including buffers, textures, shaders, pipelines, and synchronization primitives. It supports querying device properties, capabilities, and features through the Features struct and provides functions for resource creation, data transfer, and memory operations. Devices can be initialized from a Context, Surface, or Window, and support multiple device types such as Compute, Command, or Copy. This class enables resource management and rendering control across a wide range of hardware platforms, leveraging various GPU features and extensions.

5.94 Tellusim::DialogColor Class Reference

```
#include <interface/TellusimDialogs.h>
```

Public Types

- enum Flags {
  FlagNone = 0,
  FlagAlpha = (1 << 0),
  FlagMouse = (1 << 1),
  DefaultFlags = FlagNone,
  NumFlags = 2 }
  dialog flags
- enum Result {
  ResultCancel = 0,
  ResultOk,
  NumResults }
  dialog result
- using ChangedCallback = Function< void(Color)>
  changed callback
- using UpdateCallback = Function< bool()>
  update callback

Public Member Functions

- **DialogColor** (const char *title=nullptr, const Color &color=Color::zero)
- **DialogColor** (const String &title, const Color &color=Color::zero)
- void setPosition (int32_t x, int32_t y)
  dialog position
- int32_t **getPositionX** () const
- int32_t **getPositionY** () const
- void setTitle (const char *title)
  dialog title
- void **setTitle** (const String &title)
- String **getTitle** () const
- void setColor (const Color &color, bool callback=false)
  dialog color
- const Color & **getColor** () const
- void **setChangedCallback** (const ChangedCallback &func)
- ChangedCallback **getChangedCallback** () const
- void **setUpdateCallback** (const UpdateCallback &func)
- UpdateCallback **getUpdateCallback** () const
- Result run (Flags flags=DefaultFlags)
  run dialog

5.94.1 Detailed Description

The DialogColor class provides a platform-native color picker dialog that allows the user to select a color, with optional support for transparency. It provides functionality to set and get the current color, set the dialog position and title, and define callbacks for when the color changes or for updates.

5.95 Tellusim::DialogDirectory Class Reference

```
#include <interface/TellusimDialogs.h>
```

Public Types

- enum Flags {
  FlagNone = 0,
  FlagMouse = (1 << 0),
  DefaultFlags = FlagNone,
  NumFlags = 1 }

  dialog flags
- enum Result {
  ResultCancel = 0,
  ResultOk,
  NumResults }

  dialog result
- using UpdateCallback = Function< bool()>

  update callback

Public Member Functions

- **DialogDirectory** (const char *title=nullptr, const char *name=nullptr)
- **DialogDirectory** (const String &title, const char *name=nullptr)
- **DialogDirectory** (const char *title, const String &name)
- **DialogDirectory** (const String &title, const String &name)
- void setPosition (int32_t x, int32_t y)

  dialog position
- int32_t **getPositionX** () const
- int32_t **getPositionY** () const
- void setTitle (const char *title)

  dialog title
- void **setTitle** (const String &title)
- String **getTitle** () const
- void setDirectory (const char *name)

  dialog directory
- void **setDirectory** (const String &name)
- String **getDirectory** () const
- void **setUpdateCallback** (const UpdateCallback &func)
- UpdateCallback **getUpdateCallback** () const
- Result run (Flags flags=DefaultFlags)

  run dialog

5.95.1 Detailed Description

The DialogDirectory class provides a platform-native directory selection dialog, allowing users to choose a directory location. It supports setting the dialog position, title, and initial directory to be displayed.

5.96 Tellusim::DialogFileOpen Class Reference

```
#include <interface/TellusimDialogs.h>
```

Public Types

- enum Flags {
  FlagNone = 0,
  FlagHidden = (1 << 0),
  FlagMouse = (1 << 1),
  DefaultFlags = FlagNone,
  NumFlags = 2 }
  dialog flags
- enum Result {
  ResultCancel = 0,
  ResultOk,
  NumResults }
  dialog result
- using UpdateCallback = Function< bool()>
  update callback

Public Member Functions

- **DialogFileOpen** (const char *title=nullptr, const char *name=nullptr)
- **DialogFileOpen** (const String &title, const char *name=nullptr)
- **DialogFileOpen** (const char *title, const String &name)
- **DialogFileOpen** (const String &title, const String &name)
- void setPosition (int32_t x, int32_t y)
  dialog position
- int32_t **getPositionX** () const
- int32_t **getPositionY** () const
- void setTitle (const char *title)
  dialog title
- void **setTitle** (const String &title)
- String **getTitle** () const
- void setFilter (const char *filter)
  dialog filter
- void **setFilter** (const String &filter)
- String **getFilter** () const
- void setFile (const char *name)
  dialog file
- void **setFile** (const String &name)
- String **getFile** () const
- void **setUpdateCallback** (const UpdateCallback &func)
- UpdateCallback **getUpdateCallback** () const
- Result run (Flags flags=DefaultFlags)
  run dialog

5.96.1 Detailed Description

The DialogFileOpen class provides a platform-native file open dialog, allowing users to select a file from their system. It supports setting the dialog position, title, filter, and the initial file to be displayed.

5.97 Tellusim::DialogFileSave Class Reference

```
#include <interface/TellusimDialogs.h>
```

Public Types

- enum Flags {
  FlagNone = 0,
  FlagHidden = (1 << 0),
  FlagOverwrite = (1 << 1),
  FlagMouse = (1 << 2),
  DefaultFlags = (FlagOverwrite),
  NumFlags = 3 }

  dialog flags
- enum Result {
  ResultCancel = 0,
  ResultOk,
  NumResults }

  dialog result
- using UpdateCallback = Function< bool()>

  update callback

Public Member Functions

- **DialogFileSave** (const char *title=nullptr, const char *name=nullptr)
- **DialogFileSave** (const String &title, const char *name=nullptr)
- **DialogFileSave** (const char *title, const String &name)
- **DialogFileSave** (const String &title, const String &name)
- void setPosition (int32_t x, int32_t y)

  dialog position
- int32_t **getPositionX** () const
- int32_t **getPositionY** () const
- void setTitle (const char *title)

  dialog title
- void **setTitle** (const String &title)
- String **getTitle** () const
- void setFilter (const char *filter)

  dialog filter
- void **setFilter** (const String &filter)
- String **getFilter** () const
- void setFile (const char *name)

  dialog file
- void **setFile** (const String &name)
- String **getFile** () const
- void **setUpdateCallback** (const UpdateCallback &func)
- UpdateCallback **getUpdateCallback** () const
- Result run (Flags flags=DefaultFlags)

  run dialog

5.97.1 Detailed Description

The DialogFileSave class provides a platform-native file save dialog, enabling users to specify a location and name for saving a file. It supports setting the dialog position, title, filter, and the initial file name to be displayed.

5.98 Tellusim::DialogMenu Class Reference

`#include <interface/TellusimDialogs.h>`

Public Types

- enum Flags {
  FlagNone = 0,
  FlagMouse = (1 << 0),
  DefaultFlags = FlagNone,
  NumFlags = 1 }
  dialog flags
- enum Result {
  ResultCancel = 0,
  ResultClick,
  NumResults }
  dialog result
- using ClickedCallback = Function< void()>
  click item
- using ChangedCallback = Function< void(bool)>
  check item
- using UpdateCallback = Function< bool()>
  update callback

Public Member Functions

- void setPosition (int32_t x, int32_t y)
  dialog position
- int32_t **getPositionX** () const
- int32_t **getPositionY** () const
- uint32_t getNumItems () const
  number of items
- void setItemText (uint32_t index, const char *text)
  item text
- void **setItemText** (uint32_t index, const String &text)
- String **getItemText** (uint32_t index) const
- void setItemKey (uint32_t index, const char *key)
  item key
- String **getItemKey** (uint32_t index) const
- void setItemImage (uint32_t index, const Image &image)
  item image
- Image **getItemImage** (uint32_t index) const
- void setItemChecked (uint32_t index, bool checked, bool callback=false)
  item checked
- bool **isItemChecked** (uint32_t index) const
- void setItemEnabled (uint32_t index, bool enabled)
  item enabled
- bool **isItemEnabled** (uint32_t index) const
- void setItemHidden (uint32_t index, bool hidden)
  item hidden
- bool **isItemHidden** (uint32_t index) const

- void setItemsGroup (uint32_t index, uint32_t size)

  item group
- uint32_t **getItemGroupIndex** (uint32_t index) const
- uint32_t **getItemGroupSize** (uint32_t index) const
- uint32_t addItem (const char *text, const char *key=nullptr)

  text item
- uint32_t **addItem** (const String &text, const char *key=nullptr)
- uint32_t **addItem** (const char *text, const Image &image, const char *key=nullptr)
- uint32_t **addItem** (const String &text, const Image &image, const char *key=nullptr)
- uint32_t **addItem** (const char *text, const ClickedCallback &func, const char *key=nullptr)
- uint32_t **addItem** (const String &text, const ClickedCallback &func, const char *key=nullptr)
- uint32_t **addItem** (const char *text, const Image &image, const ClickedCallback &func, const char *key=nullptr)
- uint32_t **addItem** (const String &text, const Image &image, const ClickedCallback &func, const char *key=nullptr)
- ClickedCallback **getItemClickedCallback** (uint32_t index) const
- uint32_t **addItem** (const char *text, bool checked, const ChangedCallback &func, const char *key=nullptr)
- uint32_t **addItem** (const String &text, bool checked, const ChangedCallback &func, const char *key=nullptr)
- uint32_t **addItem** (const char *text, const Image &image, bool checked, const ChangedCallback &func, const char *key=nullptr)
- uint32_t **addItem** (const String &text, const Image &image, bool checked, const ChangedCallback &func, const char *key=nullptr)
- ChangedCallback **getItemChangedCallback** (uint32_t index) const
- void **setUpdateCallback** (const UpdateCallback &func)
- UpdateCallback **getUpdateCallback** () const
- Result run (Flags flags=DefaultFlags)

  run dialog

5.98.1 Detailed Description

The DialogMenu class provides a platform-native customizable menu dialog, allowing users to create and manage a list of interactive menu items. It supports various features such as setting the position and text of items, associating keys and images with items, and enabling or disabling them. Additionally, it allows for checking/unchecking items, hiding them, and grouping them together.

5.99 Tellusim::DialogMessage Class Reference

```
#include <interface/TellusimDialogs.h>
```

Public Types

- enum Flags {
  FlagNone = 0,
  FlagYes = (1 << 0),
  FlagNo = (1 << 1),
  FlagOk = (1 << 2),
  FlagCancel = (1 << 3),
  FlagClose = (1 << 4),
  FlagMessage = (1 << 5),
  FlagWarning = (1 << 6),
  FlagQuestion = (1 << 7),

FlagError = (1 << 8),
FlagMouse = (1 << 9),
FlagYesNo = (FlagYes | FlagNo),
FlagOkCancel = (FlagOk | FlagCancel),
DefaultFlags = (FlagOk),
NumFlags = 10 }

dialog flags

- enum Result {
ResultClose = 0,
ResultCancel,
ResultOk,
ResultNo,
ResultYes,
NumResults }

  dialog result
- using UpdateCallback = Function< bool()>

  update callback

Public Member Functions

- **DialogMessage** (const char *title=nullptr, const char *message=nullptr)
- **DialogMessage** (const String &title, const char *message=nullptr)
- **DialogMessage** (const char *title, const String &message)
- **DialogMessage** (const String &title, const String &message)
- void setPosition (int32_t x, int32_t y)

  dialog position
- int32_t **getPositionX** () const
- int32_t **getPositionY** () const
- void setTitle (const char *title)

  dialog title
- void **setTitle** (const String &title)
- String **getTitle** () const
- void setMessage (const char *message)

  dialog message
- void **setMessage** (const String &message)
- String **getMessage** () const
- void **setUpdateCallback** (const UpdateCallback &func)
- UpdateCallback **getUpdateCallback** () const
- Result run (Flags flags=DefaultFlags)

  run dialog

5.99.1 Detailed Description

The DialogMessage class provides a platform-native, customizable message dialog that displays a title and message, allowing for user interaction. It supports setting the dialog position, title, and message, as well as defining update callbacks. The class includes various flags to customize the dialog buttons and appearance.

5.100 Tellusim::DialogProgress Class Reference

```
#include <interface/TellusimDialogs.h>
```

Public Types

- enum Flags {
  FlagNone = 0,
  FlagMouse = (1 << 0),
  DefaultFlags = FlagNone,
  NumFlags = 1 }
  dialog flags
- enum Result {
  ResultCancel = 0,
  ResultOk,
  NumResults }
  dialog result

Public Member Functions

- **DialogProgress** (const char *title=nullptr, const char *message=nullptr)
- **DialogProgress** (const String &title, const char *message=nullptr)
- **DialogProgress** (const char *title, const String &message)
- **DialogProgress** (const String &title, const String &message)
- void setPosition (int32_t x, int32_t y)
  dialog position
- int32_t **getPositionX** () const
- int32_t **getPositionY** () const
- void setTitle (const char *title)
  dialog title
- void **setTitle** (const String &title)
- String **getTitle** () const
- void setMessage (const char *message)
  dialog message
- void **setMessage** (const String &message)
- String **getMessage** () const
- void setProgress (uint32_t progress)
  dialog progress
- uint32_t **getProgress** () const
- Result run (Flags flags=DefaultFlags)
  run dialog
- void close ()
  close dialog

5.100.1 Detailed Description

The DialogProgress class provides a platform-native progress dialog that displays a title, message, and progress bar to indicate ongoing operations. It allows setting the dialog position, title, message, and progress percentage.

5.101 Tellusim::Directory Class Reference

```
#include <core/TellusimDirectory.h>
```

Public Types

- enum Attributes {
  AttributeNone = 0,
  AttributeRead = (1 << 0),
  AttributeWrite = (1 << 1),
  AttributeHidden = (1 << 2),
  AttributeExecute = (1 << 3),
  AttributeTemporary = (1 << 4),
  NumAttributes = 5 }

  attributes

Public Member Functions

- bool open (const char *name, bool children=false)

  open/close directory
- bool **open** (const String &name, bool children=false)
- void **close** ()
- bool isOpened () const

  directory status
- String **getName** () const
- uint32_t getNumFiles () const

  files
- String **getFileName** (uint32_t index) const
- Attributes **getFileAttributes** (uint32_t index) const
- uint64_t **getFileMTime** (uint32_t index) const
- uint64_t **getFileATime** (uint32_t index) const
- uint64_t **getFileCTime** (uint32_t index) const
- size_t **getFileSize** (uint32_t index) const
- const Array< String > **getFiles** () const
- uint32_t getNumDirectories () const

  directories
- String **getDirectoryName** (uint32_t index) const
- Attributes **getDirectoryAttributes** (uint32_t index) const
- uint64_t **getDirectoryCTime** (uint32_t index) const
- uint32_t **getDirectorySize** (uint32_t index) const
- const Array< String > **getDirectories** () const

Static Public Member Functions

- static bool isFile (const char *name)

  file utils
- static bool **isFile** (const String &name)
- static bool **setFileAttributes** (const char *name, Attributes attributes)
- static Attributes **getFileAttributes** (const char *name)
- static bool **setFileMTime** (const char *name, uint64_t time)
- static uint64_t **getFileMTime** (const char *name)
- static uint64_t **getFileATime** (const char *name)
- static uint64_t **getFileCTime** (const char *name)
- static size_t **getFileSize** (const char *name)
- static size_t **getFileSize** (const String &name)
- static bool **removeFile** (const char *name)
- static bool **removeFile** (const String &name)

- static bool **copyFile** (const char *name, const char *new_name, bool attributes=false)
- static bool **copyFile** (const String &name, const String &new_name, bool attributes=false)
- static bool isDirectory (const char *name)

  directory utils
- static bool **isDirectory** (const String &name)
- static bool **changeDirectory** (const char *name)
- static bool **changeDirectory** (const String &name)
- static bool **createDirectory** (const char *name, bool children=false)
- static bool **createDirectory** (const String &name, bool children=false)
- static bool **removeDirectory** (const char *name, bool children=false)
- static bool **removeDirectory** (const String &name, bool children=false)
- static bool **copyDirectory** (const char *name, const char *new_name, bool attributes=false)
- static bool **copyDirectory** (const String &name, const String &new_name, bool attributes=false)
- static bool **rename** (const char *name, const char *new_name)
- static bool **rename** (const String &name, const String &new_name)
- static String getCurrentDirectory ()

  common directories
- static String **getHomeDirectory** ()
- static String **getTempDirectory** ()
- static String **getConfigDirectory** ()
- static String **getDocumentsDirectory** ()
- static const Array< String > getDriveNames ()

  drive utils

5.101.1 Detailed Description

The Directory class provides an interface for managing and interacting with directories and files within a file system. It supports operations such as opening and closing directories, retrieving directory and file attributes, and querying metadata like modification time, size, and access time. The class allows access to both files and subdirectories, with methods to retrieve names, attributes, and sizes, as well as to manipulate these entities. It includes utility functions to check if an entity is a file or directory and to handle common directories. Additionally, the Directory class provides static methods for performing file and directory management tasks. This class is essential for applications that need to navigate and manipulate the file system structure.

5.102 Tellusim::Compute::DispatchIndirect Struct Reference

dispatch indirect parameters

```
#include <platform/TellusimCompute.h>
```

Public Attributes

- uint32_t **group_width**
- uint32_t **group_height**
- uint32_t **group_depth**
- uint32_t **padding**

5.102.1 Detailed Description

dispatch indirect parameters

5.103 Tellusim::BitonicSort::DispatchParameters Struct Reference

Public Attributes

- uint32_t **keys_offset**
- uint32_t **data_offset**
- uint32_t **size**
- uint32_t **padding**

5.104 Tellusim::PrefixScan::DispatchParameters Struct Reference

Public Attributes

- uint32_t **offset**
- uint32_t **size**
- uint32_t **padding_0**
- uint32_t **padding_1**

5.105 Tellusim::RadixSort::DispatchParameters Struct Reference

Public Attributes

- uint32_t **keys_offset**
- uint32_t **data_offset**
- uint32_t **size**
- uint32_t **padding**

5.106 Tellusim::SpatialGrid::DispatchParameters Struct Reference

Public Attributes

- uint32_t **offset**
- uint32_t **size**
- uint32_t **padding_0**
- uint32_t **padding_1**

5.107 Tellusim::SpatialTree::DispatchParameters Struct Reference

Public Attributes

- uint32_t **offset**
- uint32_t **size**
- uint32_t **padding_0**
- uint32_t **padding_1**

5.108 Tellusim::Command::DrawArraysIndirect Struct Reference

draw arrays indirect parameters

`#include <platform/TellusimCommand.h>`

Public Attributes

- uint32_t **num_vertices**
- uint32_t **num_instances**
- uint32_t **base_vertex**
- uint32_t **base_instance**

5.108.1 Detailed Description

draw arrays indirect parameters

5.109 Tellusim::Command::DrawElementsIndirect Struct Reference

draw elements indirect parameters

`#include <platform/TellusimCommand.h>`

Public Attributes

- uint32_t **num_indices**
- uint32_t **num_instances**
- uint32_t **base_index**
- int32_t **base_vertex**
- uint32_t **base_instance**

5.109.1 Detailed Description

draw elements indirect parameters

5.110 Tellusim::Command::DrawMeshIndirect Struct Reference

draw mesh indirect parameters

`#include <platform/TellusimCommand.h>`

Public Attributes

- uint32_t **group_width**
- uint32_t **group_height**
- uint32_t **group_depth**
- uint32_t **padding**

5.110.1 Detailed Description

draw mesh indirect parameters

5.111 Tellusim::EncoderASTC Class Reference

`#include <graphics/TellusimEncoderASTC.h>`

Public Types

- enum Mode {
  ModeASTC44RGBAu8n = 0,
  ModeASTC54RGBAu8n,
  ModeASTC55RGBAu8n,
  NumModes }
  Encoder modes.
- enum Flags {
  FlagNone = 0,
  FlagASTC44RGBAu8n = (1 << ModeASTC44RGBAu8n),
  FlagASTC54RGBAu8n = (1 << ModeASTC54RGBAu8n),
  FlagASTC55RGBAu8n = (1 << ModeASTC55RGBAu8n),
  FlagCube = (1 << (NumModes + 1)),
  FlagsAll = (FlagASTC44RGBAu8n | FlagASTC54RGBAu8n | FlagASTC55RGBAu8n) }
  Encoder flags.

Public Member Functions

- void clear ()
  clear encoder
- bool isCreated (Mode mode) const
  check encoder
- bool create (const Device &device, Mode mode)
  create encoder
- bool **create** (const Device &device, Flags flags)
- bool dispatch (Compute &compute, Mode mode, Texture &dest, Texture &src, const Slice &dest_slice, const Slice &src_slice, uint32_t components=4) const
- bool **dispatch** (Compute &compute, Mode mode, Texture &dest, Texture &src, const Slice &src_slice, uint32_t components=4) const
- bool **dispatch** (Compute &compute, Mode mode, Texture &dest, Texture &src, uint32_t components=4) const

5.111.1 Detailed Description

The EncoderASTC class provides a compute-based interface for compressing textures into ASTC formats with support for 4x4, 5x4, and 5x5 block sizes in RGBAu8n format.

5.111.2 Member Function Documentation

5.111.2.1 dispatch()

```
bool Tellusim::EncoderASTC::dispatch (
            Compute & compute,
            Mode mode,
            Texture & dest,
            Texture & src,
            const Slice & dest_slice,
            const Slice & src_slice,
            uint32_t components = 4 ) const
```

dispatch texture encoder

Parameters

mode	Compression mode.
dest	Destination proxy texture.
src	Source texture to compress.
dest_slice	Destination texture slice.
src_slice	Source texture slice.
components	Number of components.

5.112 Tellusim::EncoderBC15 Class Reference

```
#include <graphics/TellusimEncoderBC15.h>
```

Public Types

- enum Mode {
  ModeBC1RGBu8n = 0,
  ModeBC2RGBAu8n,
  ModeBC3RGBAu8n,
  ModeBC4Ru8n,
  ModeBC5RGu8n,
  NumModes }

  Encoder modes.
- enum Flags {
  FlagNone = 0,
  FlagBC1RGBu8n = (1 << ModeBC1RGBu8n),
  FlagBC2RGBAu8n = (1 << ModeBC2RGBAu8n),
  FlagBC3RGBAu8n = (1 << ModeBC3RGBAu8n),
  FlagBC4Ru8n = (1 << ModeBC4Ru8n),
  FlagBC5RGu8n = (1 << ModeBC5RGu8n),
  FlagCube = (1 << (NumModes + 1)),
  FlagsBC13 = (FlagBC1RGBu8n | FlagBC2RGBAu8n | FlagBC3RGBAu8n),
  FlagsBC45 = (FlagBC4Ru8n | FlagBC5RGu8n),
  FlagsAll = (FlagsBC13 | FlagsBC45) }

  Encoder flags.

Public Member Functions

- void clear ()
  clear encoder
- bool isCreated (Mode mode) const
  check encoder
- bool create (const Device &device, Mode mode)
  create encoder
- bool **create** (const Device &device, Flags flags)
- bool dispatch (Compute &compute, Mode mode, Texture &dest, Texture &src, const Slice &dest_slice, const Slice &src_slice) const
- bool **dispatch** (Compute &compute, Mode mode, Texture &dest, Texture &src, const Slice &src_slice) const
- bool **dispatch** (Compute &compute, Mode mode, Texture &dest, Texture &src) const

5.112.1 Detailed Description

The EncoderBC15 class implements GPU-based texture compression using BC1-BC5 (S3TC) formats, supporting both color (RGB/RGBA) and single or dual-channel compression modes.

5.112.2 Member Function Documentation

5.112.2.1 dispatch()

```
bool Tellusim::EncoderBC15::dispatch (
             Compute & compute,
             Mode mode,
             Texture & dest,
             Texture & src,
             const Slice & dest_slice,
             const Slice & src_slice ) const
```

dispatch texture encoder

Parameters

mode	Compression mode.
dest	Destination proxy texture.
src	Source texture to compress.
dest_slice	Destination texture slice.
src_slice	Source texture slice.

5.113 Tellusim::EncoderBC67 Class Reference

```
#include <graphics/TellusimEncoderBC67.h>
```

Public Types

- enum Mode {
  ModeBC6RGBf16s = 0,
  ModeBC6RGBf16u,
  ModeBC7RGBAu8n,
  NumModes }
  Encoder modes.
- enum Flags {
  FlagNone = 0,
  FlagBC6RGBf16s = (1 << ModeBC6RGBf16s),
  FlagBC6RGBf16u = (1 << ModeBC6RGBf16u),
  FlagBC7RGBAu8n = (1 << ModeBC7RGBAu8n),
  FlagCube = (1 << (NumModes + 1)),
  FlagsAll = (FlagBC6RGBf16s | FlagBC6RGBf16u | FlagBC7RGBAu8n) }
  Encoder flags.

Public Member Functions

- void clear ()
  clear encoder
- bool isCreated (Mode mode) const
  check encoder
- bool create (const Device &device, Mode mode)
  create encoder
- bool **create** (const Device &device, Flags flags)
- bool dispatch (Compute &compute, Mode mode, Texture &dest, Texture &src, const Slice &dest_slice, const Slice &src_slice, uint32_t components=4) const
- bool **dispatch** (Compute &compute, Mode mode, Texture &dest, Texture &src, const Slice &src_slice, uint32_t components=4) const
- bool **dispatch** (Compute &compute, Mode mode, Texture &dest, Texture &src, uint32_t components=4) const

5.113.1 Detailed Description

The EncoderBC67 class provides GPU-based texture compression using BC6H and BC7 formats, supporting high dynamic range (HDR) and standard color data encoding in both signed and unsigned formats.

5.113.2 Member Function Documentation

5.113.2.1 dispatch()

```
bool Tellusim::EncoderBC67::dispatch (
            Compute & compute,
            Mode mode,
            Texture & dest,
            Texture & src,
            const Slice & dest_slice,
            const Slice & src_slice,
            uint32_t components = 4 ) const
```

dispatch texture encoder

Parameters

mode	Compression mode.
dest	Destination proxy texture.
src	Source texture to compress.
dest_slice	Destination texture slice.
src_slice	Source texture slice.
components	Number of components.

5.114 Tellusim::f32u32 Union Reference

`#include <math/TellusimFloat.h>`

Public Member Functions

- **f32u32** (float32_t f)
- **f32u32** (uint32_t u)
- **f32u32** (int32_t i)
- uint32_t exponent () const

  access to number
- uint32_t **mantissa** () const

Public Attributes

- float32_t **f**
- uint32_t **u**
- int32_t **i**

5.114.1 Detailed Description

32-bit floating-point as integer

5.115 Tellusim::f64u64 Union Reference

`#include <math/TellusimFloat.h>`

Public Member Functions

- **f64u64** (float64_t f)
- **f64u64** (uint64_t u)
- **f64u64** (int64_t i)
- uint64_t exponent () const

  access to number
- uint64_t **mantissa** () const

Public Attributes

- float64_t **f**
- uint64_t **u**
- int64_t **i**

5.115.1 Detailed Description

64-bit floating-point as integer

5.116 Tellusim::Face Struct Reference

```
#include <TellusimTypes.h>
```

Public Member Functions

- **Face** (uint32_t base)
- **Face** (uint32_t base, uint32_t size)

Public Attributes

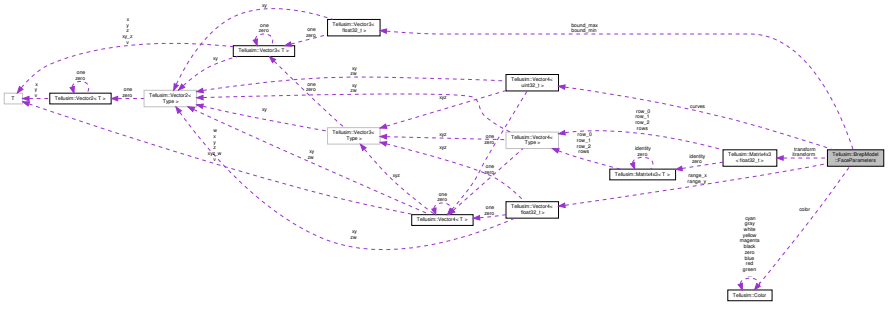
- uint32_t **base** = 0
- uint32_t **size** = 1

5.116.1 Detailed Description

The Face struct represents a single face of a cube image. It contains two members: base, which specifies the base index of the face, and size, which indicates the number of faces. It is typically used in the context of cube maps, where each face of the cube is treated as an individual image.

5.117 Tellusim::BrepModel::FaceParameters Struct Reference

Collaboration diagram for Tellusim::BrepModel::FaceParameters:



Public Attributes

- Matrix4x3f **transform**
- Matrix4x3f **itransform**
- Vector3f **bound_min**
- float32_t **brep_winding**
- Vector3f **bound_max**
- float32_t **face_winding**
- Color **color**
- Vector4u **curves**
- Vector4f **range_x**
- Vector4f **range_y**
-

```
union {
  struct {
    float32_t radius
  } sphere
  struct {
    float32_t radius
  } cylinder
  struct {
    float32_t radius
    float32_t angle
  } cone
  struct {
    float32_t radius_0
    float32_t radius_1
  } torus
  struct {
    float32_t radius_0
    float32_t radius_1
  } lemon
  struct {
    float32_t height
    float32_t pcoord_y
    uint32_t vertex_index
    uint32_t padding
    float32_t length [2]
  } revolve
  struct {
    float32_t width
    float32_t pcoord_x
    uint32_t vertex_index
    uint32_t padding
    float32_t direction [3]
  } extrude
  struct {
    float32_t width
    float32_t height
    float32_t pcoord_x
    float32_t pcoord_y
    uint32_t vertex_index
    uint32_t padding [3]
    float32_t length [4]
  } surface
  struct {
    float32_t parameters_0 [4]
```

```
        float32_t parameters_1 [4]
        float32_t parameters_2 [4]
    }
};
```

5.118 Tellusim::Device::Features Struct Reference

device features

`#include <platform/TellusimDevice.h>`

Public Attributes

- bool **threadAccess**
- bool **sparseBuffer**
- bool **bufferTable**
- bool **sparseTexture**
- bool **sparseArrayTexture**
- bool **cubeArrayTexture**
- bool **textureTable**
- bool **baseInstanceIndex**
- bool **drawIndirectIndex**
- bool **drawIndirectCount**
- bool **taskIndirectCount**
- bool **vertexStorage**
- bool **vertexIndexLayer**
- bool **geometryPassthrough**
- bool **fragmentBarycentric**
- bool **fragmentStencilExport**
- bool **dualSourceBlending**
- bool **depthRangeOneToOne**
- bool **conservativeRaster**
- bool **conditionalRendering**
- bool **rayTracing**
- bool **computeTracing**
- bool **fragmentTracing**
- bool **indirectTracing**
- uint32_t **recursionDepth**
- bool **subgroupVote**
- bool **subgroupMath**
- bool **subgroupShuffle**
- uint32_t **subgroupSize**
- uint32_t **minSubgroupSize**
- uint32_t **maxSubgroupSize**
- bool **shaderu8**
- bool **shaderf16**
- bool **shaderu16**
- bool **shaderf64**
- bool **shaderu64**
- bool **atomicGroupf32**
- bool **atomicGroupu64**
- bool **atomicBufferf32**

- bool **atomicBufferu64**
- bool **atomicTexturef32**
- bool **atomicTextureu32**
- bool **atomicTextureu64**
- bool **matrix16f16**
- bool **matrix16x8x8f16**
- bool **matrix16x8x16f16**
- bool **matrix16f16f32**
- bool **matrix16x8x8f16f32**
- bool **matrix16x8x16f16f32**
- uint32_t **uniformAlignment**
- uint32_t **storageAlignment**
- uint32_t **maxTextureSamples**
- uint32_t **maxTexture2DSize**
- uint32_t **maxTexture3DSize**
- uint32_t **maxTextureLayers**
- uint32_t **maxGroupSizeX**
- uint32_t **maxGroupSizeY**
- uint32_t **maxGroupSizeZ**
- uint32_t **maxGroupCountX**
- uint32_t **maxGroupCountY**
- uint32_t **maxGroupCountZ**
- uint32_t **maxTaskCount**
- uint32_t **maxTaskMemory**
- uint32_t **maxTaskMeshes**
- uint32_t **maxMeshMemory**
- uint32_t **maxMeshVertices**
- uint32_t **maxMeshPrimitives**
- uint32_t **maxViewportCount**
- uint32_t **maxClipCullCount**
- uint64_t **maxUniformSize**
- uint64_t **maxStorageSize**
- uint32_t **groupMemory**
- uint64_t **videoMemory**
- uint32_t **vendorID**
- uint32_t **deviceID**
- uint32_t **pciBusID**
- uint32_t **pciDomainID**
- uint32_t **pciDeviceID**

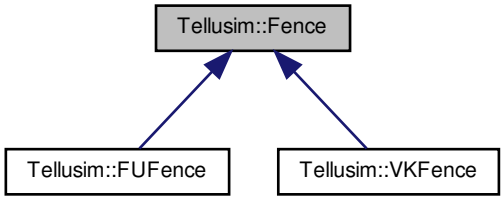
5.118.1 Detailed Description

device features

5.119 Tellusim::Fence Class Reference

`#include <platform/TellusimFence.h>`

Inheritance diagram for Tellusim::Fence:



Public Types

- enum Flags {
  FlagNone = 0,
  FlagSemaphore = (1 << 0),
  FlagSignaled = (1 << 1),
  FlagShared = (1 << 2),
  FlagExtern = (1 << 3),
  DefaultFlags = FlagNone,
  NumFlags = 4 }

  Fence flags.

Public Member Functions

- Platform getPlatform () const

  fence platform
- const char * **getPlatformName** () const
- uint32_t getIndex () const

  fence device index
- void clear ()

  clear fence
- bool isCreated () const

  check fence
- bool create (Flags flags=DefaultFlags)

  create fence
- Flags getFlags () const

  fence flags
- bool **hasFlag** (Flags flags) const
- bool **hasFlags** (Flags flags) const
- String **getFlagsName** () const
- String getDescription () const

  fence description

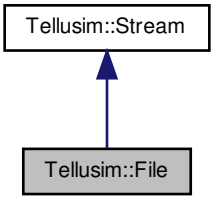
5.119.1 Detailed Description

The Fence class provides functionality for managing synchronization fences within a graphics or computing pipeline. It includes methods for querying platform and device-specific information, as well as creating and managing the state of a fence. Fences can be used to synchronize operations across different parts of the pipeline, ensuring that specific tasks are completed before others are initiated.

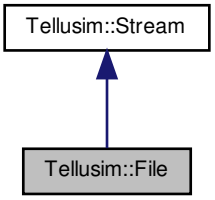
5.120 Tellusim::File Class Reference

`#include <core/TellusimFile.h>`

Inheritance diagram for Tellusim::File:



Collaboration diagram for Tellusim::File:



Public Member Functions

- bool open (const char *name, const char *mode)

  open/close file
- bool **open** (const String &name, const char *mode)
- bool **open** (int32_t fd, const char *name, const char *mode)
- bool **popen** (const char *command, const char *mode)
- bool **popen** (const String &command, const char *mode)
- void **close** ()

Static Public Member Functions

- static bool isFile (const char *name)

  file utils
- static bool **isFile** (const String &name)
- static uint64_t **getMTime** (const char *name)
- static size_t **getSize** (const char *name)
- static bool **remove** (const char *name)

5.120.1 Detailed Description

The File class extends the Stream class and provides a specialized implementation for managing files in a system. It offers functionality for opening, reading, writing, and closing files, along with managing the file status, position, and size. The class supports file operations using both file descriptors and file names, and it includes utilities for retrieving file status and performing manipulations.

5.121 Tellusim::FixedPool< Type, Index > Class Template Reference

`#include <core/TellusimPool.h>`

Public Member Functions

- **FixedPool** (uint32_t size)
- void init (uint32_t size)

  initialize pool
- void shutdown ()

  shutdown pool
- bool isInitialized ()

  pool status
- uint32_t allocate ()

  allocate object memory
- void free (uint32_t offset)

  free object memory
- uint32_t getMemory () const

  memory usage in bytes
- uint32_t getAllocations () const

  number of allocations

5.121.1 Detailed Description

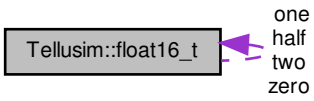
template<class Type, class Index = uint32_t>
class Tellusim::FixedPool< Type, Index >

Fixed pool memory allocator

5.122 Tellusim::float16_t Struct Reference

`#include <math/TellusimFloat.h>`

Collaboration diagram for Tellusim::float16_t:



Public Member Functions

- **float16_t** (uint16_t u)
- **float16_t** (float32_t f)
- **float16_t** (float64_t f)
- void set (float32_t f)

  handle normal, denormal, infinity and NaN cases
- float32_t **get** () const
- void setFast (float32_t f)

  handle normal cases only
- float32_t **getFast** () const
- operator uint16_t () const

  conversion to numbers
- **operator float32_t** () const
- **operator float64_t** () const
- float16_t & operator= (uint16_t u)

  conversion from numbers
- float16_t & **operator=** (float32_t f)
- float16_t & **operator=** (float16_t f)
- uint16_t exponent () const

  access to number
- uint16_t **mantissa** () const

Public Attributes

- uint16_t **bits**

Static Public Attributes

- static const float16_t zero

  constant values
- static const float16_t **half**
- static const float16_t **one**
- static const float16_t **two**

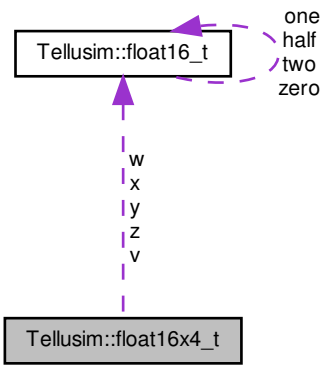
5.122.1 Detailed Description

16-bit floating-point number class mantissa bits 10, exponent bits 5 strict conversion is performed by default

5.123 Tellusim::float16x4_t Struct Reference

`#include <math/TellusimSimd.h>`

Collaboration diagram for Tellusim::float16x4_t:



Public Types

- enum { **Size** = 4 }

Public Member Functions

- float16x4_t (const float32x4_t &v)
- **float16x4_t** (const float16_t *v)
- **float16x4_t** (const float16_t *v, float16_t w)
- **float16x4_t** (float16_t v)
- **float16x4_t** (float16_t x, float16_t y, float16_t z, float16_t w)
- **float16x4_t** (uint64_t v)
- **float16x4_t** (const float16x4_t &v)
- void set (float16x4_t &v)

  update vector data
- void **set** (float16_t X, float16_t Y, float16_t Z, float16_t W)
- void **set** (const float16_t *1 v, float16_t W)
- void **set** (const float16_t *1 v)
- void **get** (float16_t *1 v) const
- template<uint32_t Index>
  void **set** (float16_t V)
- template<uint32_t Index>
  float16_t **get** () const
- float16x4_t & operator= (const float16x4_t &v)

  assignment operator

Public Attributes

- union {
  struct {
  float16_t **x**
  float16_t **y**
  float16_t **z**
  float16_t **w**
  } **f16**
  float16_t **v** [Size]
  uint64_t **vec**
  };

5.123.1 Detailed Description

Vector of four float16_t components

5.123.2 Constructor & Destructor Documentation

5.123.2.1 float16x4_t()

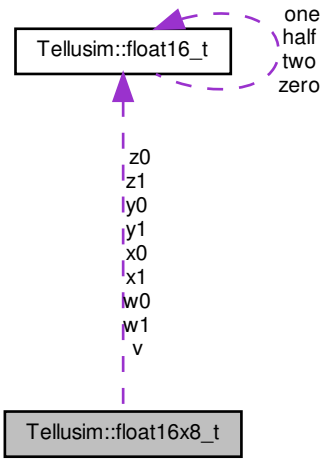
```
Tellusim::float16x4_t::float16x4_t (
            const float32x4_t & v )  [explicit]
```

Vector of four float16_t components

5.124 Tellusim::float16x8_t Struct Reference

```
#include <math/TellusimSimd.h>
```

Collaboration diagram for Tellusim::float16x8_t:



Public Types

- enum { **Size** = 8 }

Public Member Functions

- float16x8_t (const float32x8_t &v)
- **float16x8_t** (const float16_t *v)
- **float16x8_t** (const uint64_t *v)
- **float16x8_t** (uint64_t v0, uint64_t v1)
- **float16x8_t** (float16_t v)
- **float16x8_t** (const float16x4_t &v0, const float16x4_t &v1)
- **float16x8_t** (float16_t x0, float16_t y0, float16_t z0, float16_t w0, float16_t x1, float16_t y1, float16_t z1, float16_t w1)
- int16x8_t asi16x8 () const
  cast vector data
- uint16x8_t **asu16x8** () const
- int32x4_t **asi32x4** () const
- uint32x4_t **asu32x4** () const
- float32x4_t asf32x4 () const
  cast vector data
- void set (const float16x8_t &v)
  update vector data
- void **set** (float16_t X0, float16_t Y0, float16_t Z0, float16_t W0, float16_t X1, float16_t Y1, float16_t Z1, float16_t W1)
- void **set** (const float16_t *1 v)
- void **get** (float16_t *1 v) const
- void **set** (const uint64_t *1 v)
- void **get** (uint64_t *1 v) const
- template<uint32_t Index>
  void **set** (float16_t V)
- template<uint32_t Index>
  float16_t **get** () const
- float16x8_t & operator= (const float16x8_t &v)
  assignment operator

Public Attributes

-
```
union {
  struct {
    float16_t x0
    float16_t y0
    float16_t z0
    float16_t w0
    float16_t x1
    float16_t y1
    float16_t z1
    float16_t w1
  } f16
  struct {
    uint64_t vec0
    uint64_t vec1
  }
  float16_t v [Size]
  uint64_t vec [2]
};
```

5.124.1 Detailed Description

Vector of eight float16_t components

5.124.2 Constructor & Destructor Documentation

5.124.2.1 float16x8_t()

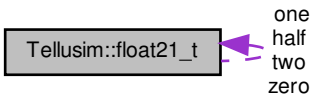
```
Tellusim::float16x8_t::float16x8_t (
            const float32x8_t & v )  [explicit]
```

Vector of eight float16_t components

5.125 Tellusim::float21_t Struct Reference

```
#include <math/TellusimFloat.h>
```

Collaboration diagram for Tellusim::float21_t:



Public Member Functions

- **float21_t** (uint32_t u)
- **float21_t** (float32_t f)
- **float21_t** (float64_t f)
- void set (float32_t f)

  handle normal, denormal, infinity and NaN cases
- float32_t **get** () const
- void setFast (float32_t f)

  handle normal cases only
- float32_t **getFast** () const
- operator uint32_t () const

  conversion to numbers
- **operator float32_t** () const
- **operator float64_t** () const
- float21_t & operator= (uint32_t u)

  conversion from numbers
- float21_t & **operator=** (float32_t f)
- float21_t & **operator=** (float21_t f)
- uint32_t exponent () const

  access to number
- uint32_t **mantissa** () const

Public Attributes

- uint32_t **bits**

Static Public Attributes

- static const float21_t zero

  constant values
- static const float21_t **half**
- static const float21_t **one**
- static const float21_t **two**

5.125.1 Detailed Description

21-bit floating-point number class mantissa bits 14, exponent bits 6 strict conversion is performed by default

5.126 Tellusim::float24_t Struct Reference

`#include <math/TellusimFloat.h>`

Collaboration diagram for Tellusim::float24_t:

Public Member Functions

- **float24_t** (uint32_t u)
- **float24_t** (float32_t f)
- **float24_t** (float64_t f)
- void set (float32_t f)

  handle normal, denormal, infinity and NaN cases
- float32_t **get** () const
- void setFast (float32_t f)

  handle normal cases only
- float32_t **getFast** () const
- operator uint32_t () const

  conversion to numbers
- **operator float32_t** () const
- **operator float64_t** () const
- float24_t & operator= (uint32_t u)

  conversion from numbers
- float24_t & **operator=** (float32_t f)
- float24_t & **operator=** (float24_t f)
- uint32_t exponent () const

  access to number
- uint32_t **mantissa** () const

Public Attributes

- uint32_t **bits**

Static Public Attributes

- static const float24_t zero
  constant values
- static const float24_t **half**
- static const float24_t **one**
- static const float24_t **two**

5.126.1 Detailed Description

24-bit floating-point number class mantissa bits 17, exponent bits 6 strict conversion is performed by default

5.127 Tellusim::float32x4_t Struct Reference

`#include <math/TellusimSimd.h>`

Public Types

- enum { **Size** = 4 }

Public Member Functions

- float32x4_t (const float64x4_t &v)
- **float32x4_t** (const float32_t *v)
- **float32x4_t** (const float32_t *v, float32_t w)
- **float32x4_t** (float32_t v)
- **float32x4_t** (const int32x4_t &v)
- **float32x4_t** (const uint32x4_t &v)
- **float32x4_t** (const float16x4_t &v)
- **float32x4_t** (float32_t x, float32_t y, float32_t z, float32_t w=0.0f)
- int32x4_t asi32x4 () const
  cast vector data
- uint32x4_t **asu32x4** () const
- int16x8_t **asi16x8** () const
- uint16x8_t **asu16x8** () const
- float16x8_t **asf16x8** () const
- void set (const float32x4_t &v)
  update vector data
- void **set** (float32_t X, float32_t Y, float32_t Z, float32_t W)
- void **set** (const float32_t *1 v, float32_t W)
- void **set** (const float32_t *1 v)
- void **get** (float32_t *1 v) const
- template<uint32_t Index>
  void **set** (float32_t V)

- template<uint32_t Index>
  float32_t **get** () const
- template<uint32_t Index>
  float32x4_t **get4** () const
- float32x4_t & operator*= (float32_t v)

  vector to scalar operators
- float32x4_t & **operator/=** (float32_t v)
- float32x4_t & **operator+=** (float32_t v)
- float32x4_t & **operator-=** (float32_t v)
- float32x4_t & operator*= (const float32x4_t &v)

  vector to vector operators
- float32x4_t & **operator/=** (const float32x4_t &v)
- float32x4_t & **operator+=** (const float32x4_t &v)
- float32x4_t & **operator-=** (const float32x4_t &v)
- float32x4_t zxyw () const

  swizzle vector
- float32x4_t **zwxy** () const
- float32x4_t **yxwz** () const
- float32_t sum () const

  sum vector components

Public Attributes

-
  union {
    struct {
      float32_t **x**
      float32_t **y**
      float32_t **z**
      float32_t **w**
    }
    float32_t **v** [Size]
  };

5.127.1 Detailed Description

Vector of four float32_t components

5.127.2 Constructor & Destructor Documentation

5.127.2.1 float32x4_t()

```
Tellusim::float32x4_t::float32x4_t (
            const float64x4_t & v )  [explicit]
```

Vector of four float32_t components

5.128 Tellusim::float32x8_t Struct Reference

`#include <math/TellusimSimd.h>`

Public Types

- enum { **Size** = 8 }

Public Member Functions

- float32x8_t (const float64x8_t &v)
- **float32x8_t** (const float32_t *v)
- **float32x8_t** (float32_t v)
- **float32x8_t** (const int32x8_t &v)
- **float32x8_t** (const uint32x8_t &v)
- **float32x8_t** (const float32x4_t &v0, const float32x4_t &v1)
- **float32x8_t** (const float16x8_t &v)
- **float32x8_t** (float32_t x0, float32_t y0, float32_t z0, float32_t w0, float32_t x1, float32_t y1, float32_t z1, float32_t w1)
- int32x8_t asi32x8 () const

  cast vector data
- uint32x8_t **asu32x8** () const
- void set (const float32x8_t &v)

  update vector data
- void **set** (float32_t X0, float32_t Y0, float32_t Z0, float32_t W0, float32_t X1, float32_t Y1, float32_t Z1, float32_t W1)
- void **set** (const float32_t *1 v)
- void **get** (float32_t *1 v) const
- template<uint32_t Index>
  void **set** (float32_t V)
- template<uint32_t Index>
  float32_t **get** () const
- template<uint32_t Index>
  float32x8_t **get8** () const
- float32x8_t & operator*= (float32_t v)

  vector to scalar operators
- float32x8_t & **operator/=** (float32_t v)
- float32x8_t & **operator+=** (float32_t v)
- float32x8_t & **operator-=** (float32_t v)
- float32x8_t & operator*= (const float32x8_t &v)

  vector to vector operators
- float32x8_t & **operator/=** (const float32x8_t &v)
- float32x8_t & **operator+=** (const float32x8_t &v)
- float32x8_t & **operator-=** (const float32x8_t &v)
- float32x8_t xyzw10 () const

  swizzle vector
- float32x8_t **zwxy01** () const
- float32x8_t **yxwz01** () const
- float32x4_t **xyzw0** () const
- float32x4_t **xyzw1** () const
- float32_t sum () const

  sum vector components

Public Attributes

- union {
  struct {
  float32_t **x0**
  float32_t **y0**
  float32_t **z0**
  float32_t **w0**
  float32_t **x1**
  float32_t **y1**
  float32_t **z1**
  float32_t **w1**
  }
  float32_t **v** [Size]
  };

5.128.1 Detailed Description

Vector of eight float32_t components

5.128.2 Constructor & Destructor Documentation

5.128.2.1 float32x8_t()

```
Tellusim::float32x8_t::float32x8_t (
            const float64x8_t & v )  [explicit]
```

Vector of eight float32_t components

5.129 Tellusim::float64x2_t Struct Reference

```
#include <math/TellusimSimd.h>
```

Public Types

- enum { **Size** = 2 }

Public Member Functions

- **float64x2_t** (const float64_t *v)
- **float64x2_t** (float64_t v)
- **float64x2_t** (float64_t x, float64_t y)
- void set (const float64x2_t &v)

  update vector data
- void **set** (float64_t X, float64_t Y)
- void **set** (const float64_t *1 v)
- void **get** (float64_t *1 v) const
- template<uint32_t Index>
  void **set** (float64_t V)
- template<uint32_t Index>
  float64_t **get** () const
- template<uint64_t Index>
  float64x2_t **get2** () const
- float64x2_t & operator*= (float64_t v)

  vector to scalar operators
- float64x2_t & **operator/=** (float64_t v)
- float64x2_t & **operator+=** (float64_t v)
- float64x2_t & **operator-=** (float64_t v)
- float64x2_t & operator*= (const float64x2_t &v)

  vector to vector operators
- float64x2_t & **operator/=** (const float64x2_t &v)
- float64x2_t & **operator+=** (const float64x2_t &v)
- float64x2_t & **operator-=** (const float64x2_t &v)
- float64x2_t yx () const

  swizzle vector
- float64_t sum () const

  sum vector components

Public Attributes

-
  union {
    struct {
      float64_t **x**
      float64_t **y**
    }
    float64_t **v** [Size]
  };

5.129.1 Detailed Description

Vector of two float64_t components

5.130 Tellusim::float64x4_t Struct Reference

`#include <math/TellusimSimd.h>`

Public Types

- enum { **Size** = 4 }

Public Member Functions

- **float64x4_t** (const float64_t *v)
- **float64x4_t** (const float64_t *v, float64_t w)
- **float64x4_t** (float64_t v)
- **float64x4_t** (const int32x4_t &v)
- **float64x4_t** (const uint32x4_t &v)
- **float64x4_t** (const float32x4_t &v)
- **float64x4_t** (const float64x2_t &v0, const float64x2_t &v1)
- **float64x4_t** (float64_t x, float64_t y, float64_t z, float64_t w=0.0)
- void set (const float64x4_t &v)

  update vector data
- void **set** (float64_t X, float64_t Y, float64_t Z, float64_t W)
- void **set** (const float64_t *1 v, float64_t W)
- void **set** (const float64_t *1 v)
- void **get** (float64_t *1 v) const
- template<uint32_t Index>
  void **set** (float64_t V)
- template<uint32_t Index>
  float64_t **get** () const
- template<uint64_t Index>
  float64x4_t **get4** () const
- float64x4_t & operator*= (float64_t v)

  vector to scalar operators
- float64x4_t & **operator/=** (float64_t v)
- float64x4_t & **operator+=** (float64_t v)
- float64x4_t & **operator-=** (float64_t v)
- float64x4_t & operator*= (const float64x4_t &v)

  vector to vector operators
- float64x4_t & **operator/=** (const float64x4_t &v)
- float64x4_t & **operator+=** (const float64x4_t &v)
- float64x4_t & **operator-=** (const float64x4_t &v)
- float64x4_t zxyw () const

  swizzle vector
- float64x4_t **zwxy** () const
- float64x4_t **yxwz** () const
- float64x2_t **xy** () const
- float64x2_t **zw** () const
- float64_t sum () const

  sum vector components

Public Attributes

-
  union {
    struct {
      float64_t **x**
      float64_t **y**
      float64_t **z**
      float64_t **w**
    }
    float64_t **v** [Size]
  };

5.130.1 Detailed Description

Vector of four float64_t components

5.131 Tellusim::float64x8_t Struct Reference

`#include <math/TellusimSimd.h>`

Public Types

- enum { **Size** = 8 }

Public Member Functions

- **float64x8_t** (const float64_t *v)
- **float64x8_t** (float64_t v)
- **float64x8_t** (const int32x8_t &v)
- **float64x8_t** (const uint32x8_t &v)
- **float64x8_t** (const float32x8_t &v)
- **float64x8_t** (const float64x4_t &v0, const float64x4_t &v1)
- **float64x8_t** (float64_t x0, float64_t y0, float64_t z0, float64_t w0, float64_t x1, float64_t y1, float64_t z1, float64_t w1)
- void set (const float64x8_t &v)

  update vector data
- void **set** (float64_t X0, float64_t Y0, float64_t Z0, float64_t W0, float64_t X1, float64_t Y1, float64_t Z1, float64_t W1)
- void **set** (const float64_t *1 v)
- void **get** (float64_t *1 v) const
- template<uint32_t Index>
  void **set** (float64_t V)
- template<uint32_t Index>
  float64_t **get** () const
- template<uint64_t Index>
  float64x8_t **get8** () const
- float64x8_t & operator*= (float64_t v)

  vector to scalar operators
- float64x8_t & **operator/=** (float64_t v)
- float64x8_t & **operator+=** (float64_t v)
- float64x8_t & **operator-=** (float64_t v)
- float64x8_t & operator*= (const float64x8_t &v)

  vector to vector operators
- float64x8_t & **operator/=** (const float64x8_t &v)
- float64x8_t & **operator+=** (const float64x8_t &v)
- float64x8_t & **operator-=** (const float64x8_t &v)
- float64x8_t xyzw10 () const

  swizzle vector
- float64x8_t **zwxy01** () const
- float64x8_t **yxwz01** () const
- float64x4_t **xyzw0** () const
- float64x4_t **xyzw1** () const
- float64_t sum () const

  sum vector components

Public Attributes

- union {
  struct {
  float64_t **x0**
  float64_t **y0**
  float64_t **z0**
  float64_t **w0**
  float64_t **x1**
  float64_t **y1**
  float64_t **z1**
  float64_t **w1**
  }
  float64_t **v** [Size]
  };

5.131.1 Detailed Description

Vector of eight float64_t components

5.132 Tellusim::Font Class Reference

`#include <interface/TellusimFont.h>`

Public Member Functions

- void clear ()
  clear font
- bool isLoaded () const
  check font
- bool load (const char *name)
  load font
- bool **load** (Stream &stream)
- float32_t getAdvance (const FontStyle &style, uint32_t code)
  glyph advance
- Rect getRect (const Vector3f &position, const FontStyle &style, const char *str)
  text rectangle
- Rect **getRect** (const Vector3f &position, const FontStyle &style, const uint32_t *str)
- template<class Type >
  Rect **getRect** (const Vector3f &position, uint32_t size, const Type *str)
- Rect getRect (const FontBatch *batches, uint32_t num_batches)
  batched text rectangle
- Rect **getRect** (const FontBatch32 *batches, uint32_t num_batches)
- void create (const Device &device, const FontStyle &style, const char *str)
  create text
- void **create** (const Device &device, const FontStyle &style, const uint32_t *str)
- template<class Type >
  void **create** (const Device &device, uint32_t size, const Type *str)
- void create (const Device &device, const FontBatch *batches, uint32_t num_batches)

create batched text

- void **create** (const Device &device, const FontBatch32 *batches, uint32_t num_batches)
- void draw (Command &command, const Vector3f &position, const FontStyle &style, const char *str)

  draw text
- void **draw** (Command &command, const Vector3f &position, const FontStyle &style, const uint32_t *str)
- template<class Type >
  void **draw** (Command *command, const Vector3f &position, uint32_t size, const Type *str)
- template<class Type >
  void **draw** (Command *command, const Vector3f &position, uint32_t size, const Color &color, const Type *str)
- void draw (Command &command, const FontBatch *batches, uint32_t num_batches)

  draw batched text
- void **draw** (Command &command, const FontBatch32 *batches, uint32_t num_batches)
- bool flush (const Device &device)

  flush textures

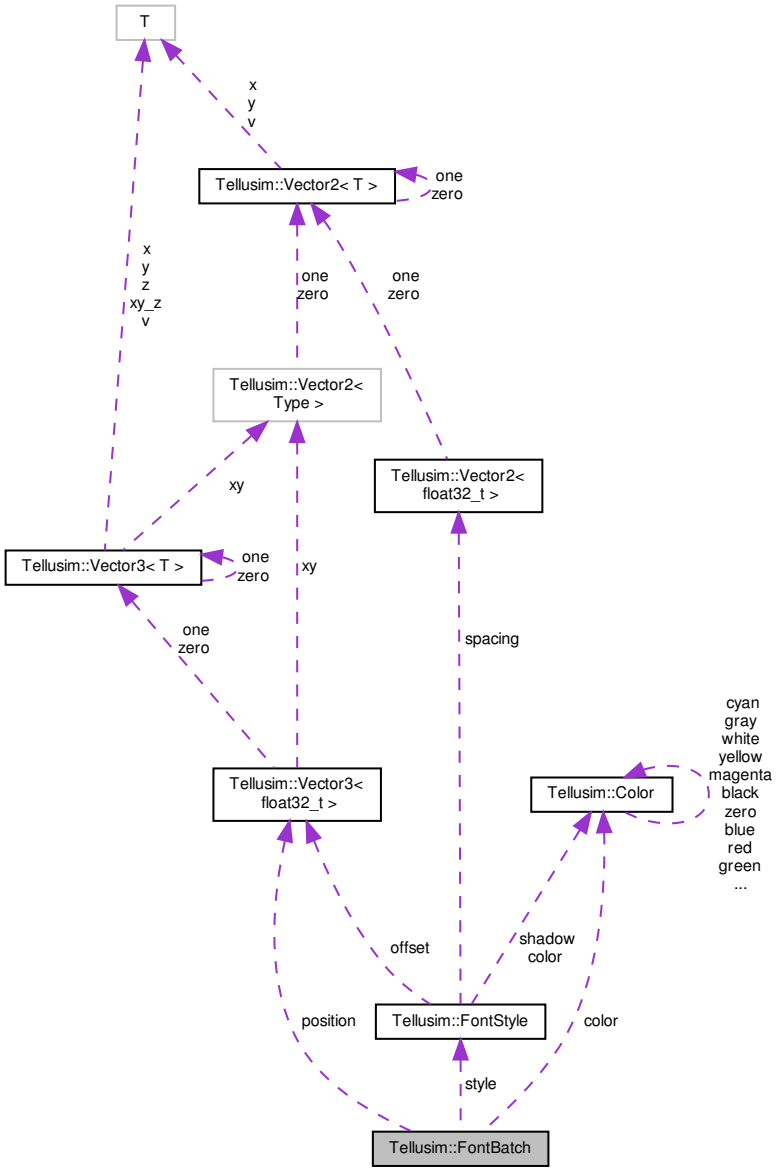
5.132.1 Detailed Description

The Font class provides functionality for loading, managing, and rendering fonts, supporting multiple font styles and sizes. It allows checking if a font is loaded, retrieving glyph advances, and calculating text bounding rectangles for both single and batched text. The class supports creating and drawing text using a specified font style, either character by character or in batches.

5.133 Tellusim::FontBatch Struct Reference

```
#include <interface/TellusimTypes.h>
```

Collaboration diagram for Tellusim::FontBatch:



Public Member Functions

- **FontBatch** (const Vector3f &position, const char *str)
- **FontBatch** (const Vector3f &position, const FontStyle *style, const char *str)

Public Attributes

- Vector3f **position** = Vector3f::zero
- Color **color** = Color::white
- const FontStyle * **style** = nullptr
- const char * **str** = nullptr

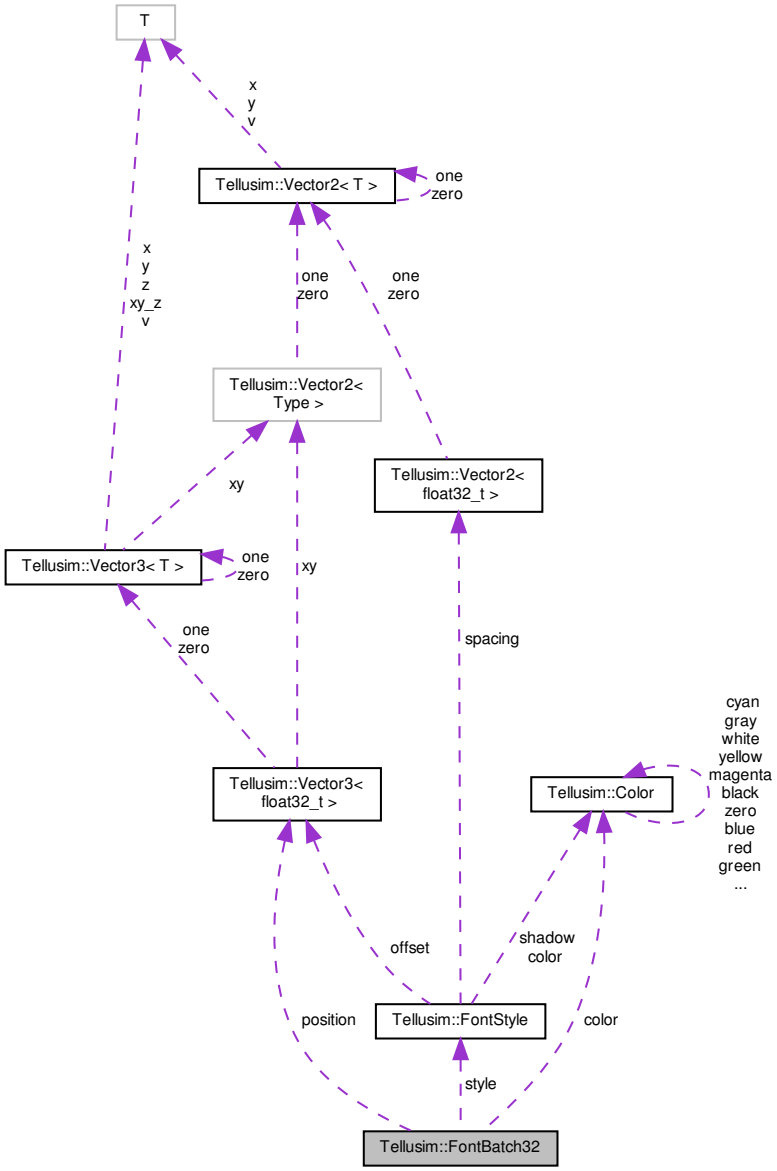
5.133.1 Detailed Description

The FontBatch struct represents a batch of UTF-8 encoded text to be rendered, with properties for positioning, color, and style. It includes constructors for setting the position and text, and optionally the font style.

5.134 Tellusim::FontBatch32 Struct Reference

```
#include <interface/TellusimTypes.h>
```

Collaboration diagram for Tellusim::FontBatch32:



Public Member Functions

- **FontBatch32** (const Vector3f &position, const uint32_t *str)
- **FontBatch32** (const Vector3f &position, const FontStyle *style, const uint32_t *str)

Public Attributes

- Vector3f **position** = Vector3f::zero
- Color **color** = Color::white
- const FontStyle * **style** = nullptr
- const uint32_t * **str** = nullptr

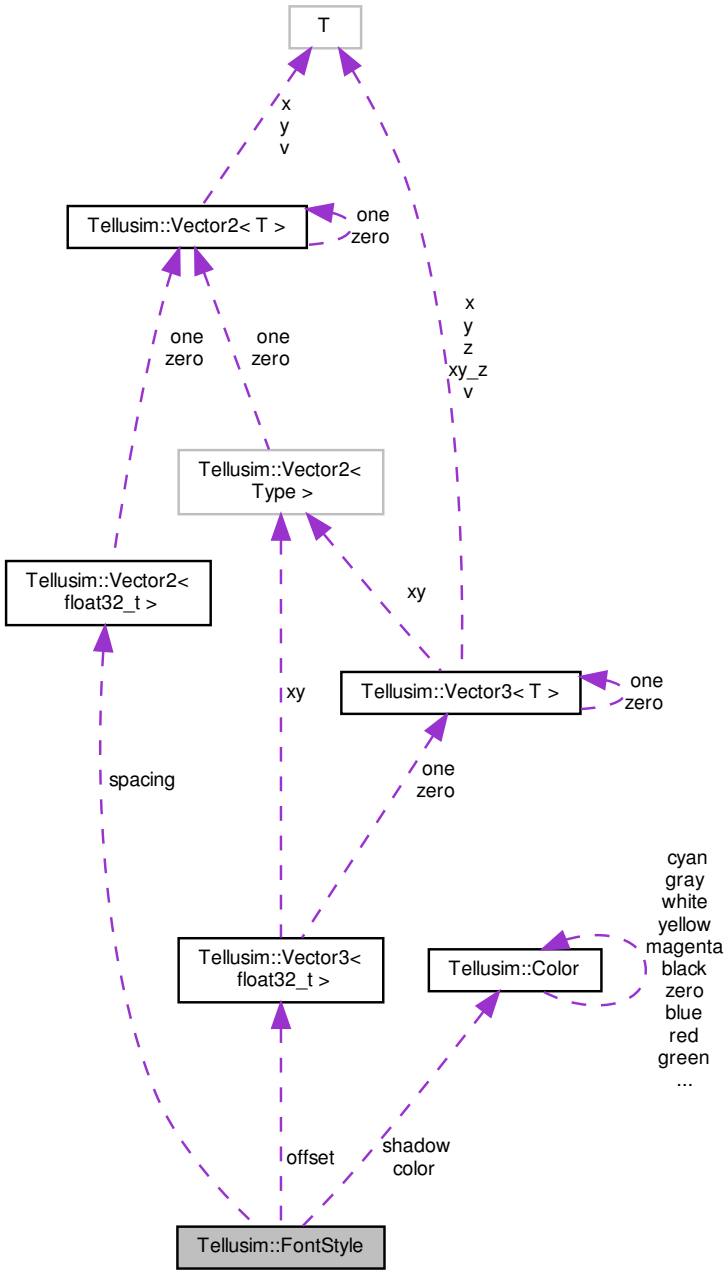
5.134.1 Detailed Description

The FontBatch32 struct represents a batch of UTF-32 encoded text to be rendered, with properties for positioning, color, and style. It includes constructors for setting the position and text, and optionally the font style.

5.135 Tellusim::FontStyle Struct Reference

```
#include <interface/TellusimTypes.h>
```

Collaboration diagram for Tellusim::FontStyle:



Public Member Functions

- **FontStyle** (uint32_t size)
- **FontStyle** (const Color &color)
- **FontStyle** (uint32_t size, const Color &color)

Public Attributes

- uint32_t **size** = 16
- uint32_t **scale** = 100
- bool **fixed** = false
- bool **kerning** = true
- Vector2f **spacing** = Vector2f::zero
- Color **color** = Color::white
- Vector3f **offset** = Vector3f::zero
- Color **shadow** = Color::black

5.135.1 Detailed Description

The FontStyle struct defines properties for configuring font appearance, including size, scale, and color, as well as settings for fixed-width fonts, kerning, spacing, and shadow effects.

5.136 Tellusim::FourierTransform Class Reference

`#include <parallel/TellusimFourierTransform.h>`

Public Types

- enum Mode {
  ModeRf16i = 0,
  ModeRf32i,
  ModeRGf16i,
  ModeRGf32i,
  ModeRGBf16c,
  ModeRGBf21c,
  ModeRGBf16p,
  ModeRGBf32p,
  NumModes }

  Transform modes.
- enum Flags {
  FlagNone = 0,
  FlagRf16i = (1 << ModeRf16i),
  FlagRf32i = (1 << ModeRf32i),
  FlagRGf16i = (1 << ModeRGf16i),
  FlagRGf32i = (1 << ModeRGf32i),
  FlagRGBf16c = (1 << ModeRGBf16c),
  FlagRGBf21c = (1 << ModeRGBf21c),
  FlagRGBf16p = (1 << ModeRGBf16p),
  FlagRGBf32p = (1 << ModeRGBf32p),
  FlagsInterleaved = (FlagRf16i | FlagRf32i | FlagRGf16i | FlagRGf32i),
  FlagsComplex = (FlagRGBf16c | FlagRGBf21c),
  FlagsPlanar = (FlagRGBf16p | FlagRGBf32p),
  FlagsAll = (FlagsInterleaved | FlagsComplex | FlagsPlanar) }

  Transform flags.
- enum Operation {
  ForwardCtoC = 0,
  BackwardCtoC,
  ForwardRtoC,
  BackwardCtoR,
  NumOperations }

  Transform operations.

Public Member Functions

- void clear ()

  clear transform
- bool isCreated (Mode mode) const

  check transform
- bool **isCreated** (Flags flags) const
- uint32_t getMaxWidth () const

  transform parameters
- uint32_t **getMaxHeight** () const
- bool create (const Device &device, Mode mode, uint32_t width, uint32_t height, Async *async=nullptr)

  create transform
- bool **create** (const Device &device, Flags flags, uint32_t width, uint32_t height, Async *async=nullptr)
- bool dispatch (Compute &compute, Mode mode, Operation op, Texture &dest, Texture &src, const Slice &dest_slice, const Slice &src_slice) const
- bool **dispatch** (Compute &compute, Mode mode, Operation op, Texture &dest, Texture &src, const Slice &src_slice) const
- bool **dispatch** (Compute &compute, Mode mode, Operation op, Texture &dest, Texture &src) const

5.136.1 Detailed Description

The FourierTransform class provides a highly flexible and efficient implementation of the Fourier Transform, supporting a variety of input/output data formats and operations. It is designed for tasks such as signal processing, image processing, and simulations requiring complex transformations. The class supports different modes, such as interleaved, complex, and planar formats, along with various transform operations (forward and backward, real-to-complex and complex-to-real). It allows for the creation and dispatch of transformations on textures, enabling GPU-accelerated processing for large datasets. Additionally, the class offers flexibility in handling different data types and sizes, ensuring scalability across a range of applications.

5.136.2 Member Function Documentation

5.136.2.1 dispatch()

```
bool Tellusim::FourierTransform::dispatch (
            Compute & compute,
            Mode mode,
            Operation op,
            Texture & dest,
            Texture & src,
            const Slice & dest_slice,
            const Slice & src_slice ) const
```

dispatch transform

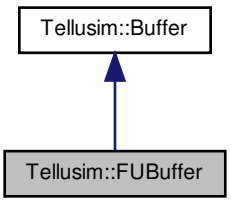
Parameters

mode	Transform mode.
op	Transform operation.
dest	Destination texture.
src	Source texture.

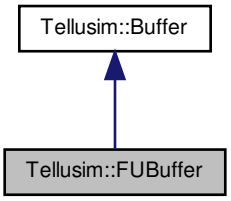
5.137 Tellusim::FUBuffer Class Reference

`#include <platform/TellusimBuffer.h>`

Inheritance diagram for Tellusim::FUBuffer:



Collaboration diagram for Tellusim::FUBuffer:



Public Member Functions

- **FUBuffer** (const Array< Buffer > &buffers, bool owner=false)
- void setMask (uint32_t mask)

  device mask
- uint32_t **getMask** () const
- uint32_t getNumBuffers () const

  Fusion buffers.
- const Buffer **getBuffer** (uint32_t index) const
- Buffer **getBuffer** (uint32_t index)

Additional Inherited Members

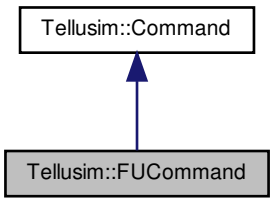
5.137.1 Detailed Description

The FUBuffer class extends the Buffer class to manage a fusion of multiple buffers from different devices. It provides methods to set and retrieve a device mask, track the number of fusion buffers, and access individual buffers within the fusion. This functionality is particularly useful for high-performance computing scenarios that require managing and interacting with buffers from multiple devices in parallel workflows.

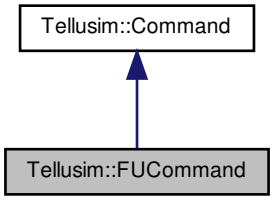
5.138 Tellusim::FUCommand Class Reference

`#include <platform/TellusimCommand.h>`

Inheritance diagram for Tellusim::FUCommand:



Collaboration diagram for Tellusim::FUCommand:



Public Member Functions

- **FUCommand** (const Array< Command > &commands, bool owner=false)
- void setMask (uint32_t mask)

  device mask
- uint32_t **getMask** () const
- uint32_t getNumCommands () const

  Fusion commands.
- const Command **getCommand** (uint32_t index) const
- Command **getCommand** (uint32_t index)

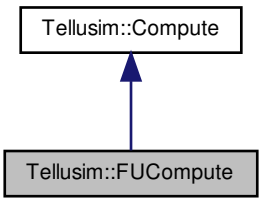
5.138.1 Detailed Description

The FUCommand class extends the Command class to execute operations involving a fusion of multiple devices. It provides methods to set and retrieve a device mask, track the number of fusion commands, and access individual commands within the fusion. This functionality is particularly useful for high-performance computing scenarios that require managing and interacting with buffers from multiple devices in parallel workflows.

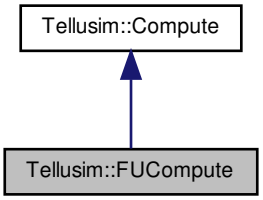
5.139 Tellusim::FUCompute Class Reference

`#include <platform/TellusimCompute.h>`

Inheritance diagram for Tellusim::FUCompute:



Collaboration diagram for Tellusim::FUCompute:



Public Member Functions

- **FUCompute** (const Array< Compute > &computes, bool owner=false)
- void setMask (uint32_t mask)

  device mask
- uint32_t **getMask** () const
- uint32_t getNumComputes () const

  Fusion computes.
- const Compute **getCompute** (uint32_t index) const
- Compute **getCompute** (uint32_t index)

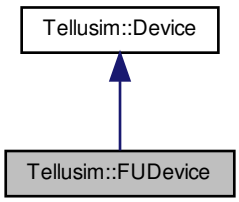
5.139.1 Detailed Description

The FUCompute class extends the Compute class to execute operations involving a fusion of multiple devices. It provides methods to set and retrieve a device mask, track the number of fusion computes, and access individual computes within the fusion. This functionality is particularly useful for high-performance computing scenarios that require managing and interacting with buffers from multiple devices in parallel workflows.

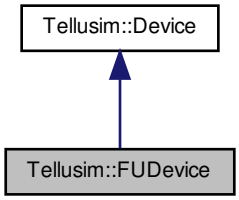
5.140 Tellusim::FUDevice Class Reference

`#include <platform/TellusimDevice.h>`

Inheritance diagram for Tellusim::FUDevice:



Collaboration diagram for Tellusim::FUDevice:



Public Member Functions

- **FUDevice** (const Array< Device > &devices, bool owner=false)
- void setMask (uint32_t mask)

  device mask
- uint32_t **getMask** () const
- uint32_t getNumDevices () const

  Fusion devices.
- const Device **getDevice** (uint32_t index) const
- Device **getDevice** (uint32_t index)

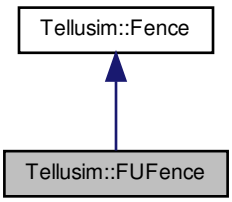
5.140.1 Detailed Description

The FUDevice class extends the Device class to manage a fusion of multiple devices from different platforms. It provides methods to set and retrieve a device mask, track the number of fusion devices, and access individual devices within the fusion. This functionality is particularly useful for high-performance computing scenarios that require managing and interacting with multiple devices in parallel workflows.

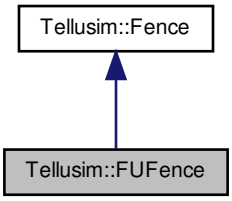
5.141 Tellusim::FUFence Class Reference

`#include <platform/TellusimFence.h>`

Inheritance diagram for Tellusim::FUFence:



Collaboration diagram for Tellusim::FUFence:



Public Member Functions

- **FUFence** (const Array< Fence > &fences, bool owner=false)
- void setMask (uint32_t mask)

  device mask
- uint32_t **getMask** () const
- uint32_t getNumFences () const

  Fusion fences.
- const Fence **getFence** (uint32_t index) const
- Fence **getFence** (uint32_t index)

Additional Inherited Members

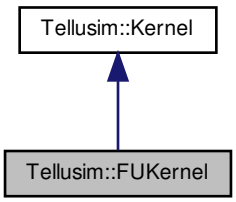
5.141.1 Detailed Description

The FUFence class extends the Fence class to manage a fusion of multiple fences from different devices. It provides methods to set and retrieve a device mask, track the number of fusion fences, and access individual fences within the fusion. This functionality is particularly useful for high-performance computing scenarios where synchronization across multiple devices is required.

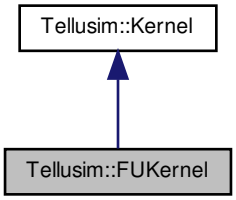
5.142 Tellusim::FUKernel Class Reference

`#include <platform/TellusimKernel.h>`

Inheritance diagram for Tellusim::FUKernel:



Collaboration diagram for Tellusim::FUKernel:



Public Member Functions

- **FUKernel** (const Array< Kernel > &kernels, bool owner=false)
- void setMask (uint32_t mask)

  device mask
- uint32_t **getMask** () const
- uint32_t getNumKernels () const

  Fusion kernels.
- const Kernel **getKernel** (uint32_t index) const
- Kernel **getKernel** (uint32_t index)

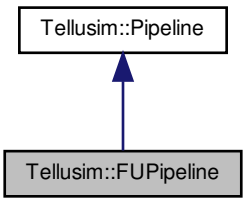
5.142.1 Detailed Description

The FUKernel class extends the Kernel class to manage a fusion of multiple compute kernels across different devices. It provides methods to set and retrieve a device mask, track the number of fused kernels, and access individual kernels within the fusion. This design is useful for high-performance computing workflows that require coordinated execution of compute operations across multiple devices.

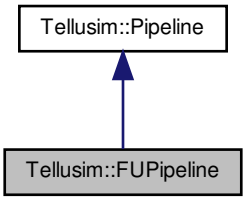
5.143 Tellusim::FUPipeline Class Reference

`#include <platform/TellusimPipeline.h>`

Inheritance diagram for Tellusim::FUPipeline:



Collaboration diagram for Tellusim::FUPipeline:



Public Member Functions

- **FUPipeline** (const Array< Pipeline > &pipelines, bool owner=false)
- void setMask (uint32_t mask)

  device mask
- uint32_t **getMask** () const
- uint32_t getNumPipelines () const

  Fusion pipelines.
- const Pipeline **getPipeline** (uint32_t index) const
- Pipeline **getPipeline** (uint32_t index)

Additional Inherited Members

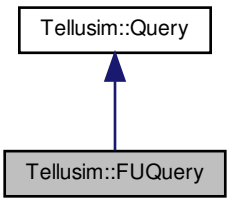
5.143.1 Detailed Description

The FUPipeline class extends the Pipeline class to manage a fusion of multiple compute pipelines across different devices. It provides methods to set and retrieve a device mask, track the number of fused pipelines, and access individual pipelines within the fusion. This design is useful for high-performance computing workflows that require coordinated execution of rendering operations across multiple devices.

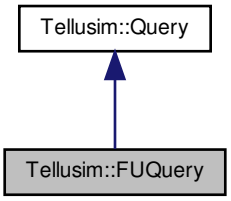
5.144 Tellusim::FUQuery Class Reference

`#include <platform/TellusimQuery.h>`

Inheritance diagram for Tellusim::FUQuery:



Collaboration diagram for Tellusim::FUQuery:



Public Member Functions

- **FUQuery** (const Array< Query > &queries, bool owner=false)
- void setMask (uint32_t mask)

  device mask
- uint32_t **getMask** () const
- uint32_t getNumQueries () const

  Fusion queries.
- const Query **getQuery** (uint32_t index) const
- Query **getQuery** (uint32_t index)

Additional Inherited Members

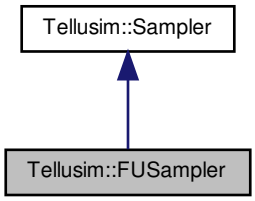
5.144.1 Detailed Description

The FUQuery class extends the Query class to manage a fusion of multiple queries from different devices. It provides methods to set and retrieve a device mask, track the number of fusion queries, and access individual queries within the fusion. This functionality is particularly useful for high-performance computing scenarios, where managing and interacting with queries across multiple devices in parallel workflows is required.

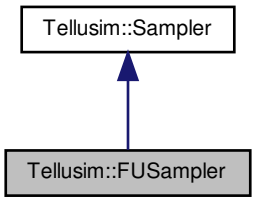
5.145 Tellusim::FUSampler Class Reference

`#include <platform/TellusimSampler.h>`

Inheritance diagram for Tellusim::FUSampler:



Collaboration diagram for Tellusim::FUSampler:



Public Member Functions

- **FUSampler** (const Array< Sampler > &samplers, bool owner=false)
- void setMask (uint32_t mask)

  device mask
- uint32_t **getMask** () const
- uint32_t getNumSamplers () const

  Fusion samplers.
- const Sampler **getSampler** (uint32_t index) const
- Sampler **getSampler** (uint32_t index)

Additional Inherited Members

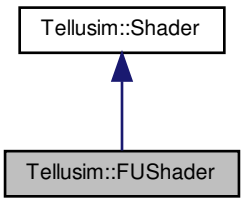
5.145.1 Detailed Description

The FUSampler class extends the Sampler class to handle fusion samplers across multiple devices. It provides functionality for managing a device mask and accessing a collection of fusion samplers, making it useful in high-performance computing and multi-device workflows. The class allows querying the number of fusion samplers and provides methods to retrieve individual samplers from the fusion, enabling flexible sampling across different devices.

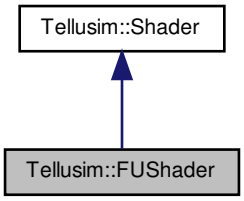
5.146 Tellusim::FUShader Class Reference

`#include <platform/TellusimShader.h>`

Inheritance diagram for Tellusim::FUShader:



Collaboration diagram for Tellusim::FUShader:



Public Member Functions

- **FUShader** (const Array< Shader > &shaders, bool owner=false)
- void setMask (uint32_t mask)

  device mask
- uint32_t **getMask** () const
- uint32_t getNumShaders () const

  Fusion shaders.
- const Shader **getShader** (uint32_t index) const
- Shader **getShader** (uint32_t index)

Additional Inherited Members

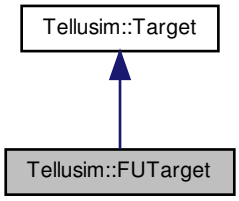
5.146.1 Detailed Description

The FUShader class extends the Shader class to manage a fusion of multiple shaders across different devices. It provides methods to set and retrieve a device mask, track the number of fusion shaders, and access individual shaders within the fusion. This functionality is particularly useful for high-performance computing scenarios that require managing and interacting with shaders from multiple devices in parallel workflows.

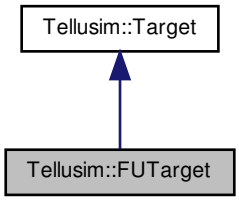
5.147 Tellusim::FUTarget Class Reference

`#include <platform/TellusimTarget.h>`

Inheritance diagram for Tellusim::FUTarget:



Collaboration diagram for Tellusim::FUTarget:



Public Member Functions

- **FUTarget** (const Array< Target > &targets, bool owner=false)
- void setMask (uint32_t mask)

  device mask
- uint32_t **getMask** () const
- uint32_t getNumTargets () const

  Fusion targets.
- const Target **getTarget** (uint32_t index) const
- Target **getTarget** (uint32_t index)

Additional Inherited Members

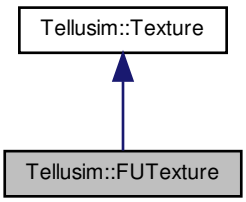
5.147.1 Detailed Description

The FUTarget class extends the Target class to support managing multiple targets across various devices. It provides functionality for handling a fusion of targets from different devices, which is useful for multi-device workflows.

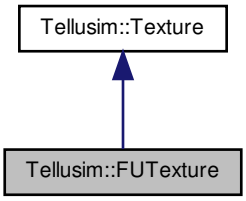
5.148 Tellusim::FUTexture Class Reference

`#include <platform/TellusimTexture.h>`

Inheritance diagram for Tellusim::FUTexture:



Collaboration diagram for Tellusim::FUTexture:



Public Member Functions

- **FUTexture** (const Array< Texture > &textures, bool owner=false)
- void setMask (uint32_t mask)

  device mask
- uint32_t **getMask** () const
- uint32_t getNumTextures () const

  Fusion textures.
- const Texture **getTexture** (uint32_t index) const
- Texture **getTexture** (uint32_t index)

Additional Inherited Members

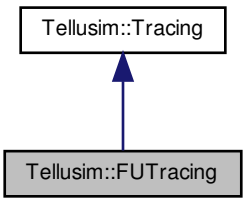
5.148.1 Detailed Description

The FUTexture class extends the Texture class to manage a fusion of multiple textures from different devices. It provides methods for setting and retrieving a device mask, tracking the number of fusion textures, and accessing individual textures within the fusion. This functionality is essential for managing textures across multiple devices, especially in high-performance or multi-device workflows where efficient resource handling is required.

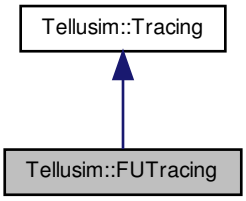
5.149 Tellusim::FUTracing Class Reference

`#include <platform/TellusimTracing.h>`

Inheritance diagram for Tellusim::FUTracing:



Collaboration diagram for Tellusim::FUTracing:



Public Member Functions

- **FUTracing** (const Array< Tracing > &tracings, bool owner=false)
- void setMask (uint32_t mask)

  device mask
- uint32_t **getMask** () const
- uint32_t getNumTracings () const

  Fusion tracings.
- const Tracing **getTracing** (uint32_t index) const
- Tracing **getTracing** (uint32_t index)

Additional Inherited Members

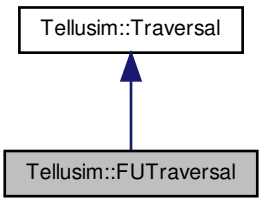
5.149.1 Detailed Description

The FUTracing class extends the Tracing class to manage a fusion of multiple tracing instances from different devices. It provides methods to set and retrieve a device mask, track the number of fusion tracings, and access individual tracings within the fusion. This functionality is particularly useful for high-performance computing scenarios that require managing and interacting with buffers from multiple devices in parallel workflows.

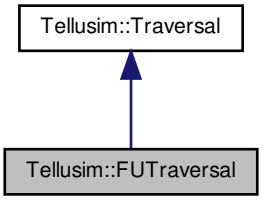
5.150 Tellusim::FUTraversal Class Reference

`#include <platform/TellusimTraversal.h>`

Inheritance diagram for Tellusim::FUTraversal:



Collaboration diagram for Tellusim::FUTraversal:



Public Member Functions

- **FUTraversal** (const Array< Traversal > &traversals, bool owner=false)
- void setMask (uint32_t mask)

  device mask
- uint32_t **getMask** () const
- uint32_t getNumTraversals () const

  Fusion traversals.
- const Traversal **getTraversal** (uint32_t index) const
- Traversal **getTraversal** (uint32_t index)

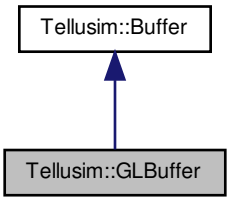
5.150.1 Detailed Description

The FUTraversal class extends the Traversal class to manage a fusion of multiple ray-tracing pipelines across different devices. It provides methods to set and retrieve a device mask, track the number of fused traversals, and access individual traversals within the fusion. This design is particularly beneficial for high-performance computing scenarios that require synchronized execution of ray-tracing operations across multiple devices.

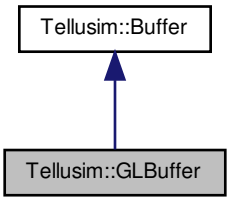
5.151 Tellusim::GLBuffer Class Reference

`#include <platform/TellusimBuffer.h>`

Inheritance diagram for Tellusim::GLBuffer:



Collaboration diagram for Tellusim::GLBuffer:



Public Member Functions

- bool create (Flags flags, uint32_t target, uint32_t buffer_id)
  create external buffer
- uint32_t **getTarget** () const
- uint32_t **getBufferID** () const

Additional Inherited Members

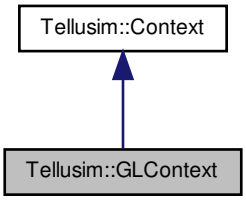
5.151.1 Detailed Description

The GLBuffer class is an OpenGL-specific implementation of the Buffer class, providing access to OpenGL buffer resources. It enables the creation of external buffers by specifying the buffer target and ID, allowing for efficient interaction with OpenGL buffer management system. This class provides functions to retrieve the buffer target and ID while inheriting the create method from the Buffer class for initializing buffers in OpenGL-based applications.

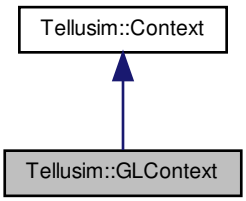
5.152 Tellusim::GLContext Class Reference

`#include <platform/TellusimContext.h>`

Inheritance diagram for Tellusim::GLContext:



Collaboration diagram for Tellusim::GLContext:



Public Member Functions

- bool create (void *context)
  create context
- void * getGLDisplay () const
  current context
- void * **getGLVisual** () const
- void * **getGLConfig** () const
- void * **getGLSurface** () const
- void * **getGLContext** () const

Static Public Member Functions

- static void * getProcAddress (const char *name)
  OpenGL functions.
- static bool error (uint32_t result)
  check OpenGL errors
- static bool **check** ()

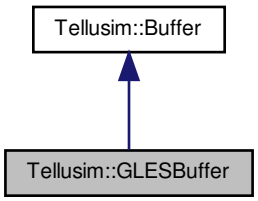
5.152.1 Detailed Description

The GLContext class is an OpenGL-specific implementation of the Context class. It initializes the rendering context using an externally provided OpenGL context. The class provides access to OpenGL context-related resources, including the display, visual, configuration, surface, and the context.

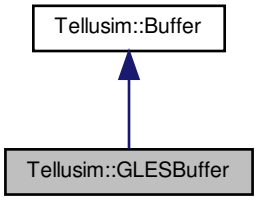
5.153 Tellusim::GLESBuffer Class Reference

```
#include <platform/TellusimBuffer.h>
```

Inheritance diagram for Tellusim::GLESBuffer:



Collaboration diagram for Tellusim::GLESBuffer:



Public Member Functions

- bool create (Flags flags, uint32_t target, uint32_t buffer_id)

  create external buffer
- uint32_t **getTarget** () const
- uint32_t **getBufferID** () const

Additional Inherited Members

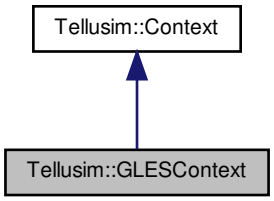
5.153.1 Detailed Description

The GLESBuffer class is an OpenGLES-specific implementation of the Buffer class, providing access to OpenGLES buffer resources. It enables the creation of external buffers by specifying the buffer target and ID, allowing for efficient interaction with OpenGLES buffer management system. This class provides functions to retrieve the buffer target and ID while inheriting the create method from the Buffer class for initializing buffers in OpenGLES-based applications.

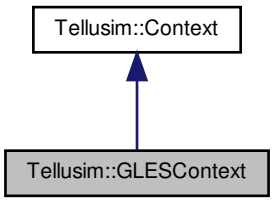
5.154 Tellusim::GLESContext Class Reference

`#include <platform/TellusimContext.h>`

Inheritance diagram for Tellusim::GLESContext:



Collaboration diagram for Tellusim::GLESContext:



Public Member Functions

- bool create (void *context)
  create context
- void * getGLESDisplay () const
  current context
- void * **getGLESConfig** () const
- void * **getGLESContext** () const

Static Public Member Functions

- static void * getProcAddress (const char *name)

  OpenGLES functions.
- static bool error (uint32_t result)

  check OpenGLES errors
- static bool **check** ()

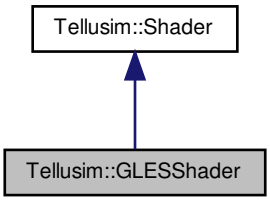
5.154.1 Detailed Description

The GLESContext class is an OpenGLES-specific implementation of the Context class. It initializes the rendering context using an externally provided OpenGLES context. The class provides access to OpenGLES context-related resources, including the display, configuration, and the context.

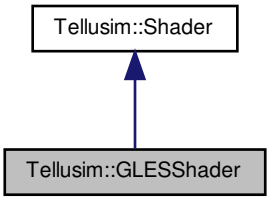
5.155 Tellusim::GLESShader Class Reference

```
#include <platform/TellusimShader.h>
```

Inheritance diagram for Tellusim::GLESShader:



Collaboration diagram for Tellusim::GLESShader:



Public Member Functions

- bool **attachShader** (uint32_t program_id)
- uint32_t **getShaderType** () const
- uint32_t **getShaderID** () const

Additional Inherited Members

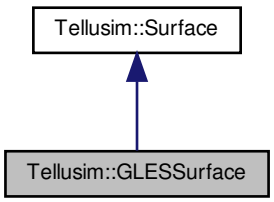
5.155.1 Detailed Description

The GLESShader class extends the Shader class to specialize in managing shaders for OpenGLES. It provides methods to attach the shader to a program, retrieve the shader type, and get the shader ID for integration with OpenGLES.

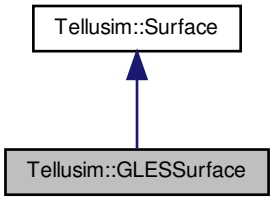
5.156 Tellusim::GLESSurface Class Reference

```
#include <platform/TellusimSurface.h>
```

Inheritance diagram for Tellusim::GLESSurface:



Collaboration diagram for Tellusim::GLESSurface:



Public Member Functions

- **GLESSurface** (GLESContext &context)
- void * getContext () const

  current context
- void setColorTextureID (uint32_t texture_id)

  texture identifiers
- void **setDepthTextureID** (uint32_t texture_id)
- uint32_t **getColorTextureID** () const
- uint32_t **getDepthTextureID** () const
- void setFramebufferID (uint32_t framebuffer_id)

  framebuffer identifier
- uint32_t **getFramebufferID** () const
- uint32_t getColorInternalFormat () const

  surface formats
- uint32_t **getDepthInternalFormat** () const

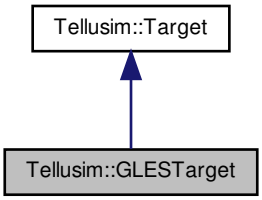
5.156.1 Detailed Description

The GLESSurface class extends the Surface class to provide OpenGLES-specific functionality for managing a rendering surface. It includes methods for interacting with OpenGLES contexts and managing texture identifiers and framebuffers, enabling rendering operations in the context of OpenGLES. The class supports managing color and depth textures, along with their respective internal formats, and facilitates handling of framebuffer objects essential for rendering.

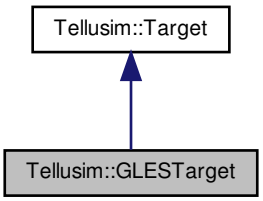
5.157 Tellusim::GLESTarget Class Reference

```
#include <platform/TellusimTarget.h>
```

Inheritance diagram for Tellusim::GLESTarget:



Collaboration diagram for Tellusim::GLESTarget:



Public Member Functions

- uint32_t **getFramebufferID** () const

Additional Inherited Members

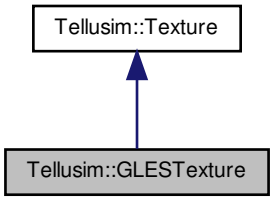
5.157.1 Detailed Description

The GLESTarget class is an OpenGLES-specific implementation of the Target class, designed to manage OpenGLES framebuffers. It provides access to the framebuffer ID, allowing for efficient handling of rendering operations within OpenGLES.

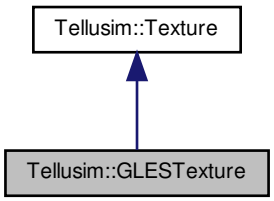
5.158 Tellusim::GLESTexture Class Reference

```
#include <platform/TellusimTexture.h>
```

Inheritance diagram for Tellusim::GLESTexture:



Collaboration diagram for Tellusim::GLESTexture:



Public Member Functions

- bool create (uint32_t target, uint32_t texture_id, Flags flags=DefaultFlags, Format format=FormatUnknown)
  create external texture
- uint32_t **getTarget** () const
- uint32_t **getInternalFormat** () const
- uint32_t **getTextureID** () const

Additional Inherited Members

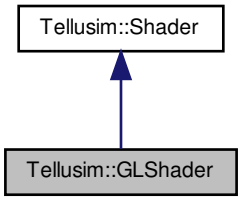
5.158.1 Detailed Description

The GLESTexture class is an OpenGLES-specific implementation of the Texture class, providing access to OpenGLES texture resources. It enables the creation of external textures by specifying the texture target, ID, and additional flags, allowing for efficient interaction with OpenGLES texture management system. This class provides functions to retrieve the texture target, internal format, and ID while inheriting the create method from the Texture class for initializing textures in OpenGLES-based applications.

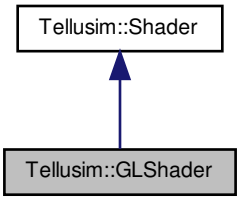
5.159 Tellusim::GLShader Class Reference

```
#include <platform/TellusimShader.h>
```

Inheritance diagram for Tellusim::GLShader:



Collaboration diagram for Tellusim::GLShader:



Public Member Functions

- bool **attachShader** (uint32_t program_id)
- uint32_t **getShaderType** () const
- uint32_t **getShaderID** () const

Additional Inherited Members

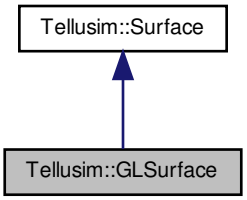
5.159.1 Detailed Description

The GLShader class extends the Shader class to specialize in managing shaders for OpenGL. It provides methods to attach the shader to a program, retrieve the shader type, and get the shader ID for integration with OpenGL.

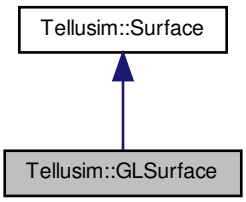
5.160 Tellusim::GLSurface Class Reference

```
#include <platform/TellusimSurface.h>
```

Inheritance diagram for Tellusim::GLSurface:



Collaboration diagram for Tellusim::GLSurface:



Public Member Functions

- **GLSurface** (GLContext &context)
- void * getContext () const

  current context
- void setColorTextureID (uint32_t texture_id)

  texture identifiers
- void **setDepthTextureID** (uint32_t texture_id)
- uint32_t **getColorTextureID** () const
- uint32_t **getDepthTextureID** () const
- void setFramebufferID (uint32_t framebuffer_id)

  framebuffer identifier
- uint32_t **getFramebufferID** () const
- uint32_t getColorInternalFormat () const

  surface formats
- uint32_t **getDepthInternalFormat** () const

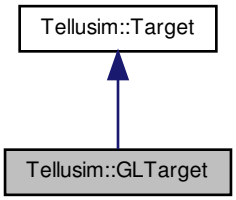
5.160.1 Detailed Description

The GLSurface class extends the Surface class to provide OpenGL-specific functionality for managing a rendering surface. It includes methods for interacting with OpenGL contexts and managing texture identifiers and framebuffers, enabling rendering operations in the context of OpenGL. The class supports managing color and depth textures, along with their respective internal formats, and facilitates handling of framebuffer objects essential for rendering.

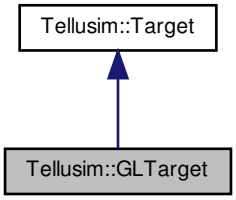
5.161 Tellusim::GLTarget Class Reference

```
#include <platform/TellusimTarget.h>
```

Inheritance diagram for Tellusim::GLTarget:



Collaboration diagram for Tellusim::GLTarget:



Public Member Functions

- uint32_t **getFramebufferID** () const

Additional Inherited Members

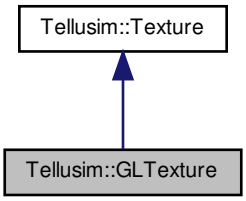
5.161.1 Detailed Description

The GLTarget class is an OpenGL-specific implementation of the Target class, designed to manage OpenGL framebuffers. It provides access to the framebuffer ID, allowing for efficient handling of rendering operations within OpenGL.

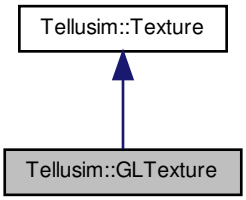
5.162 Tellusim::GLTexture Class Reference

```
#include <platform/TellusimTexture.h>
```

Inheritance diagram for Tellusim::GLTexture:



Collaboration diagram for Tellusim::GLTexture:



Public Member Functions

- bool create (uint32_t target, uint32_t texture_id, Flags flags=DefaultFlags, Format format=FormatUnknown)
  create external texture
- uint32_t **getTarget** () const
- uint32_t **getInternalFormat** () const
- uint32_t **getTextureID** () const

Additional Inherited Members

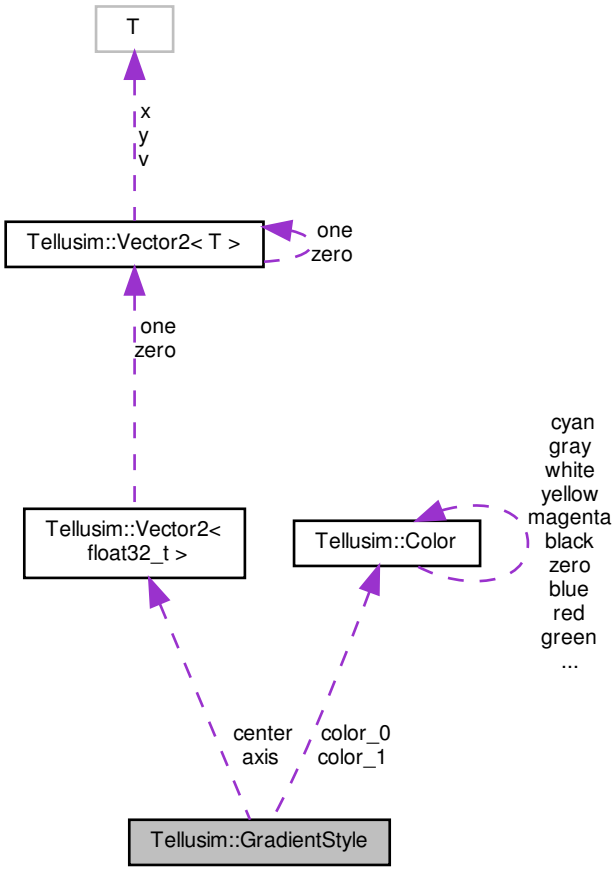
5.162.1 Detailed Description

The GLTexture class is an OpenGL-specific implementation of the Texture class, providing access to OpenGL texture resources. It enables the creation of external textures by specifying the texture target, ID, and additional flags, allowing for efficient interaction with OpenGL texture management system. This class provides functions to retrieve the texture target, internal format, and ID while inheriting the create method from the Texture class for initializing textures in OpenGL-based applications.

5.163 Tellusim::GradientStyle Struct Reference

`#include <interface/TellusimTypes.h>`

Collaboration diagram for Tellusim::GradientStyle:



Public Member Functions

- **GradientStyle** (const Color &c0, const Color &c1)
- **GradientStyle** (float32_t radius, const Vector2f &center)
- **GradientStyle** (float32_t length, const Vector2f &center, const Vector2f &axis)
- **GradientStyle** (float32_t radius, const Vector2f &center, const Color &c0, const Color &c1)
- **GradientStyle** (float32_t length, const Vector2f &center, const Vector2f &axis, const Color &c0, const Color &c1)
- bool isValid () const

  check style
- **operator bool** () const

Public Attributes

- float32_t **radius** = 0.0f
- float32_t **length** = 0.0f
- Vector2f **center** = Vector2f::zero
- Vector2f **axis** = Vector2f::zero
- Color **color_0** = Color::white
- Color **color_1** = Color::black

5.163.1 Detailed Description

The GradientStyle struct defines the properties for a gradient, including the radius, length, center, axis, and two colors that define the gradient appearance.

5.164 Tellusim::HeapPool< Threshold > Class Template Reference

`#include <core/TellusimPool.h>`

Public Member Functions

- HeapPool ()
  constructor
- **HeapPool** (uint32_t size)
- void init (uint32_t size)
  initialize pool
- void shutdown ()
  shutdown pool
- bool isInitialized ()
  pool status
- void append (uint32_t size)
  append memory
- uint32_t allocate (uint32_t alignment, uint32_t size, bool optimal=true)
  allocate memory
- void free (uint32_t offset)
  free memory
- uint32_t getSize () const
  pool size in bytes
- uint32_t getMemory () const
  memory usage in bytes
- uint32_t getAllocations () const
  number of allocations
- uint32_t getBlocks () const
  heap fragmentation
- const Map< uint32_t, uint32_t > & getState () const
  heap state

5.164.1 Detailed Description

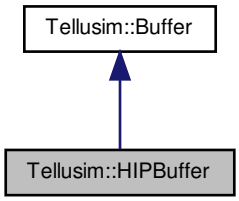
template<uint32_t Threshold = 8>
class Tellusim::HeapPool< Threshold >

Heap pool memory allocator

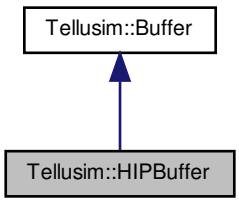
5.165 Tellusim::HIPBuffer Class Reference

`#include <platform/TellusimBuffer.h>`

Inheritance diagram for Tellusim::HIPBuffer:



Collaboration diagram for Tellusim::HIPBuffer:



Public Member Functions

- void * **getBufferPtr** () const
- uint8_t * **getBufferData** () const
- void * **getBufferEvent** () const
- uint32_t **getArrayFormat** () const
- uint32_t **getArrayChannels** () const
- void * **getSharedMemory** () const

Additional Inherited Members

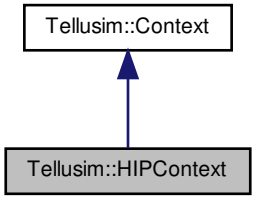
5.165.1 Detailed Description

The HIPBuffer class is a HIP-specific implementation of the Buffer class, providing access to HIP buffer resources and memory management. It allows retrieval of buffer pointers, data, events, and details about array format and channels. Additionally, the class supports interaction with shared memory, enabling efficient resource management and interoperation in HIP-based applications.

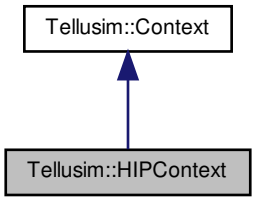
5.166 Tellusim::HIPContext Class Reference

`#include <platform/TellusimContext.h>`

Inheritance diagram for Tellusim::HIPContext:



Collaboration diagram for Tellusim::HIPContext:



Public Member Functions

- int32_t getDevice () const
  current device
- void * **getHIPContext** () const
- void * **getStream** () const

Static Public Member Functions

- static void * getProcAddress (const char *name)

  Hip functions.
- static bool error (uint32_t result)

  check Hip errors

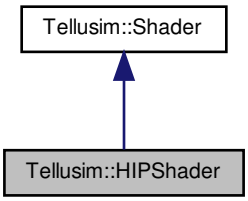
5.166.1 Detailed Description

The HIPContext class is a HIP-specific implementation of the Context class. It provides functionality to manage and interact with a HIP context, stream, and device. The class allows retrieving the current CUDA device, context, and stream.

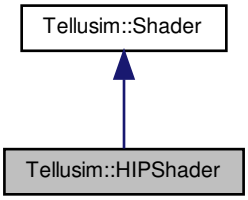
5.167 Tellusim::HIPShader Class Reference

`#include <platform/TellusimShader.h>`

Inheritance diagram for Tellusim::HIPShader:



Collaboration diagram for Tellusim::HIPShader:



Public Member Functions

- void * **getModule** () const
- void * **getFunction** () const

Additional Inherited Members

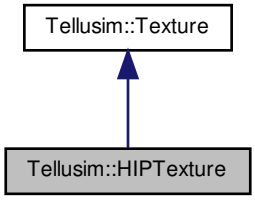
5.167.1 Detailed Description

The HIPShader class extends the Shader class to specialize in managing shaders for HIP. It provides methods to retrieve the underlying HIP module and function, enabling integration with HIP.

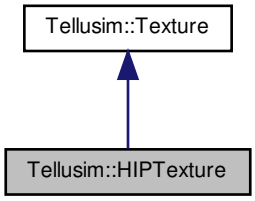
5.168 Tellusim::HIPTexture Class Reference

```
#include <platform/TellusimTexture.h>
```

Inheritance diagram for Tellusim::HIPTexture:



Collaboration diagram for Tellusim::HIPTexture:



Public Member Functions

- void * **getTextureArray** () const
- void * **getTextureLevel** (uint32_t index) const
- const void * **getChannelFormat** () const

Additional Inherited Members

5.168.1 Detailed Description

The HIPTexture class is a HIP-specific implementation of the Texture class, providing access to HIP texture resources and management. It allows retrieval of texture arrays, specific texture levels, and channel format details. This class is designed to facilitate efficient texture handling and interaction with HIP-based applications, supporting advanced texture operations and resource management.

5.169 Tellusim::Image Class Reference

`#include <format/TellusimImage.h>`

Public Types

- enum Flags {
  FlagNone = 0,
  FlagMipmaps = (1 << 0),
  FlagNoClear = (1 << 1),
  FlagNoAllocate = (1 << 2),
  FlagFast = (1 << 3),
  FlagBest = (1 << 4),
  FlagPerceptual = (1 << 5),
  FlagPanorama = (1 << 6),
  FlagNormalize = (1 << 7),
  FlagGamma = (1 << 8),
  FlagSRGB = (1 << 9),
  NumFlags = 10 }

  Image flags.
- enum Filter {
  FilterUnknown = 0,
  FilterPoint,
  FilterLinear,
  FilterCubic,
  FilterSinc,
  FilterBox,
  FilterMax,
  FilterMin,
  FilterMip,
  FilterCR,
  NumFilters }

  Image filters.

Public Member Functions

- **Image** (const char *name, Flags flags=FlagNone, uint32_t offset=0)
- **Image** (Stream &stream, Flags flags=FlagNone, uint32_t offset=0)
- **Image** (Type type, Format format, const Size &size, Flags flags=FlagNone)
- **Image** (Type type, Format format, const Size &size, uint32_t layers, Flags flags=FlagNone)
- void clear ()

  clear image
- bool isLoaded () const

check image

- bool **isAllocated** () const
- Type getType () const

image type

- const char * **getTypeName** () const
- bool **is2DType** () const
- bool **is3DType** () const
- bool **isCubeType** () const
- Format getFormat () const

image format

- const char * **getFormatName** () const
- bool **isColorFormat** () const
- bool **isDepthFormat** () const
- bool **isPixelFormat** () const
- bool **isPlainFormat** () const
- bool **isMixedFormat** () const
- bool **isBlockFormat** () const
- bool **isStencilFormat** () const
- bool **isNormFormat** () const
- bool **isSRGBFormat** () const
- bool **isFloatFormat** () const
- bool **isSignedFormat** () const
- bool **isUnsignedFormat** () const
- bool **isIntegerFormat** () const
- bool **isi8Format** () const
- bool **isu8Format** () const
- bool **is8BitFormat** () const
- bool **isi16Format** () const
- bool **isu16Format** () const
- bool **isf16Format** () const
- bool **is16BitFormat** () const
- bool **isi32Format** () const
- bool **isu32Format** () const
- bool **isf32Format** () const
- bool **is32BitFormat** () const
- bool **isi64Format** () const
- bool **isu64Format** () const
- bool **isf64Format** () const
- bool **is64BitFormat** () const
- bool **isBC15Format** () const
- bool **isBC67Format** () const
- bool **isETC2Format** () const
- bool **isASTCFormat** () const
- uint32_t **getComponents** () const
- uint32_t **getPixelSize** () const
- uint32_t **getBlockSize** () const
- uint32_t **getBlockWidth** () const
- uint32_t **getBlockHeight** () const
- uint32_t getWidth () const

image dimension

- uint32_t **getHeight** () const
- uint32_t **getDepth** () const
- uint32_t **getFaces** () const
- uint32_t **getLayers** () const

- uint32_t **getMipmaps** () const
- uint32_t **findMipmap** (const Size &size) const
- uint32_t **getWidth** (uint32_t mipmap) const
- uint32_t **getHeight** (uint32_t mipmap) const
- uint32_t **getDepth** (uint32_t mipmap) const
- bool **hasLayers** () const
- bool **hasMipmaps** () const
- Size **getSize** () const
- Region **getRegion** () const
- Slice **getSlice** () const
- Size **getSize** (uint32_t mipmap) const
- Region **getRegion** (uint32_t mipmap) const
- Slice **getSlice** (uint32_t mipmap) const
- void setMetaInfo (const String &str)

  image meta info
- String **getMetaInfo** () const
- String getDescription () const

  image description
- size_t getOffset (const Slice &slice, uint32_t alignment=1) const

  image layout
- size_t **getStride** (uint32_t mipmap=0, uint32_t alignment=1) const
- size_t **getMipmapSize** (uint32_t mipmap, uint32_t alignment=1) const
- size_t **getLayerSize** (uint32_t alignment=1) const
- size_t **getDataSize** (uint32_t alignment=1) const
- bool create (Type type, Format format, const Size &size, Flags flags=FlagNone)

  create image
- bool **create** (Type type, Format format, const Size &size, uint32_t layers, Flags flags=FlagNone)
- bool **create2D** (Format format, uint32_t size, Flags flags=FlagNone)
- bool **create3D** (Format format, uint32_t size, Flags flags=FlagNone)
- bool **createCube** (Format format, uint32_t size, Flags flags=FlagNone)
- bool **create2D** (Format format, uint32_t width, uint32_t height, Flags flags=FlagNone)
- bool **create3D** (Format format, uint32_t width, uint32_t height, uint32_t depth, Flags flags=FlagNone)
- bool **create2D** (Format format, uint32_t width, uint32_t height, uint32_t layers, Flags flags=FlagNone)
- bool **createCube** (Format format, uint32_t size, uint32_t layers, Flags flags=FlagNone)
- bool info (const char *name, Flags flags=FlagNone, uint32_t offset=0, Async *async=nullptr)

  info image
- bool **info** (const String &name, Flags flags=FlagNone, uint32_t offset=0, Async *async=nullptr)
- bool **info** (Stream &stream, Flags flags=FlagNone, uint32_t offset=0, Async *async=nullptr)
- bool **info** (const char *name, Async *async)
- bool **info** (const String &name, Async *async)
- bool **info** (Stream &stream, Async *async)
- bool load (const char *name, Flags flags=FlagNone, uint32_t offset=0, Async *async=nullptr)

  load image
- bool **load** (const String &name, Flags flags=FlagNone, uint32_t offset=0, Async *async=nullptr)
- bool **load** (Stream &stream, Flags flags=FlagNone, uint32_t offset=0, Async *async=nullptr)
- bool **load** (const char *name, Async *async)
- bool **load** (const String &name, Async *async)
- bool **load** (Stream &stream, Async *async)
- bool save (const char *name, Flags flags=FlagNone, uint32_t quality=95) const

  save image
- bool **save** (const String &name, Flags flags=FlagNone, uint32_t quality=95) const
- bool **save** (Stream &stream, Flags flags=FlagNone, uint32_t quality=95) const
- bool swap (uint32_t component_0, uint32_t component_1)

image components

- bool **copy** (const Image &src, uint32_t dest_component, uint32_t src_component)
- bool flipX (const Region &region, const Slice &slice)

  flip horizontally
- bool **flipX** (const Region &region)
- bool **flipX** ()
- bool flipY (const Region &region, const Slice &slice)

  flip vertically
- bool **flipY** (const Region &region)
- bool **flipY** ()
- bool copy (const Image &src, const Origin &dest_origin, const Region &src_region, const Slice &dest_slice, const Slice &src_slice)

  copy image
- bool **copy** (const Image &src, const Origin &dest_origin, const Region &src_region)
- bool **copy** (const Image &src, const Origin &dest_origin, const Slice &dest_slice)
- bool **copy** (const Image &src, const Slice &dest_slice, const Slice &src_slice)
- bool **copy** (const Image &src, const Origin &dest_origin)
- bool **copy** (const Image &src, const Slice &dest_slice)
- Image toType (Type type, Flags flags, Async *async=nullptr) const

  convert image to type
- Image **toType** (Type type, Async *async=nullptr) const
- Image toFormat (Format format, Flags flags, Async *async=nullptr) const

  convert image to format
- Image **toFormat** (Format format, Async *async=nullptr) const
- Image getSlice (const Slice &slice) const

  get image slice
- Image getComponent (uint32_t component) const

  get image component
- Image getRegion (const Region &region, const Slice &slice) const

  get image region
- Image **getRegion** (const Region &region) const
- Image getRotated (int32_t angle, const Slice &slice) const

  get rotated image
- Image **getRotated** (int32_t angle) const
- Image getResized (const Size &size, Filter min, Filter mag, Flags flags, Async *async=nullptr) const

  get resized image
- Image **getResized** (const Size &size, Filter min, Filter mag=FilterCubic, Async *async=nullptr) const
- Image **getResized** (const Size &size, Async *async=nullptr) const
- Image getMipmapped (Filter filter, Flags flags, Async *async=nullptr) const

  get mipmapped image
- Image **getMipmapped** (Filter filter, Async *async=nullptr) const
- Image **getMipmapped** (Async *async=nullptr) const
- int32_t compare (const Image &image) const

  compare images
- const uint8_t * getData (const Slice &slice=Slice()) const

  image data
- uint8_t * **getData** (const Slice &slice=Slice())
- const uint8_t * **getData** (const Origin &origin, const Slice &slice=Slice()) const
- uint8_t * **getData** (const Origin &origin, const Slice &slice=Slice())
- bool **setData** (const void *src, const Slice &slice=Slice(), uint32_t alignment=1, size_t stride=0)
- bool **getData** (void *dest, const Slice &slice=Slice(), uint32_t alignment=1, size_t stride=0) const
- size_t getMemory () const

  memory usage

Static Public Member Functions

- static const char * **getTypeName** (Type type)

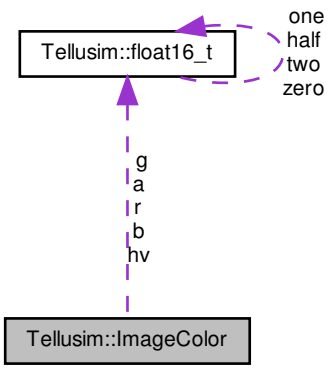
5.169.1 Detailed Description

The Image class provides a comprehensive set of methods for working with images, supporting various image types (2D, 3D, Cube) and formats. It allows the creation, manipulation, and conversion of images, with functionality for operations like loading, saving, resizing, flipping, rotating, and comparing images. The class supports a variety of image flags to control behavior (e.g., mipmaps, compression), as well as different image filters for resizing and minification. It offers methods for querying image properties such as dimensions, format, components, and layout, and includes the ability to manipulate image slices, regions, and components. The class also provides advanced features like handling image metadata, creating mipmaps, and obtaining the raw image data for processing or manipulation. Additionally, it supports asynchronous operations for optimized performance when dealing with large image data.

5.170 Tellusim::ImageColor Struct Reference

```
#include <format/TellusimImage.h>
```

Collaboration diagram for Tellusim::ImageColor:



Public Types

- enum { **Size** = 4 }

Public Member Functions

- **ImageColor** (const ImageColor &c)
- **ImageColor** (int32_t i)
- **ImageColor** (uint32_t u)
- **ImageColor** (float16_t h)
- **ImageColor** (float32_t f)
- **ImageColor** (int32_t l, int32_t a)
- **ImageColor** (uint32_t l, uint32_t a)
- **ImageColor** (float16_t l, float16_t a)
- **ImageColor** (float32_t l, float32_t a)
- **ImageColor** (int32_t r, int32_t g, int32_t b, int32_t a)
- **ImageColor** (uint32_t r, uint32_t g, uint32_t b, uint32_t a)
- **ImageColor** (float16_t r, float16_t g, float16_t b, float16_t a)
- **ImageColor** (float32_t r, float32_t g, float32_t b, float32_t a)
- **ImageColor** (const Color &color, Format format)
- void set (const Color &color, Format format)

  color value
- Color **get** (Format format) const
- ImageColor & operator= (const ImageColor &c)

  assignment operator

Public Attributes

-
```
union {
  struct {
    int32_t r
    int32_t g
    int32_t b
    int32_t a
  } i
  struct {
    uint32_t r
    uint32_t g
    uint32_t b
    uint32_t a
  } u
  struct {
    float16_t r
    float16_t g
    float16_t b
    float16_t a
  } h
  struct {
    float32_t r
    float32_t g
    float32_t b
    float32_t a
  } f
  int32_t iv [Size]
  uint32_t uv [Size]
  float16_t hv [Size]
  float32_t fv [Size]
};
```

5.170.1 Detailed Description

The ImageColor struct represents a color value with multiple formats, including integer, unsigned integer, half-precision, and full-precision floating point. It supports flexibility in initialization, allowing the color to be specified as individual components or as a single value for all components. The struct provides multiple constructors for different data types, including support for copying, as well as conversion between different color formats.

5.171 Tellusim::ImageSampler Class Reference

`#include <format/TellusimImage.h>`

Public Types

- using **Filter** = Image::Filter

Public Member Functions

- **ImageSampler** (Image &image, const Slice &slice=Slice())
- **ImageSampler** (const Image &image, const Slice &slice=Slice())
- void clear ()

  clear sampler
- bool isCreated () const

  check sampler
- Type getType () const

  sampler type
- bool **is2DType** () const
- bool **is3DType** () const
- bool **isCubeType** () const
- Format getFormat () const

  sampler format
- const char * **getFormatName** () const
- uint32_t getWidth () const

  sampler dimension
- uint32_t **getHeight** () const
- uint32_t **getDepth** () const
- uint32_t **getFaces** () const
- size_t **getTexels** () const
- Size **getSize** () const
- Region **getRegion** () const
- size_t getStride () const

  sampler layout
- size_t **getLayerSize** () const
- uint32_t **getPixelSize** () const
- uint32_t **getComponents** () const
- const uint8_t * getData () const

  sampler data
- uint8_t * **getData** ()
- bool create (Image &image, const Slice &slice=Slice())

  create sampler
- bool **create** (const Image &image, const Slice &slice=Slice())

- bool **create** (Type type, Format format, const Size &size, size_t stride, void *data)
- bool **create** (Type type, Format format, const Size &size, size_t stride, const void *data)
- bool **create** (Type type, Format format, const Size &size, size_t stride, size_t layer_size, void *data)
- bool **create** (Type type, Format format, const Size &size, size_t stride, size_t layer_size, const void *data)
- bool clear (const Color &color)

  clear image
- bool **clear** (const ImageColor &color)
- bool mad (const Color &m, const Color &a)

  multiply accumulate image
- void set2D (uint32_t x, uint32_t y, const ImageColor &color)

  2D image colors
- ImageColor **get2D** (uint32_t x, uint32_t y, bool repeat=false) const
- ImageColor **get2D** (float64_t x, float64_t y, bool repeat=false, Filter filter=Image::FilterLinear) const
- void set3D (uint32_t x, uint32_t y, uint32_t z, const ImageColor &color)

  3D image colors
- ImageColor **get3D** (uint32_t x, uint32_t y, uint32_t z, bool repeat=false) const
- ImageColor **get3D** (float32_t x, float32_t y, float32_t z, bool repeat=false, Filter filter=Image::FilterLinear) const
- void setCube (float32_t x, float32_t y, float32_t z, const ImageColor &color)

  Cube image colors.
- ImageColor **getCube** (float32_t x, float32_t y, float32_t z, Filter filter=Image::FilterLinear) const
- uint32_t **getCubeFace** (float32_t x, float32_t y, float32_t z, float32_t &tx, float32_t &ty) const
- void setTexel (size_t t, const ImageColor &color)

  Texel image colors.
- ImageColor **getTexel** (size_t t) const

5.171.1 Detailed Description

The ImageSampler class provides a flexible and efficient interface for working with image data in various formats and dimensions. It supports 2D, 3D, and Cube image types, allowing for easy manipulation of image color data at the texel level. The class offers constructors for creating samplers from both mutable and immutable images, with optional slice support.

5.172 Tellusim::ImageStream Class Reference

`#include <format/TellusimImage.h>`

Public Member Functions

- virtual bool **info** (Stream &stream, Image &image, Image::Flags flags, uint32_t offset, Async *async)
- virtual bool **load** (Stream &stream, Image &image, Image::Flags flags, uint32_t offset, Async *async)
- virtual bool **save** (Stream &stream, const Image &image, Image::Flags flags, uint32_t quality)

Static Public Member Functions

- static bool check (const String &name, uint32_t magic=0)

  image stream formats
- static String getLoadFormats ()

  list of supported formats
- static String **getSaveFormats** ()

Protected Types

- enum **Flags** {
  FlagNone = 0,
  FlagLoad = (1 << 0),
  FlagSave = (1 << 1),
  FlagLoadSave = (FlagLoad | FlagSave) }

Protected Member Functions

- **ImageStream** (Flags flags, const char *name, uint32_t magic=0)
- **ImageStream** (Flags flags, const InitializerList< const char *> &names, uint32_t magic=0)
- **ImageStream** (Flags flags, const InitializerList< const char *> &names, const InitializerList< uint32_t > &magics)

5.172.1 Detailed Description

The ImageStream class is a base class designed for creating custom image stream formats, providing virtual methods for loading and saving images through streams. It supports handling different image types and formats, serving as a foundation for implementing specific image stream formats and enabling the customization and extension of image handling functionality. The class also includes static methods for checking supported formats and retrieving lists of compatible load and save formats, offering flexibility in managing and working with various image stream formats.

5.173 Tellusim::Info Class Reference

```
#include <core/TellusimSystem.h>
```

Public Member Functions

- size_t getSystemMemory () const
  System info.
- uint64_t **getSystemUptime** () const
- String **getSystemName** () const
- String **getSystemVersion** () const
- String **getKernelVersion** () const
- uint32_t getCPUCount () const
  CPU info.
- String **getCPUName** (uint32_t index) const
- String **getCPUVendor** (uint32_t index) const
- uint32_t **getCPUCores** (uint32_t index) const
- uint32_t **getCPUThreads** (uint32_t index) const
- uint64_t **getCPUFrequency** (uint32_t index) const
- uint32_t **getCPUTemperature** (uint32_t index) const
- uint32_t **getCPUUtilization** (uint32_t index) const
- uint32_t **getCPUFanSpeed** (uint32_t index) const
- uint32_t **getCPUPower** (uint32_t index) const
- uint32_t getGPUCount () const
  GPU info.
- String **getGPUName** (uint32_t index) const

- String **getGPUVendor** (uint32_t index) const
- String **getGPUSerial** (uint32_t index) const
- String **getGPUDevice** (uint32_t index) const
- String **getGPUVersion** (uint32_t index) const
- size_t **getGPUMemory** (uint32_t index) const
- uint32_t **getGPUScreens** (uint32_t index) const
- uint64_t **getGPUFrequency** (uint32_t index) const
- uint32_t **getGPUTemperature** (uint32_t index) const
- uint32_t **getGPUUtilization** (uint32_t index) const
- uint32_t **getGPUFanSpeed** (uint32_t index) const
- uint32_t **getGPUPower** (uint32_t index) const
- bool **isGPUThrottling** (uint32_t index) const

5.173.1 Detailed Description

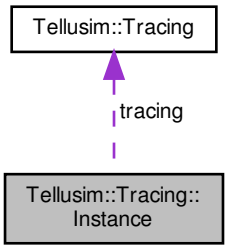
The Info class provides detailed system and hardware information, offering methods to retrieve data on system memory, uptime, and version, as well as kernel details. CPU-related queries include processor details like core count, temperature, and frequency. For GPU information, aspects such as memory size, frequency, utilization, and power are available, along with the ability to detect throttling states. The class offers a comprehensive suite of functions for gathering detailed system performance and hardware status.

5.174 Tellusim::Tracing::Instance Struct Reference

tracing instance

`#include <platform/TellusimTracing.h>`

Collaboration diagram for Tellusim::Tracing::Instance:



Public Attributes

- float32_t **transform** [12]
- uint32_t **data**
- uint32_t **mask**
- uint32_t **flags**
- uint32_t **offset**
- Tracing * **tracing**

5.174.1 Detailed Description

tracing instance

5.175 Tellusim::int16x8_t Struct Reference

`#include <math/TellusimSimd.h>`

Public Types

- enum { **Size** = 8 }

Public Member Functions

- int16x8_t (const int32x8_t &v)
- **int16x8_t** (const int16_t *v)
- **int16x8_t** (const int32x4_t &v0, const int32x4_t &v1)
- **int16x8_t** (int16_t v)
- **int16x8_t** (int16_t x0, int16_t y0, int16_t z0, int16_t w0, int16_t x1, int16_t y1, int16_t z1, int16_t w1)
- int32x4_t asi32x4 () const

  cast vector data
- uint16x8_t **asu16x8** () const
- uint32x4_t **asu32x4** () const
- float16x8_t **asf16x8** () const
- float32x4_t **asf32x4** () const
- void set (const int16x8_t &v)

  update vector data
- void **set** (int16_t X0, int16_t Y0, int16_t Z0, int16_t W0, int16_t X1, int16_t Y1, int16_t Z1, int16_t W1)
- void **set** (const int16_t *1 v)
- void **get** (int16_t *1 v) const
- template<uint32_t Index>
  void **set** (int16_t V)
- template<uint32_t Index>
  int16_t **get** () const
- int16x8_t & operator*= (int16_t v)

  vector to scalar operators
- int16x8_t & **operator+=** (int16_t v)
- int16x8_t & **operator-=** (int16_t v)
- int16x8_t & **operator &=** (int16_t v)
- int16x8_t & **operator|=** (int16_t v)
- int16x8_t & **operator^=** (int16_t v)
- int16x8_t & **operator<<=** (int16_t v)
- int16x8_t & **operator>>=** (int16_t v)
- int16x8_t & operator*= (const int16x8_t &v)

  vector to vector operators
- int16x8_t & **operator+=** (const int16x8_t &v)
- int16x8_t & **operator-=** (const int16x8_t &v)
- int16x8_t & **operator &=** (const int16x8_t &v)
- int16x8_t & **operator|=** (const int16x8_t &v)
- int16x8_t & **operator^=** (const int16x8_t &v)

- int16x8_t xyzw10 () const
  swizzle vector
- int16x8_t **zwxy01** () const
- int16x8_t **yxwz01** () const
- int32x4_t xyzw0 () const
  swizzle vector
- int32x4_t **xyzw1** () const
- int16_t sum () const
  sum vector components

Public Attributes

- union {
  struct {
  int16_t **x0**
  int16_t **y0**
  int16_t **z0**
  int16_t **w0**
  int16_t **x1**
  int16_t **y1**
  int16_t **z1**
  int16_t **w1**
  }
  int16_t **v** [Size]
  };

5.175.1 Detailed Description

SSE Utils AVX utils NEON utils Vector of eight int16_t components

5.175.2 Constructor & Destructor Documentation

5.175.2.1 int16x8_t()

```
Tellusim::int16x8_t::int16x8_t (
            const int32x8_t & v )  [explicit]
```

Vector of eight int16_t components

5.176 Tellusim::int32x4_t Struct Reference

```
#include <math/TellusimSimd.h>
```

Public Types

- enum { **Size** = 4 }

Public Member Functions

- int32x4_t (const uint32x4_t &v)
- **int32x4_t** (const float32x4_t &v)
- **int32x4_t** (const float64x4_t &v)
- **int32x4_t** (const int32_t *v)
- **int32x4_t** (const int32_t *v, int32_t w)
- **int32x4_t** (int32_t v)
- **int32x4_t** (int32_t x, int32_t y, int32_t z, int32_t w=0)
- int16x8_t asi16x8 () const

  cast vector data
- uint16x8_t asu16x8 () const

  cast vector data
- uint32x4_t **asu32x4** () const
- float16x8_t **asf16x8** () const
- float32x4_t **asf32x4** () const
- void set (const int32x4_t &v)

  update vector data
- void **set** (int32_t X, int32_t Y, int32_t Z, int32_t W)
- void **set** (const int32_t *1 v, int32_t W)
- void **set** (const int32_t *1 v)
- void **get** (int32_t *1 v) const
- template<uint32_t Index>
  void **set** (int32_t V)
- template<uint32_t Index>
  int32_t **get** () const
- template<uint32_t Index>
  int32x4_t **get4** () const
- int32x4_t & operator*= (int32_t v)

  vector to scalar operators
- int32x4_t & **operator+=** (int32_t v)
- int32x4_t & **operator-=** (int32_t v)
- int32x4_t & **operator &=** (int32_t v)
- int32x4_t & **operator|=** (int32_t v)
- int32x4_t & **operator^=** (int32_t v)
- int32x4_t & **operator<<=** (int32_t v)
- int32x4_t & **operator>>=** (int32_t v)
- int32x4_t & operator*= (const int32x4_t &v)

  vector to vector operators
- int32x4_t & **operator+=** (const int32x4_t &v)
- int32x4_t & **operator-=** (const int32x4_t &v)
- int32x4_t & **operator &=** (const int32x4_t &v)
- int32x4_t & **operator|=** (const int32x4_t &v)
- int32x4_t & **operator^=** (const int32x4_t &v)
- int32x4_t zwxy () const

  swizzle vector
- int32x4_t **yxwz** () const
- int32_t sum () const

  sum vector components

Public Attributes

- union {
  struct {
  int32_t **x**
  int32_t **y**
  int32_t **z**
  int32_t **w**
  }
  int32_t **v** [Size]
  };

5.176.1 Detailed Description

Vector of four int32_t components

5.176.2 Constructor & Destructor Documentation

5.176.2.1 int32x4_t()

```
Tellusim::int32x4_t::int32x4_t (
            const uint32x4_t & v )  [explicit]
```

Vector of four int32_t components

5.177 Tellusim::int32x8_t Struct Reference

```
#include <math/TellusimSimd.h>
```

Public Types

- enum { **Size** = 8 }

Public Member Functions

- int32x8_t (const uint32x8_t &v)
- **int32x8_t** (const float32x8_t &v)
- **int32x8_t** (const float64x8_t &v)
- **int32x8_t** (const int32_t *v)
- **int32x8_t** (int32_t v)
- **int32x8_t** (const int16x8_t &v)
- **int32x8_t** (const int32x4_t &v0, const int32x4_t &v1)
- **int32x8_t** (int32_t x0, int32_t y0, int32_t z0, int32_t w0, int32_t x1, int32_t y1, int32_t z1, int32_t w1)
- uint32x8_t asu32x8 () const

  cast vector data
- float32x8_t **asf32x8** () const
- void set (const int32x8_t &v)

  update vector data
- void **set** (int32_t X0, int32_t Y0, int32_t Z0, int32_t W0, int32_t X1, int32_t Y1, int32_t Z1, int32_t W1)
- void **set** (const int32_t *1 v)
- void **get** (int32_t *1 v) const
- template<uint32_t Index>
  void **set** (int32_t V)
- template<uint32_t Index>
  int32_t **get** () const
- template<uint32_t Index>
  int32x8_t **get8** () const
- int32x8_t & operator*= (int32_t v)

  vector to scalar operators
- int32x8_t & **operator+=** (int32_t v)
- int32x8_t & **operator-=** (int32_t v)
- int32x8_t & **operator &=** (int32_t v)
- int32x8_t & **operator|=** (int32_t v)
- int32x8_t & **operator^=** (int32_t v)
- int32x8_t & **operator<<=** (int32_t v)
- int32x8_t & **operator>>=** (int32_t v)
- int32x8_t & operator*= (const int32x8_t &v)

  vector to vector operators
- int32x8_t & **operator+=** (const int32x8_t &v)
- int32x8_t & **operator-=** (const int32x8_t &v)
- int32x8_t & **operator &=** (const int32x8_t &v)
- int32x8_t & **operator|=** (const int32x8_t &v)
- int32x8_t & **operator^=** (const int32x8_t &v)
- int32x8_t xyzw10 () const

  swizzle vector
- int32x8_t **zwxy01** () const
- int32x8_t **yxwz01** () const
- int32x4_t **xyzw0** () const
- int32x4_t **xyzw1** () const
- int32_t sum () const

  sum vector components

Public Attributes

- union {
  struct {
  int32_t **x0**
  int32_t **y0**
  int32_t **z0**
  int32_t **w0**
  int32_t **x1**
  int32_t **y1**
  int32_t **z1**
  int32_t **w1**
  }
  int32_t **v** [Size]
  };

5.177.1 Detailed Description

Vector of eight int32_t components

5.177.2 Constructor & Destructor Documentation

5.177.2.1 int32x8_t()

```
Tellusim::int32x8_t::int32x8_t (
            const uint32x8_t & v )  [explicit]
```

Vector of eight int32_t components

5.178 Tellusim::IsPtr< Type > Struct Template Reference

```
#include <TellusimBase.h>
```

Public Types

- enum { **Result** = 0 }

5.178.1 Detailed Description

template<class Type>
struct Tellusim::IsPtr< Type >

Pointer type

5.179 Tellusim::RadixMap< Key, Type, Size >::Iterator Class Reference

Iterator.

```
#include <core/TellusimRadix.h>
```

Public Member Functions

- **Iterator** (const Iterator &it)
- void **clear** ()
- Iterator & **operator=** (const Iterator &it)
- **operator bool** () const
- bool **operator==** (const Iterator &it) const
- bool **operator!=** (const Iterator &it) const
- Iterator & **operator++** ()
- Iterator & **operator--** ()
- Iterator **operator++** (int32_t)
- Iterator **operator--** (int32_t)
- Iterator **next** ()
- Iterator **prev** ()
- Type & **operator*** ()
- Type * **operator->** ()
- Type & **get** ()

Friends

- class **RadixMap**

5.179.1 Detailed Description

template<class Key, class Type, uint32_t Size = 32>
class Tellusim::RadixMap< Key, Type, Size >::Iterator

Iterator.

5.180 Tellusim::Json Class Reference

```
#include <format/TellusimJson.h>
```

Public Member Functions

- **Json** (Type type)
- **Json** (const char *name, Type type=TypeUnknown)
- **Json** (const String &name, Type type=TypeUnknown)
- **Json** (Json *parent, const char *name, Type type=TypeUnknown)
- **Json** (Json *parent, const String &name, Type type=TypeUnknown)
- void clear ()

  clear json
- bool create (const char *str, size_t size=0, bool owner=false)

  create json
- bool **create** (const String &str, size_t size=0, bool owner=false)
- bool load (const char *name)

  load json
- bool **load** (const String &name)
- bool **load** (Stream &stream)
- bool save (const char *name, bool compact=false) const

  save json
- bool **save** (const String &name, bool compact=false) const
- bool **save** (Stream &stream, bool compact=false) const
- const Json getRoot () const

  json root
- Json **getRoot** ()
- uint32_t setParent (Json &parent, bool check=true)

  json parent
- const Json **getParent** () const
- Json **getParent** ()
- Json addChild (const char *name, Type type=TypeUnknown, bool check=true)

  json children
- uint32_t **addChild** (Json &child, bool check=true)
- bool **removeChild** (Json &child)
- void **releaseChildren** ()
- uint32_t **findChild** (const char *name) const
- bool **isChild** (const char *name) const
- const Json **getChild** (const char *name) const
- Json **getChild** (const char *name)
- uint32_t **getNumChildren** () const
- const Array< Json > **getChildren** () const
- Array< Json > **getChildren** ()
- const Json **getChild** (uint32_t index) const
- Json **getChild** (uint32_t index)
- String getPathName () const

  json path name
- void setName (const char *name)

  json name
- void **setName** (const String &name)
- String **getName** () const
- void setType (Type type)

  json type
- Type **getType** () const
- const char * **getTypeName** () const
- bool **isUnknown** () const
- bool **isNull** () const

- bool **isBool** () const
- bool **isNumber** () const
- bool **isString** () const
- bool **isObject** () const
- bool **isArray** () const
- void setData (bool value)
- void **setData** (int32_t value, uint32_t radix=10)
- void **setData** (uint32_t value, uint32_t radix=10)
- void **setData** (uint64_t value, uint32_t radix=10)
- void **setData** (float32_t value, uint32_t digits=6, bool compact=true, bool exponent=true)
- void **setData** (float64_t value, uint32_t digits=12, bool compact=true, bool exponent=true)
- void **setData** (const char *value)
- void **setData** (const String &value)
- template<class Type >
  Json **setData** (const char *name, Type value, Json::Type type=TypeUnknown)
- String **getData** () const
- bool **getDataBool** () const
- int32_t **getDatai32** (uint32_t radix=10) const
- uint32_t **getDatau32** (uint32_t radix=10) const
- uint64_t **getDatau64** (uint32_t radix=10) const
- float32_t **getDataf32** () const
- float64_t **getDataf64** () const
- String **getNumber** () const
- String **getString** () const
- bool **getData** (const char *name, bool value) const
- int32_t **getData** (const char *name, int32_t value, uint32_t radix=10) const
- uint32_t **getData** (const char *name, uint32_t value, uint32_t radix=10) const
- uint64_t **getData** (const char *name, uint64_t value, uint32_t radix=10) const
- float32_t **getData** (const char *name, float32_t value) const
- float64_t **getData** (const char *name, float64_t value) const
- String **getData** (const char *name, const String &value=String::null) const
- void setData (const char **values, uint32_t size)
- void **setData** (const String *values, uint32_t size)
- void **setData** (const int32_t *values, uint32_t size, uint32_t radix=10)
- void **setData** (const uint32_t *values, uint32_t size, uint32_t radix=10)
- void **setData** (const float32_t *values, uint32_t size, uint32_t digits=6, bool compact=true, bool exponent=true)
- void **setData** (const float64_t *values, uint32_t size, uint32_t digits=12, bool compact=true, bool exponent=true)
- template<class Type >
  Json **setData** (const char *name, Type *values, uint32_t size, Json::Type type=TypeUnknown)
- template<class Type >
  void **setData** (const Array< Type > &values)
- template<class Type >
  void **setData** (const char *name, const Array< Type > &values)
- uint32_t **getData** (String *values, uint32_t size) const
- uint32_t **getData** (int32_t *values, uint32_t size, uint32_t radix=10) const
- uint32_t **getData** (uint32_t *values, uint32_t size, uint32_t radix=10) const
- uint32_t **getData** (float32_t *values, uint32_t size) const
- uint32_t **getData** (float64_t *values, uint32_t size) const
- template<class Type >
  uint32_t **getData** (const char *name, Type *values, uint32_t size) const
- template<class Type >
  uint32_t **getData** (Array< Type > &values) const
- template<class Type >
  uint32_t **getData** (const char *name, Array< Type > &values) const

Static Public Member Functions

- static const char * **getTypeName** (Type type)

5.180.1 Detailed Description

The Json class provides a flexible and efficient interface for working with Json data. It supports a wide range of functionalities for creating, manipulating, and querying Json data, including handling Json objects, arrays, strings, numbers, booleans, and more. This class is designed to work with nested structures and offers various methods for loading, saving, and editing Json content.

5.180.2 Member Function Documentation

5.180.2.1 setData() [1/2]

```
void Tellusim::Json::setData (
            bool value )
```

json data

Parameters

radix	The decimal number radix (use 16 for hexadecimal numbers).
digits	The number of digits in the floating-point representation.
compact	Remove redundant zeros at the end of the number.
exponent	Use exponent representation.

5.180.2.2 setData() [2/2]

```
void Tellusim::Json::setData (
            const char ** values,
            uint32_t size )
```

json array data

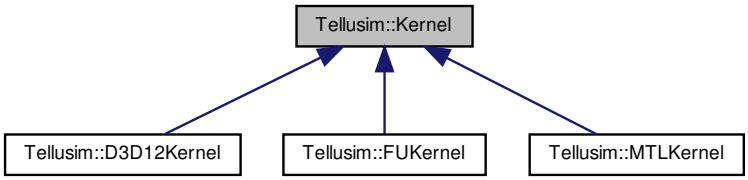
Parameters

radix	The decimal number radix (use 16 for hexadecimal numbers).
digits	The number of digits in the floating-point representation.
compact	Remove redundant zeros at the end of the number.
exponent	Use exponent representation.

5.181 Tellusim::Kernel Class Reference

```
#include <platform/TellusimKernel.h>
```

Inheritance diagram for Tellusim::Kernel:



Public Member Functions

- Platform getPlatform () const
  kernel platform
- const char * **getPlatformName** () const
- uint32_t getIndex () const
  kernel device index
- void clear ()
  clear kernel
- bool isCreated () const
  check kernel
- void setName (const char *name)
  kernel name
- String **getName** () const
- bool create ()
  create kernel
- void setParameters (const Kernel &kernel)
  kernel parameters
- bool **saveState** (Stream &stream) const
- void setShader (Shader &shader, bool owner=false)
  shader pointer
- Shader **getComputeShader** () const
- bool loadShader (const char *name, const char *format,...) 1(3
  load shaders
- bool bool **loadShaderGLSL** (const char *name, const char *format,...) 1(3
- bool bool bool **loadShader** (const char *name, const String &macros=String::null, const char **includes=nullptr, uint32_t size=0)
- bool **loadShaderGLSL** (const char *name, const String &macros=String::null, const char **includes=nullptr, uint32_t size=0)
- bool **loadShaderSPIRV** (const char *name)
- bool createShader (const char *src, const char *format,...) 1(3
  create shaders
- bool bool **createShaderGLSL** (const char *src, const char *format,...) 1(3

- bool bool bool **createShader** (const char *src, const String &macros=String::null, const char **includes=nullptr, uint32_t size=0)
- bool **createShaderGLSL** (const char *src, const String &macros=String::null, const char **includes=nullptr, uint32_t size=0)
- bool **createShaderSPIRV** (const Array< uint32_t > &data)
- uint32_t addSampler ()

  sampler parameters
- Kernel & **setSamplers** (uint32_t num)
- uint32_t **getNumSamplers** () const
- Kernel & **setSamplerOffset** (uint32_t offset)
- uint32_t **getSamplerOffset** () const
- Kernel & **setSamplerArray** (uint32_t index, uint32_t num, bool array)
- uint32_t **getSamplerArray** (uint32_t index) const
- uint32_t addTexture ()

  texture parameters
- Kernel & **setTextures** (uint32_t num)
- uint32_t **getNumTextures** () const
- Kernel & **setTextureOffset** (uint32_t offset)
- uint32_t **getTextureOffset** () const
- Kernel & **setTextureArray** (uint32_t index, uint32_t num, bool array)
- uint32_t **getTextureArray** (uint32_t index) const
- uint32_t addSurface ()

  surface parameters
- Kernel & **setSurfaces** (uint32_t num)
- uint32_t **getNumSurfaces** () const
- Kernel & **setSurfaceOffset** (uint32_t offset)
- uint32_t **getSurfaceOffset** () const
- Kernel & **setSurfaceArray** (uint32_t index, uint32_t num, bool array)
- uint32_t **getSurfaceArray** (uint32_t index) const
- uint32_t addUniform (BindFlags flags=BindFlagNone)

  uniform parameters
- Kernel & **setUniforms** (uint32_t num, BindFlags flags=BindFlagNone)
- uint32_t **getNumUniforms** () const
- Kernel & **setUniformOffset** (uint32_t offset)
- uint32_t **getUniformOffset** () const
- Kernel & **setUniformFlags** (uint32_t index, BindFlags flags)
- BindFlags **getUniformFlags** (uint32_t index) const
- uint32_t addStorage (BindFlags flags=BindFlagNone)

  storage parameters
- Kernel & **setStorages** (uint32_t num, BindFlags flags=BindFlagNone)
- uint32_t **getNumStorages** () const
- Kernel & **setStorageOffset** (uint32_t offset)
- uint32_t **getStorageOffset** () const
- Kernel & **setStorageFlags** (uint32_t index, BindFlags flags)
- BindFlags **getStorageFlags** (uint32_t index) const
- uint32_t addTracing ()

  tracing parameters
- Kernel & **setTracings** (uint32_t num)
- uint32_t **getNumTracings** () const
- Kernel & **setTracingOffset** (uint32_t offset)
- uint32_t **getTracingOffset** () const
- uint32_t addTexel ()

  texel parameters
- Kernel & **setTexels** (uint32_t num)

- uint32_t **getNumTexels** () const
- Kernel & **setTexelOffset** (uint32_t offset)
- uint32_t **getTexelOffset** () const
- uint32_t addTable (TableType type, uint32_t size)

  table parameters
- uint32_t **getNumTables** () const
- Kernel & **setTableOffset** (uint32_t offset)
- uint32_t **getTableOffset** () const
- Kernel & **setTableType** (uint32_t index, TableType type, uint32_t size, BindFlags flags=BindFlagNone)
- TableType **getTableType** (uint32_t index) const
- uint32_t **getTableSize** (uint32_t index) const
- Kernel & **setTableFlags** (uint32_t index, BindFlags flags)
- BindFlags **getTableFlags** (uint32_t index) const
- void setGroupSize (uint32_t width, uint32_t height=1, uint32_t depth=1)

  thread group size
- uint32_t **getGroupSizeX** () const
- uint32_t **getGroupSizeY** () const
- uint32_t **getGroupSizeZ** () const

5.181.1 Detailed Description

The Kernel class represents a compute kernel responsible for managing shaders, textures, buffers, samplers, surfaces, uniforms, and storage within a computational pipeline. It allows users to set and retrieve various parameters, including device index, platform, and kernel state, while supporting shader compilation, loading, and creation in formats like native, GLSL, and SPIRV. The class also provides functionality for configuring kernel parameters such as texture, surface, and storage settings, along with memory alignment and sampler configurations, offering comprehensive control over the pipeline compute operations.

5.182 Tellusim::MeshTransform::KeyData< Type > Struct Template Reference

transform data

```
#include <format/TellusimMesh.h>
```

Public Attributes

- float64_t **time**
- Type **data**

5.182.1 Detailed Description

template<class Type>
struct Tellusim::MeshTransform::KeyData< Type >

transform data

5.183 Tellusim::Layer Struct Reference

```
#include <TellusimTypes.h>
```

Public Member Functions

- **Layer** (uint32_t base)
- **Layer** (uint32_t base, uint32_t size)

Public Attributes

- uint32_t **base** = 0
- uint32_t **size** = 1

5.183.1 Detailed Description

The Layer struct defines a layer in a texture array, where base specifies the starting index of the layer, and size represents the number of layers. Layers allow the storage of multiple images in a single texture, typically used for 2D and Cube array textures.

5.184 Tellusim::SpatialTree::LeafNodef16 Struct Reference

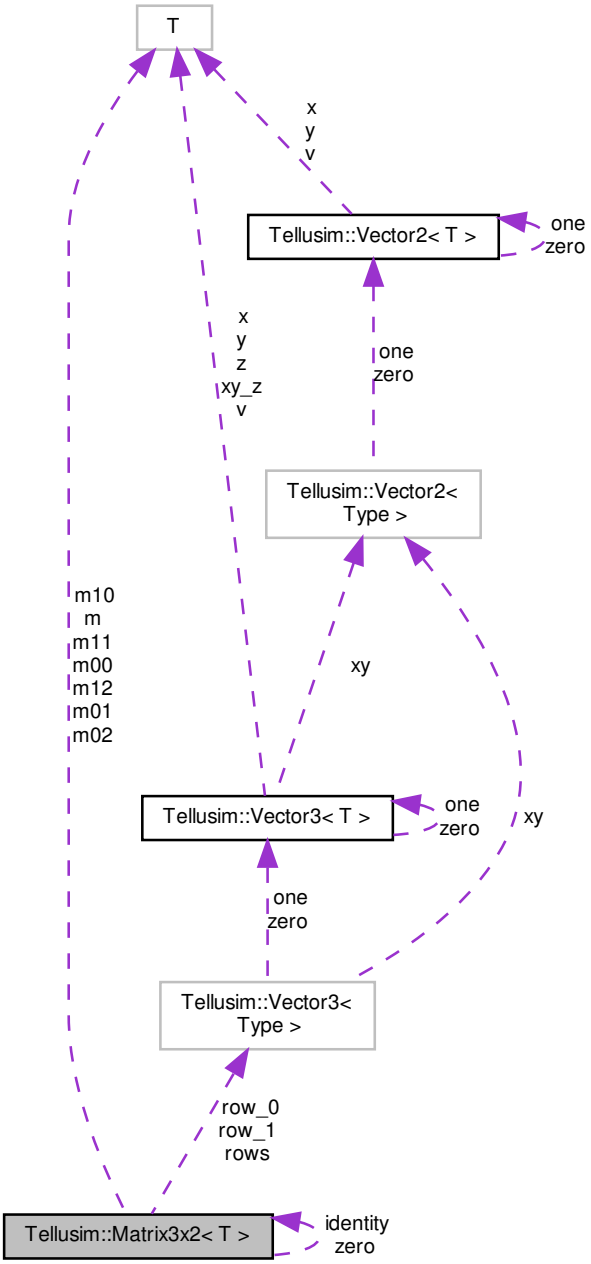
Public Attributes

- float32_t **center** [3]
- uint32_t **data**
- uint16_t **size** [3]
- uint16_t **is_enabled**
- uint32_t **data_1**
- uint32_t **data_2**
- uint32_t **left**
- uint32_t **right**
- uint32_t **parent**
- uint32_t **spatial**

5.185 Tellusim::Matrix3x2< T > Struct Template Reference

`#include <math/TellusimMatrix.h>`

Collaboration diagram for Tellusim::Matrix3x2< T >:



Public Types

- enum {
  Rows = 2,
  Columns = 3,
  Size = Columns ∗ Rows }
- using **Vector2** = Tellusim::Vector2< Type >
- using **Vector3** = Tellusim::Vector3< Type >

Public Member Functions

- **Matrix3x2** (const Matrix3x2 &m)
- **Matrix3x2** (const Vector3 &row_0, const Vector3 &row_1)
- **Matrix3x2** (const Vector2 &col_0, const Vector2 &col_1, const Vector2 &col_2)
- **Matrix3x2** (Type m00, Type m01, Type m02, Type m10, Type m11, Type m12)
- **Matrix3x2** (const Type *1 m, uint32_t size=Size, bool row_major=true)
- template<class CType >
  Matrix3x2 (const Tellusim::Matrix3x2< CType > &m)
- **Matrix3x2** (Type v)
- void set (const Vector3 &r0, const Vector3 &r1)

  update matrix data
- void **set** (const Vector3 &col_0, const Vector3 &col_1, const Vector3 &col_2)
- void **set** (const Type *1 m, uint32_t size=Size, bool row_major=true)
- void **get** (Type *1 m, uint32_t size=Size, bool row_major=true) const
- Matrix3x2 & operator*= (const Matrix3x2 &m1)

  matrix to matrix multiplication
- Matrix3x2 & operator+= (const Matrix3x2 &m)

  matrix to matrix operators
- Matrix3x2 & **operator-=** (const Matrix3x2 &m)
- void setZero ()

  zero matrix
- bool **isZero** () const
- void setIdentity ()

  identity matrix
- bool **isIdentity** () const
- void setOuter (const Vector2 &v0, const Vector2 &v1)

  outer product matrix
- void setScale (const Vector2 &s)

  scaling matrix
- void **setScale** (Type x, Type y)
- Vector2 **getScale** () const
- void setTranslate (const Vector2 &t)

  translation matrix
- void **setTranslate** (Type x, Type y)
- Vector2 **getTranslate** () const
- void setRotate (Type angle)

  rotation matrix
- Matrix3x2 **getRotate** () const
- Type getDeterminant () const

  matrix determinant
- void setComponents (const Vector2 &t, Type r, const Vector2 &scale)

  matrix composition
- void **getComponents** (Vector2 &t, Type &r, Vector2 &s) const
- void setRow (uint32_t index, const Vector3 &r)

  matrix rows
- const Vector3 & **getRow** (uint32_t index) const
- Vector3 & **getRow** (uint32_t index)
- void setColumn (uint32_t index, const Vector2 &c)

  matrix columns
- Vector2 **getColumn** (uint32_t index) const
- const Vector3 & operator[ ] (uint32_t index) const

  matrix data
- Vector3 & **operator[ ]** (uint32_t index)

Static Public Member Functions

- static Matrix3x2 **outer** (const Vector2 &v0, const Vector2 &v1)
- static Matrix3x2 **scale** (const Vector2 &s)
- static Matrix3x2 **scale** (Type x, Type y)
- static Matrix3x2 **scale** (Type s)
- static Matrix3x2 **translate** (const Vector2 &t)
- static Matrix3x2 **translate** (Type x, Type y)
- static Matrix3x2 **rotate** (Type angle)
- static Matrix3x2 **compose** (const Vector2 &t, Type r, const Vector2 &s)

Public Attributes

-
```
union {
  struct {
    Type m00
    Type m01
    Type m02
    Type m10
    Type m11
    Type m12
  }
  struct {
    Vector3 row_0
    Vector3 row_1
  }
  Vector3 rows [Rows]
  Type m [Size]
};
```

Static Public Attributes

- static const Matrix3x2 zero

  default matrices
- static const Matrix3x2 **identity**

5.185.1 Detailed Description

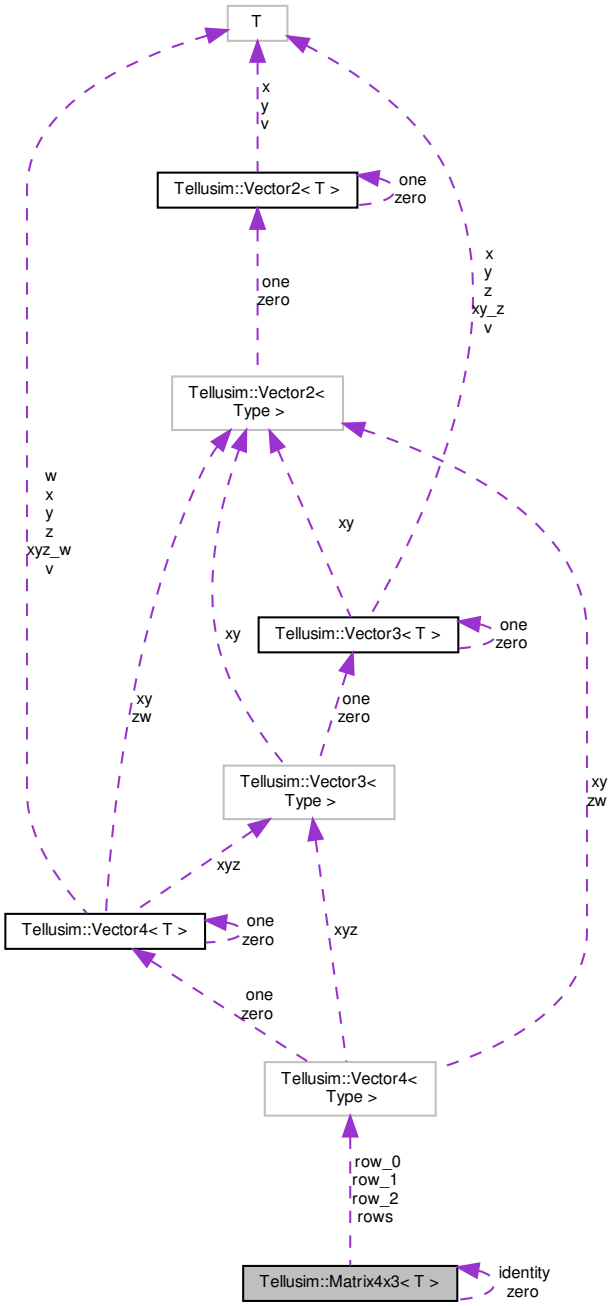
template<class T>
struct Tellusim::Matrix3x2< T >

Matrix3x2 class

5.186 Tellusim::Matrix4x3< T > Struct Template Reference

`#include <math/TellusimMatrix.h>`

Collaboration diagram for Tellusim::Matrix4x3< T >:



Public Types

- enum {
  Rows = 3,

Columns = 4,
Size = Columns ∗ Rows }

- using **Vector3** = Tellusim::Vector3< Type >
- using **Vector4** = Tellusim::Vector4< Type >
- using **Matrix4x4** = Tellusim::Matrix4x4< Type >
- using **Quaternion** = Tellusim::Quaternion< Type >

Public Member Functions

- **Matrix4x3** (const Matrix4x3 &m)
- **Matrix4x3** (const Vector4 &row_0, const Vector4 &row_1, const Vector4 &row_2)
- **Matrix4x3** (const Vector3 &col_0, const Vector3 &col_1, const Vector3 &col_2, const Vector3 &col_3)
- **Matrix4x3** (Type m00, Type m01, Type m02, Type m03, Type m10, Type m11, Type m12, Type m13, Type m20, Type m21, Type m22, Type m23)
- **Matrix4x3** (const Matrix4x4 &m)
- **Matrix4x3** (const Quaternion &q)
- **Matrix4x3** (const Type ∗1 m, uint32_t size=Size, bool row_major=true)
- template<class CType >
  Matrix4x3 (const Tellusim::Matrix4x3< CType > &m)
- template<class CType >
  Matrix4x3 (const Tellusim::Matrix4x4< CType > &m)
- **Matrix4x3** (Type v)
- void set (const Vector4 &r0, const Vector4 &r1, const Vector4 &r2)

  update matrix data
- void **set** (const Vector3 &col_0, const Vector3 &col_1, const Vector3 &col_2, const Vector3 &col_3)
- void **set** (const Type ∗1 m, uint32_t size=Size, bool row_major=true)
- void **get** (Type ∗1 m, uint32_t size=Size, bool row_major=true) const
- Matrix4x3 & operator∗= (const Matrix4x3 &m1)

  matrix to matrix multiplication
- Matrix4x3 & operator+= (const Matrix4x3 &m)

  matrix to matrix operators
- Matrix4x3 & **operator-=** (const Matrix4x3 &m)
- void setZero ()

  zero matrix
- bool **isZero** () const
- void setIdentity ()

  identity matrix
- bool **isIdentity** () const
- void setOuter (const Vector3 &v0, const Vector3 &v1)

  outer product matrix
- void setScale (const Vector3 &s)

  scaling matrix
- void **setScale** (Type x, Type y, Type z)
- Vector3 **getScale** () const
- void setTranslate (const Vector3 &t)

  translation matrix
- void **setTranslate** (Type x, Type y, Type z)
- Vector3 **getTranslate** () const
- void setRotateX (Type angle)

  rotation matrix
- void **setRotateY** (Type angle)
- void **setRotateZ** (Type angle)
- void **setRotate** (const Vector3 &axis, Type angle)

- void **setRotate** (Type x, Type y, Type z, Type angle)
- Matrix4x3 **getRotate** () const
- void setLookAt (const Vector3 &from, const Vector3 &to, const Vector3 &up)

  look at matrix
- void setPlaceTo (const Vector3 &to, const Vector3 &from, const Vector3 &up)

  place to matrix
- void setCubeAt (const Vector3 &from, uint32_t face)

  cube at matrix
- void setBasis (const Vector3 &normal, const Vector3 &t)

  right-handed orthonormal basis
- Type getDeterminant () const

  matrix determinant
- void setComponents (const Vector3 &t, const Quaternion &r)

  matrix composition
- void **setComponents** (const Vector3 &t, const Quaternion &r, const Vector3 &s)
- void **getComponents** (Vector3 &t, Quaternion &r) const
- void **getComponents** (Vector3 &t, Quaternion &r, Vector3 &s) const
- void setRow (uint32_t index, const Vector4 &r)

  matrix rows
- const Vector4 & **getRow** (uint32_t index) const
- Vector4 & **getRow** (uint32_t index)
- void setColumn (uint32_t index, const Vector3 &c)

  matrix columns
- Vector3 **getColumn** (uint32_t index) const
- const Vector4 & operator[ ] (uint32_t index) const

  matrix data
- Vector4 & **operator[ ]** (uint32_t index)

Static Public Member Functions

- static Matrix4x3 **outer** (const Vector3 &v0, const Vector3 &v1)
- static Matrix4x3 **scale** (const Vector3 &s)
- static Matrix4x3 **scale** (Type x, Type y, Type z)
- static Matrix4x3 **scale** (Type s)
- static Matrix4x3 **translate** (const Vector3 &t)
- static Matrix4x3 **translate** (Type x, Type y, Type z)
- static Matrix4x3 **rotateX** (Type angle)
- static Matrix4x3 **rotateY** (Type angle)
- static Matrix4x3 **rotateZ** (Type angle)
- static Matrix4x3 **rotate** (const Vector3 &axis, Type angle)
- static Matrix4x3 **rotate** (Type x, Type y, Type z, Type angle)
- static Matrix4x3 **lookAt** (const Vector3 &from, const Vector3 &to, const Vector3 &up)
- static Matrix4x3 **placeTo** (const Vector3 &to, const Vector3 &from, const Vector3 &up)
- static Matrix4x3 **cubeAt** (const Vector3 &from, uint32_t face)
- static Matrix4x3 **basis** (const Vector3 &normal)
- static Matrix4x3 **basis** (const Vector3 &normal, const Vector3 &t)
- static Matrix4x3 **compose** (const Vector3 &t, const Quaternion &r)
- static Matrix4x3 **compose** (const Vector3 &t, const Quaternion &r, const Vector3 &s)

Public Attributes

- union {
  struct {
  Type **m00**
  Type **m01**
  Type **m02**
  Type **m03**
  Type **m10**
  Type **m11**
  Type **m12**
  Type **m13**
  Type **m20**
  Type **m21**
  Type **m22**
  Type **m23**
  }
  struct {
  Vector4 **row_0**
  Vector4 **row_1**
  Vector4 **row_2**
  }
  Vector4 **rows** [Rows]
  Type **m** [Size]
  };

Static Public Attributes

- static const Matrix4x3 zero

  default matrices
- static const Matrix4x3 **identity**

5.186.1 Detailed Description

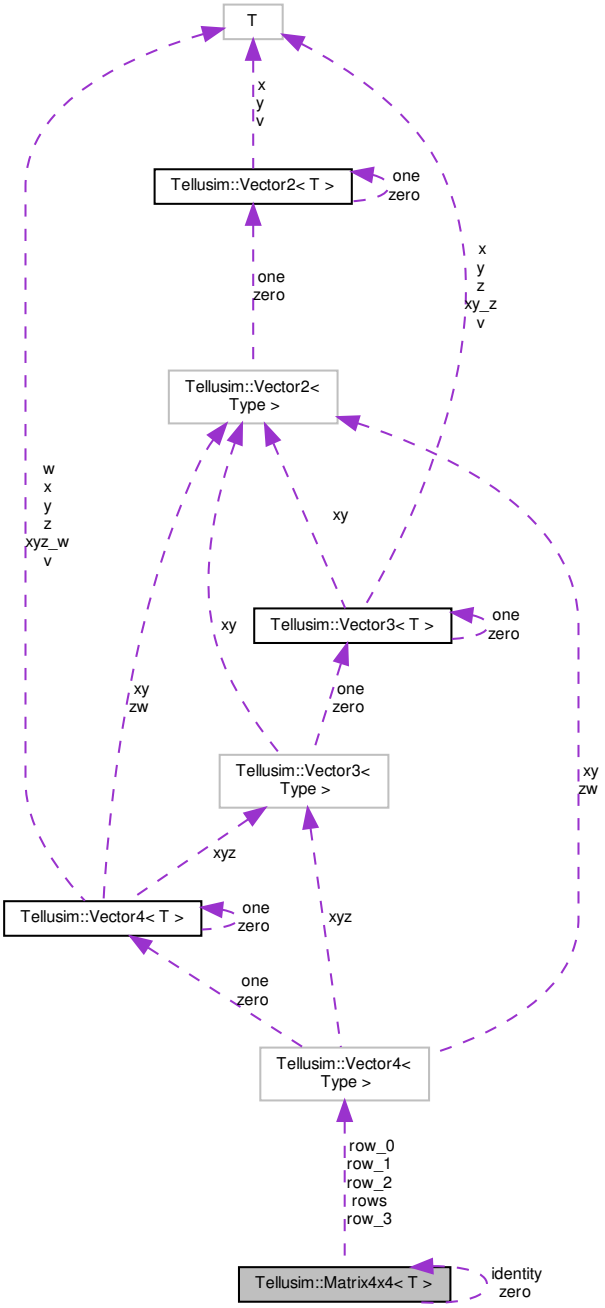
template<class T>
struct Tellusim::Matrix4x3< T >

Matrix4x3 class

5.187 Tellusim::Matrix4x4< T > Struct Template Reference

`#include <math/TellusimMatrix.h>`

Collaboration diagram for Tellusim::Matrix4x4< T >:



Public Types

- enum {
  Rows = 4,
  Columns = 4,
  Size = Columns ∗ Rows }
- using **Vector3** = Tellusim::Vector3< Type >

- using **Vector4** = Tellusim::Vector4< Type >
- using **Matrix4x3** = Tellusim::Matrix4x3< Type >
- using **Quaternion** = Tellusim::Quaternion< Type >

Public Member Functions

- **Matrix4x4** (const Matrix4x4 &m)
- **Matrix4x4** (const Vector4 &row_0, const Vector4 &row_1, const Vector4 &row_2, const Vector4 &row_3)
- **Matrix4x4** (Type m00, Type m01, Type m02, Type m03, Type m10, Type m11, Type m12, Type m13, Type m20, Type m21, Type m22, Type m23, Type m30, Type m31, Type m32, Type m33)
- **Matrix4x4** (const Matrix4x3 &m)
- **Matrix4x4** (const Quaternion &q)
- **Matrix4x4** (const Type *1 m, uint32_t size=Size, bool row_major=true)
- template<class CType >
  Matrix4x4 (const Tellusim::Matrix4x3< CType > &m)
- template<class CType >
  Matrix4x4 (const Tellusim::Matrix4x4< CType > &m)
- **Matrix4x4** (Type v)
- void set (const Vector4 &r0, const Vector4 &r1, const Vector4 &r2, const Vector4 &r3, bool row_major=true)

  update matrix data
- void **set** (const Type *1 m, uint32_t size=Size, bool row_major=true)
- void **get** (Type *1 m, uint32_t size=Size, bool row_major=true) const
- Matrix4x4 & operator*= (const Matrix4x4 &m1)

  matrix to matrix multiplication
- Matrix4x4 & operator+= (const Matrix4x4 &m)

  matrix to matrix operators
- Matrix4x4 & **operator-=** (const Matrix4x4 &m)
- void setZero ()

  zero matrix
- bool **isZero** () const
- void setIdentity ()

  identity matrix
- bool **isIdentity** () const
- void setOuter (const Vector4 &v0, const Vector4 &v1)

  outer product matrix
- void setScale (const Vector3 &s)

  scaling matrix
- void **setScale** (Type x, Type y, Type z)
- Vector3 **getScale** () const
- void setTranslate (const Vector3 &t)

  translation matrix
- void **setTranslate** (Type x, Type y, Type z)
- Vector3 **getTranslate** () const
- void setRotateX (Type angle)

  rotation matrix
- void **setRotateY** (Type angle)
- void **setRotateZ** (Type angle)
- void **setRotate** (const Vector3 &axis, Type angle)
- void **setRotate** (Type x, Type y, Type z, Type angle)
- Matrix4x4 **getRotate** () const
- void setLookAt (const Vector3 &from, const Vector3 &to, const Vector3 &up)

  look at matrix

- void setPlaceTo (const Vector3 &from, const Vector3 &to, const Vector3 &up)

  place to matrix
- void setCubeAt (const Vector3 &from, uint32_t face)

  cube at matrix
- void setBasis (const Vector3 &normal, const Vector3 &t)

  right-handed orthonormal basis
- Type getDeterminant () const

  matrix determinant
- void setComponents (const Vector3 &t, const Quaternion &r, const Vector3 &s)

  matrix composition
- void **getComponents** (Vector3 &t, Quaternion &r, Vector3 &s) const
- void setOrtho (Type left, Type right, Type bottom, Type top, Type znear, Type zfar)

  ortho matrix
- void setFrustum (Type left, Type right, Type bottom, Type top, Type znear, Type zfar, bool reverse=false)

  frustum matrix
- void setFrustum (Type left, Type right, Type bottom, Type top, Type znear, bool reverse=false)

  infinite frustum matrix
- void setPerspective (Type fov, Type aspect, Type znear, Type zfar, bool reverse=false)

  perspective matrix
- void setPerspective (Type fov, Type aspect, Type znear, bool reverse=false)

  infinite perspective matrix
- void setRow (uint32_t index, const Vector4 &r)

  matrix rows
- const Vector4 & **getRow** (uint32_t index) const
- Vector4 & **getRow** (uint32_t index)
- void setColumn (uint32_t index, const Vector4 &c)

  matrix columns
- Vector4 **getColumn** (uint32_t index) const
- const Vector4 & operator[ ] (uint32_t index) const

  matrix data
- Vector4 & **operator[ ]** (uint32_t index)

Static Public Member Functions

- static Matrix4x4 **outer** (const Vector4 &v0, const Vector4 &v1)
- static Matrix4x4 **scale** (const Vector3 &s)
- static Matrix4x4 **scale** (Type x, Type y, Type z)
- static Matrix4x4 **scale** (Type s)
- static Matrix4x4 **translate** (const Vector3 &t)
- static Matrix4x4 **translate** (Type x, Type y, Type z)
- static Matrix4x4 **rotateX** (Type angle)
- static Matrix4x4 **rotateY** (Type angle)
- static Matrix4x4 **rotateZ** (Type angle)
- static Matrix4x4 **rotate** (const Vector3 &axis, Type angle)
- static Matrix4x4 **rotate** (Type x, Type y, Type z, Type angle)
- static Matrix4x4 **lookAt** (const Vector3 &from, const Vector3 &to, const Vector3 &up)
- static Matrix4x4 **placeTo** (const Vector3 &from, const Vector3 &to, const Vector3 &up)
- static Matrix4x4 **cubeAt** (const Vector3 &from, uint32_t face)
- static Matrix4x4 **basis** (const Vector3 &normal)
- static Matrix4x4 **basis** (const Vector3 &normal, const Vector3 &t)
- static Matrix4x4 **compose** (const Vector3 &t, const Quaternion &r, const Vector3 &s)
- static Matrix4x4 **ortho** (Type left, Type right, Type bottom, Type top, Type znear, Type zfar)

- static Matrix4x4 **frustum** (Type left, Type right, Type bottom, Type top, Type znear, Type zfar, bool reverse=false)
- static Matrix4x4 **frustum** (Type left, Type right, Type bottom, Type top, Type znear, bool reverse=false)
- static Matrix4x4 **perspective** (Type fov, Type aspect, Type znear, Type zfar, bool reverse=false)
- static Matrix4x4 **perspective** (Type fov, Type aspect, Type znear, bool reverse=false)

Public Attributes

-
```
union {
  struct {
    Type m00
    Type m01
    Type m02
    Type m03
    Type m10
    Type m11
    Type m12
    Type m13
    Type m20
    Type m21
    Type m22
    Type m23
    Type m30
    Type m31
    Type m32
    Type m33
  }
  struct {
    Vector4 row_0
    Vector4 row_1
    Vector4 row_2
    Vector4 row_3
  }
  Vector4 rows [Rows]
  Type m [Size]
};
```

Static Public Attributes

- static const Matrix4x4 zero

  default matrices
- static const Matrix4x4 **identity**

5.187.1 Detailed Description

template<class T>
struct Tellusim::Matrix4x4< T >

Matrix4x4 class

5.188 Tellusim::MatrixNxM< Type, N, M > Struct Template Reference

`#include <math/TellusimNumerical.h>`

Public Types

- using **VectorN** = Tellusim::VectorN< Type, N >
- using **VectorM** = Tellusim::VectorN< Type, M >
- using **VectorNxM** = Tellusim::VectorN< Type, N *M >

Public Member Functions

- **MatrixNxM** (const MatrixNxM &matrix)
- **MatrixNxM** (uint32_t columns, uint32_t rows)
- **MatrixNxM** (const Type &value, uint32_t columns=N, uint32_t rows=M)
- **MatrixNxM** (const Type *matrix, uint32_t columns=N, uint32_t rows=M)
- **MatrixNxM** (const VectorNxM &vector, uint32_t columns=N, uint32_t rows=M)
- **MatrixNxM** (const InitializerList< Type > &list, uint32_t columns=N, uint32_t rows=M)
- **MatrixNxM** (const InitializerList< VectorN > &list, uint32_t rows=M)
- template<class CType >
  MatrixNxM (const MatrixNxM< CType, N, M > &matrix)
- void set (const Type &value, uint32_t columns=N, uint32_t rows=M)

  update matrix data
- void **set** (const Type *1 matrix, uint32_t columns=N, uint32_t rows=M)
- void **set** (const MatrixNxM &matrix)
- void **set** (const VectorNxM &vector, uint32_t columns=N, uint32_t rows=M)
- void **set** (const InitializerList< Type > &list, uint32_t columns=N, uint32_t rows=M)
- void **set** (const InitializerList< VectorN > &list, uint32_t rows=M)
- void **get** (Type *1 matrix, uint32_t columns=N, uint32_t rows=M)
- void setZero ()

  zero matrix
- void setIdentity ()

  identity matrix
- MatrixNxM & operator*= (const Type &value)

  matrix to scalar operators
- MatrixNxM & **operator/=** (const Type &value)
- MatrixNxM & operator= (const MatrixNxM &matrix)

  matrix to matrix operators
- void setRow (uint32_t index, const VectorN &r)

  matrix rows
- const VectorN & **getRow** (uint32_t index) const
- VectorN & **getRow** (uint32_t index)
- void setColumn (uint32_t index, const VectorM &c)

  matrix columns
- VectorM **getColumn** (uint32_t index) const
- const VectorN & operator[ ] (uint32_t index) const

  matrix data
- VectorN & **operator[ ]** (uint32_t index)

Public Attributes

- VectorN **rows** [M]
- uint32_t **Columns** = N
- uint32_t **Rows** = M

5.188.1 Detailed Description

template<class Type, uint32_t N, uint32_t M>
struct Tellusim::MatrixNxM< Type, N, M >

MatrixNxM class

5.189 Tellusim::Mesh Class Reference

`#include <format/TellusimMesh.h>`

Public Types

- enum Flags {
  FlagNone = 0,
  FlagEmbed = (1 << 0),
  Flag32Bit = (1 << 1) }

  Mesh flags.
- enum Basis {
  BasisUnknown = 0,
  BasisXUpRight,
  BasisYUpRight,
  BasisZUpRight,
  BasisXUpLeft,
  BasisYUpLeft,
  BasisZUpLeft,
  BasisZUpMaya,
  NumBases }

  mesh basis
- enum **Axis** {
  AxisUnknown = 0,
  AxisPX,
  AxisPY,
  AxisPZ,
  AxisNX,
  AxisNY,
  AxisNZ,
  NumAxes }

Public Member Functions

- void clear ()
  clear mesh
- bool isLoaded () const
  check mesh
- bool info (const char *name, Flags flags=FlagNone, Async *async=nullptr)
  info mesh
- bool **info** (const String &name, Flags flags=FlagNone, Async *async=nullptr)
- bool **info** (Stream &stream, Flags flags=FlagNone, Async *async=nullptr)
- bool **info** (const char *name, Async *async)
- bool **info** (const String &name, Async *async)
- bool **info** (Stream &stream, Async *async)
- bool load (const char *name, Flags flags=FlagNone, Async *async=nullptr)
  load mesh
- bool **load** (const String &name, Flags flags=FlagNone, Async *async=nullptr)
- bool **load** (Stream &stream, Flags flags=FlagNone, Async *async=nullptr)
- bool **load** (const char *name, Async *async)
- bool **load** (const String &name, Async *async)
- bool **load** (Stream &stream, Async *async)
- bool save (const char *name, Flags flags=FlagNone) const
  save mesh
- bool **save** (const String &name, Flags flags=FlagNone) const
- bool **save** (Stream &stream, Flags flags=FlagNone) const
- void setName (const char *name)
  mesh name
- String **getName** () const
- bool setBasis (Axis front, Axis right, Axis up)
  brep basis
- bool **setBasis** (Basis basis)
- Axis **getFrontAxis** () const
- Axis **getRightAxis** () const
- Axis **getUpAxis** () const
- Basis **getBasis** () const
- const char * **getFrontAxisName** () const
- const char * **getRightAxisName** () const
- const char * **getUpAxisName** () const
- String **getBasisName** () const
- void clearNodes ()
  mesh nodes
- void **reserveNodes** (uint32_t num_nodes)
- uint32_t **addNode** (MeshNode &node, bool check=true)
- bool **removeNode** (MeshNode &node)
- uint32_t **findNode** (const MeshNode &node) const
- uint32_t **findNode** (const char *name) const
- uint32_t **getNumNodes** () const
- const Array< MeshNode > **getNodes** () const
- Array< MeshNode > **getNodes** ()
- const MeshNode **getNode** (uint32_t index) const
- MeshNode **getNode** (uint32_t index)
- void createLocalTransforms (const Matrix4x3d &itransform=Matrix4x3d::identity)
- void **createGlobalTransforms** (const Matrix4x3d &transform=Matrix4x3d::identity)
- void clearGeometries ()

mesh geometries

- void **reserveGeometries** (uint32_t num_geometries)
- uint32_t **addGeometry** (MeshGeometry &geometry, bool check=true)
- uint32_t **addGeometry** (MeshGeometry &geometry, MeshNode &node, bool check=true)
- bool **removeGeometry** (MeshGeometry &geometry)
- bool **replaceGeometry** (MeshGeometry &old_geometry, MeshGeometry &geometry)
- uint32_t **findGeometry** (const MeshGeometry &geometry) const
- uint32_t **findGeometry** (const char *name) const
- uint32_t **getNumGeometries** () const
- const Array< MeshGeometry > **getGeometries** () const
- Array< MeshGeometry > **getGeometries** ()
- const MeshGeometry **getGeometry** (uint32_t index) const
- MeshGeometry **getGeometry** (uint32_t index)
- bool **hasGeometryIndices** (MeshIndices::Type type) const
- bool **hasGeometryAttribute** (MeshAttribute::Type type) const
- size_t **getNumGeometryPositions** () const
- size_t **getNumGeometryPrimitives** () const
- void clearAnimations ()

mesh animations

- void **reserveAnimations** (uint32_t num_animations)
- uint32_t **addAnimation** (MeshAnimation &animation, bool check=true)
- bool **removeAnimation** (MeshAnimation &animation)
- bool **replaceAnimation** (MeshAnimation &old_animation, MeshAnimation &animation)
- uint32_t **findAnimation** (const MeshAnimation &animation) const
- uint32_t **findAnimation** (const char *name) const
- uint32_t **getNumAnimations** () const
- const Array< MeshAnimation > **getAnimations** () const
- Array< MeshAnimation > **getAnimations** ()
- const MeshAnimation **getAnimation** (uint32_t index) const
- MeshAnimation **getAnimation** (uint32_t index)
- BoundBoxd getBoundBox () const

mesh bound box

- BoundBoxd **getBoundBox** (const MeshNode &node) const
- BoundBoxd **getBoundBox** (const MeshGeometry &geometry) const
- bool createBounds (bool force, Async *async=nullptr)
- bool **createBounds** (Async *async=nullptr)
- bool createBasis (bool force, Async *async=nullptr)
- bool **createBasis** (float32_t angle, bool force, Async *async=nullptr)
- bool **createBasis** (Async *async=nullptr)
- bool **createBasis** (float32_t angle, Async *async=nullptr)
- bool createNormals (bool force, Async *async=nullptr)
- bool **createNormals** (float32_t angle, bool force, Async *async=nullptr)
- bool **createNormals** (Async *async=nullptr)
- bool **createNormals** (float32_t angle, Async *async=nullptr)
- bool createTangents (bool force, Async *async=nullptr)
- bool **createTangents** (Async *async=nullptr)
- bool createIslands (uint32_t max_attributes, uint32_t max_primitives, bool force, Async *async=nullptr)
- bool **createIslands** (uint32_t max_attributes, uint32_t max_primitives, Async *async=nullptr)
- bool optimizeIndices (uint32_t cache, bool transparent, Async *async=nullptr)

optimize indices order for cache

- bool **optimizeIndices** (Async *async=nullptr)
- bool **optimizeIndices** (uint32_t cache, Async *async=nullptr)
- bool optimizeAttributes (Async *async=nullptr)

optimize attributes and make single indices.

- void optimizeMaterials ()

  optimize materials remove duplicates
- bool optimizeWinding (bool clockwise=false)
- void optimizeGeometries (float32_t threshold=1e-3f, uint32_t depth=16)
- void optimizeAnimations (float32_t threshold=1e-6f)
- bool optimizeOrder ()

  optimize node and geometry order
- void mergeGeometries ()

  merge geometries
- bool packAttributes (bool remove=true)

  pack attributes
- bool **unpackAttributes** (bool remove=true)
- bool setTransform (const Vector3d &scale)

  apply transform
- size_t getMemory () const

  memory usage

5.189.1 Detailed Description

The Mesh class provides a comprehensive interface for loading, saving, editing, and optimizing 3D mesh data. It supports operations on mesh components such as nodes, geometries, animations, and bounding boxes. The class enables loading from file names or streams with optional asynchronous processing. It offers methods to validate mesh structure, set coordinate system bases, and manage transformation hierarchies. The class includes optimization features such as index reordering, attribute unification, material deduplication, and transform simplification. Additionally, it supports mesh packing, attribute compression, and applying scale transformations, making it a versatile tool for preparing and managing complex 3D models efficiently.

5.189.2 Member Function Documentation

5.189.2.1 createLocalTransforms()

```
void Tellusim::Mesh::createLocalTransforms (
            const Matrix4x3d & itransform = Matrix4x3d::identity )
```

create node transformations

Parameters

itransform	Global to Local transformation matrix
transform	Local to Global transformation matrix

5.189.2.2 createBounds()

```
bool Tellusim::Mesh::createBounds (
            bool force,
            Async * async = nullptr )
```

create bounds

Parameters

force	Force bounding creation.

5.189.2.3 createBasis()

```
bool Tellusim::Mesh::createBasis (
            bool force,
            Async * async = nullptr )
```

create tangent basis

Parameters

force	Force basis creation.

5.189.2.4 createNormals()

```
bool Tellusim::Mesh::createNormals (
            bool force,
            Async * async = nullptr )
```

create normals

Parameters

force	Force normals creation.
angle	Smoothing angle in degrees.

5.189.2.5 createTangents()

```
bool Tellusim::Mesh::createTangents (
            bool force,
            Async * async = nullptr )
```

create tangents

Parameters

force	Force tangents creation.

5.189.2.6 createIslands()

```
bool Tellusim::Mesh::createIslands (
            uint32_t max_attributes,
            uint32_t max_primitives,
            bool force,
            Async * async = nullptr )
```

create islands

Parameters

max_attributes	Maximum number of attributes per island.
max_primitives	Maximum number of primitives per island.
force	Force islands creation.

5.189.2.7 optimizeWinding()

```
bool Tellusim::Mesh::optimizeWinding (
            bool clockwise = false )
```

optimize winding based on node transforms

Parameters

clockwise	Optimize for clockwise winding.

5.189.2.8 optimizeGeometries()

```
void Tellusim::Mesh::optimizeGeometries (
            float32_t threshold = 1e-3f,
            uint32_t depth = 16 )
```

optimize geometries remove duplicates

Parameters

threshold	Spatial compare threshold.
depth	Number of geometries to compare.

5.189.2.9 optimizeAnimations()

```
void Tellusim::Mesh::optimizeAnimations (
            float32_t threshold = 1e-6f )
```

optimize animation transforms

Parameters

threshold	Compare threshold.

5.190 Tellusim::MeshAnimation Class Reference

`#include <format/TellusimMesh.h>`

Public Member Functions

- **MeshAnimation** (const char *name=nullptr)
- **MeshAnimation** (Mesh &mesh, const char *name=nullptr)
- void clear ()

  clear animation
- uint32_t getIndex () const

  animation index
- void setName (const char *name)

  animation name
- String **getName** () const
- void setMesh (Mesh &mesh, bool check=true)

  animation mesh
- const Mesh **getMesh** () const
- Mesh **getMesh** ()
- float64_t getMinTime () const

  animation range
- float64_t **getMaxTime** () const
- void setNumTransforms (uint32_t num_transforms)

  animation transforms
- uint32_t **getNumTransforms** () const
- const Array< MeshTransform > **getTransforms** () const
- Array< MeshTransform > **getTransforms** ()
- const MeshTransform **getTransform** (uint32_t node) const
- MeshTransform **getTransform** (uint32_t node)
- void **setTransform** (float64_t time, uint32_t node, const Matrix4x3d &transform, float32_t threshold=1e-6f)
- void **setTranslate** (float64_t time, uint32_t node, const Vector3d &translate, float32_t threshold=1e-6f)
- void **setRotate** (float64_t time, uint32_t node, const Quaternionf &rotate, float32_t threshold=1e-6f)
- void **setScale** (float64_t time, uint32_t node, const Vector3f &scale, float32_t threshold=1e-6f)
- void **setMorph** (float64_t time, uint32_t node, const Vector4f &morph, float32_t threshold=1e-6f)
- void setTime (float64_t time, const Matrix4x3d &transform=Matrix4x3d::identity, bool loop=true, float64_t from=-Maxf32, float64_t to=Maxf32)

  animation transform
- void **setTime** (float64_t time, bool loop, float64_t from=-Maxf32, float64_t to=Maxf32)
- const Matrix4x3d & **getLocalTransform** (uint32_t node) const
- const Matrix4x3d & **getLocalTransform** (const MeshNode &node) const
- const Matrix4x3d & **getLocalTransform** (const MeshJoint &joint) const
- const Matrix4x3d & **getGlobalTransform** (uint32_t node) const
- const Matrix4x3d & **getGlobalTransform** (const MeshNode &node) const
- const Matrix4x3d & **getGlobalTransform** (const MeshJoint &joint) const
- const Vector4f & **getMorphTransform** (uint32_t node) const
- const Vector4f & **getMorphTransform** (const MeshNode &node) const

- BoundBoxd getBoundBox () const

  animation bound box
- BoundBoxd **getBoundBox** (const MeshNode &node) const
- BoundBoxd **getBoundBox** (const MeshGeometry &geometry) const
- BoundBoxd **getBoundBox** (const MeshGeometry &geometry, const Vector4f &morph) const
- void setTransform (const Vector3d &scale)

  apply transform
- void **setTransform** (const Matrix4x3d &transform)
- void optimizeTransforms (float32_t threshold=1e-6f)

  optimize animation
- size_t getMemory () const

  memory usage

Friends

- class **Mesh**

5.190.1 Detailed Description

The MeshAnimation class represents a collection of transformations applied to a mesh over time, typically used to animate 3D models. It allows for managing and retrieving a series of transformations for multiple nodes or joints within the mesh. The class provides methods to set and retrieve transformations at specific time intervals, including translation, rotation, scale, and morphing. It also supports optimization of transforms and the calculation of bounding boxes for the animated mesh. The MeshAnimation class is designed to efficiently handle complex animations, including looping and applying transformations across multiple mesh nodes or joints.

5.191 Tellusim::MeshAttachment Class Reference

```
#include <format/TellusimMesh.h>
```

Public Types

- enum Flags {
  FlagNone = 0,
  FlagBool = (1 << 0),
  FlagScalarf32 = (1 << 1),
  FlagVector4f = (1 << 2),
  FlagColor = (1 << 3),
  FlagName = (1 << 4) }

  Attachment flags.

Public Member Functions

- **MeshAttachment** (const char *name=nullptr)
- **MeshAttachment** (Type type, const char *name=nullptr)
- **MeshAttachment** (MeshNode &node, const char *name=nullptr)
- void clear ()

  clear attachment
- void setType (Type type)

  attachment type
- Type **getType** () const
- const char * **getTypeName** () const
- bool **isUnknown** () const
- bool **isLight** () const
- bool **isCamera** () const
- void setName (const char *name)

  attachment name
- String **getName** () const
- void setNode (MeshNode &node, bool check=true)

  attachment node
- const MeshNode **getNode** () const
- MeshNode **getNode** ()
- void setData (const char *data)

  attachment data
- void **setData** (const String &data)
- String **getData** () const
- void clearParameters ()

  attachment parameters
- bool **removeParameter** (const char *type)
- void **copyParameters** (const MeshAttachment &attachment)
- uint32_t **findParameter** (const char *type) const
- bool **hasParameter** (const char *type) const
- uint32_t **getNumParameters** () const
- String **getParameterType** (uint32_t index) const
- void addParameter (const char *type, bool value)

  add attachment parameters
- void **addParameter** (const char *type, float32_t value)
- void **addParameter** (const char *type, const Vector4f &vector)
- void **addParameter** (const char *type, const Color &color)
- void **addParameter** (const char *type, const char *name)
- void **addParameter** (const char *type, const String &name)
- Flags getParameterFlags (uint32_t index) const

  get attachment parameter by index
- bool **hasParameterFlag** (uint32_t index, Flags flags) const
- bool **hasParameterFlags** (uint32_t index, Flags flags) const
- bool **getParameterBool** (uint32_t index, bool value=false) const
- float32_t **getParameterScalarf32** (uint32_t index, float32_t value=0.0f) const
- const Vector4f & **getParameterVector4f** (uint32_t index, const Vector4f &vector=Vector4f::zero) const
- const Color & **getParameterColor** (uint32_t index, const Color &color=Color::white) const
- String **getParameterName** (uint32_t index, const String &name=String::null) const
- Flags getParameterFlags (const char *type) const

  get attachment parameter by type
- bool **hasParameterFlag** (const char *type, Flags flags) const
- bool **hasParameterFlags** (const char *type, Flags flags) const

- bool **getParameterBool** (const char *type, bool value=false) const
- float32_t **getParameterScalarf32** (const char *type, float32_t value=0.0f) const
- const Vector4f & **getParameterVector4f** (const char *type, const Vector4f &vector=Vector4f::zero) const
- const Color & **getParameterColor** (const char *type, const Color &color=Color::white) const
- String **getParameterName** (const char *type, const String &name=String::null) const
- void setTransform (const Vector3f &scale)

  attachment transform
- void **setTransform** (const Matrix4x3f &transform)
- const Matrix4x3f & **getTransform** () const
- int32_t compare (const MeshAttachment &attachment) const

  compare attachments
- size_t getMemory () const

  memory usage

Static Public Member Functions

- static const char * **getTypeName** (Type type)

Friends

- class **MeshNode**

5.191.1 Detailed Description

The MeshAttachment class represents an attachment that can be associated with a mesh node in a 3D scene. It supports various attachment types, including lights and cameras, allowing for flexible scene composition. The class provides methods for setting and retrieving the attachment type, name, associated node, and parameters. Each attachment can have custom parameters, such as boolean flags, scalar values, vectors, and colors, which can be added, accessed, and modified.

5.192 Tellusim::MeshAttribute Class Reference

```
#include <format/TellusimMesh.h>
```

Public Member Functions

- **MeshAttribute** (const char *name=nullptr, uint32_t index=0)
- **MeshAttribute** (Type type, Format format, const char *name=nullptr, uint32_t index=0)
- **MeshAttribute** (Type type, Format format, uint32_t size, const char *name=nullptr, uint32_t index=0)
- **MeshAttribute** (Type type, Format format, uint32_t size, uint32_t index)
- void clear ()

  clear attributes
- void setName (const char *name)

  attribute name
- String **getName** () const
- void create (Type type, Format format, uint32_t size=0)

  create attribute
- Type getType () const

attribute type

- const char * **getTypeName** () const
- bool **isUnknown** () const
- bool **isPosition** () const
- bool **isBasis** () const
- bool **isNormal** () const
- bool **isTangent** () const
- bool **isBinormal** () const
- bool **isSpatial** () const
- bool **isNormalized** () const
- bool **isTexCoord** () const
- bool **isWeights** () const
- bool **isJoints** () const
- bool **isColor** () const
- bool **isVertex** () const
- bool **isCrease** () const
- Format getFormat () const

attribute format

- const char * **getFormatName** () const
- uint32_t **getComponents** () const
- bool **isPacked** () const
- void setIndex (uint32_t index)

attribute index

- uint32_t **getIndex** () const
- void setIndices (MeshIndices &indices)

attribute indices

- const MeshIndices **getIndices** () const
- MeshIndices **getIndices** ()
- void setGeometry (MeshGeometry &geometry, bool check=true)

attribute geometry

- const MeshGeometry **getGeometry** () const
- MeshGeometry **getGeometry** ()
- void setSize (uint32_t size, bool discard=true, bool clear=false)

attribute size

- uint32_t **getSize** () const
- uint32_t **getStride** () const
- size_t **getBytes** () const
- void setData (const void *src, uint32_t size=0, uint32_t stride=0)

attribute data

- void **setData** (const void *src, const Array< uint32_t > &indices, uint32_t stride=0)
- void **getData** (void *dest, uint32_t size=0, uint32_t stride=0) const
- void **getData** (void *dest, const MeshIndices &indices, uint32_t stride=0) const
- void **getData** (void *dest, const Array< uint32_t > &indices, uint32_t stride=0) const
- const void * **getData** () const
- void * **getData** ()
- template<class Type >
  void set (const Type &value)

attribute values

- template<class Type >
  void **set** (uint32_t index, const Type &value)
- template<class Type >
  const Type & **get** (uint32_t index) const
- template<class Type >
  Type & **get** (uint32_t index)

- void setValue (uint32_t index, const void *src, size_t size)

  attribute value
- void **getValue** (uint32_t index, void *dest, size_t size) const
- const void * getPtr (uint32_t index) const

  attribute pointers
- void * **getPtr** (uint32_t index)
- int32_t compare (const MeshAttribute &attribute, const Matrix4x3f &transform=Matrix4x3f::identity, float32_t threshold=1e-6f, bool spatial=true) const

  compare attributes
- void addAttribute (const MeshAttribute &attribute)

  add attribute
- bool setTransform (const Matrix4x3f &transform)

  apply transform
- bool morphAttribute (const MeshAttribute &attribute, float32_t k)

  morph attribute
- bool packAttributes (const MeshAttribute &attribute_0, const MeshAttribute &attribute_1, Format format)

  pack attributes
- bool **unpackAttributes** (MeshAttribute &attribute_0, MeshAttribute &attribute_1) const
- MeshAttribute optimizeAttribute (MeshIndices &indices) const

  optimize attribute by removing duplicates
- MeshAttribute toDirect (const MeshIndices &indices) const

  convert attribute to direct
- MeshAttribute toFormat (Format format) const

  convert attribute to format
- MeshAttribute toType (Type type) const

  convert attribute to type
- Matrix4x3f getCovarianceMatrix () const

  covariance matrix
- Matrix4x3f getMinTransform () const

  minimal bound transform
- BoundBoxf getBoundBox () const

  attribute bound box
- BoundSpheref getBoundSphere () const

  attribute bound sphere
- size_t getMemory () const

  memory usage

Static Public Member Functions

- static const char * **getTypeName** (Type type)

Friends

- class **MeshGeometry**

5.192.1 Detailed Description

The MeshAttribute class represents per-vertex attribute data in a mesh, such as positions, normals, texture coordinates, colors, and joint weights. Each attribute has a type, format, name, index (used for distinguishing between multiple attributes of the same type), and size. The class allows setting and retrieving attribute values, either directly or via associated index data (MeshIndices), and supports various utilities for transformation, morphing, packing/unpacking, and optimization.

5.193 Tellusim::MeshGeometry Class Reference

`#include <format/TellusimMesh.h>`

Public Member Functions

- **MeshGeometry** (const char *name=nullptr)
- **MeshGeometry** (Mesh &mesh, const char *name=nullptr)
- void clear ()

  clear geometry
- void setName (const char *name)

  geometry name
- String **getName** () const
- uint32_t getIndex () const

  geometry index
- void setMesh (Mesh &mesh, bool check=true)

  geometry mesh
- const Mesh **getMesh** () const
- Mesh **getMesh** ()
- uint32_t setParent0 (MeshGeometry &parent, bool check=true)

  geometry parent
- uint32_t **setParent1** (MeshGeometry &parent, bool check=true)
- const MeshGeometry **getParent0** () const
- const MeshGeometry **getParent1** () const
- MeshGeometry **getParent0** ()
- MeshGeometry **getParent1** ()
- bool **isRoot** () const
- void reserveChildren (uint32_t num_children)

  geometry children
- uint32_t **addChild0** (MeshGeometry &child, bool check=true)
- uint32_t **addChild1** (MeshGeometry &child, bool check=true)
- bool **removeChild** (MeshGeometry &child)
- void **releaseChildren** ()
- uint32_t **findChild** (const MeshGeometry &child) const
- uint32_t **getNumChildren** () const
- const Array< MeshGeometry > **getChildren** () const
- Array< MeshGeometry > **getChildren** ()
- const MeshGeometry **getChild** (uint32_t index) const
- MeshGeometry **getChild** (uint32_t index)
- void clearIndices ()

  geometry indices
- void **reserveIndices** (uint32_t num_indices)
- uint32_t **addIndices** (MeshIndices &indices, bool check=true)
- bool **removeIndices** (MeshIndices &indices)
- bool **replaceIndices** (MeshIndices &old_indices, MeshIndices &indices)
- uint32_t **findIndices** (const MeshIndices &indices) const
- uint32_t **findIndices** (MeshIndices::Type type) const
- bool **hasIndices** (MeshIndices::Type type) const
- bool **hasSolidIndices** () const
- uint32_t **getNumIndices** (MeshIndices::Type type) const
- const MeshIndices **getIndices** (MeshIndices::Type type) const
- MeshIndices **getIndices** (MeshIndices::Type type)

- uint32_t **getNumIndices** () const
- const Array< MeshIndices > **getIndices** () const
- Array< MeshIndices > **getIndices** ()
- const MeshIndices **getIndices** (uint32_t index) const
- MeshIndices **getIndices** (uint32_t index)
- void clearAttributes ()

  geometry attributes
- void **reserveAttributes** (uint32_t num_attributes)
- uint32_t **addAttribute** (MeshAttribute &attribute, bool check=true)
- uint32_t **addAttribute** (MeshAttribute &attribute, MeshIndices &indices, bool check=true)
- bool **removeAttribute** (MeshAttribute &attribute)
- bool **replaceAttribute** (MeshAttribute &old_attribute, MeshAttribute &attribute)
- bool **replaceAttributeIndices** (const MeshIndices &old_indices, MeshIndices &indices)
- uint32_t **findAttribute** (MeshAttribute::Type type, Format format, uint32_t index=0) const
- uint32_t **findAttribute** (MeshAttribute::Type type, uint32_t index=0) const
- uint32_t **findAttribute** (const char *name, uint32_t index=0) const
- uint32_t **findAttribute** (const MeshAttribute &attribute) const
- bool **hasAttribute** (MeshAttribute::Type type, Format format, uint32_t index=0) const
- bool **hasAttribute** (MeshAttribute::Type type, uint32_t index=0) const
- bool **hasAttribute** (const char *name, uint32_t index=0) const
- bool **hasAttribute** (const MeshAttribute &attribute) const
- uint32_t **getNumAttributes** (MeshAttribute::Type type) const
- uint32_t **getNumAttributes** (const MeshIndices &indices) const
- const MeshAttribute **getAttribute** (MeshAttribute::Type type, uint32_t index=0) const
- MeshAttribute **getAttribute** (MeshAttribute::Type type, uint32_t index=0)
- uint32_t **getNumAttributes** () const
- const Array< MeshAttribute > **getAttributes** () const
- Array< MeshAttribute > **getAttributes** ()
- const MeshAttribute **getAttribute** (uint32_t index) const
- MeshAttribute **getAttribute** (uint32_t index)
- void clearJoints ()

  geometry joints
- void **reserveJoints** (uint32_t num_joints)
- uint32_t **addJoint** (MeshJoint &joint, bool check=true)
- uint32_t **addJoint** (MeshJoint &joint, MeshNode &node, bool check=true)
- bool **removeJoint** (MeshJoint &joint)
- bool **replaceJoint** (MeshJoint &old_joint, MeshJoint &joint)
- uint32_t **findJoint** (const MeshJoint &joint) const
- uint32_t **findJoint** (const MeshNode &node) const
- uint32_t **findJoint** (const char *name) const
- uint32_t **getNumJoints** () const
- const Array< MeshJoint > **getJoints** () const
- Array< MeshJoint > **getJoints** ()
- const MeshJoint **getJoint** (uint32_t index) const
- MeshJoint **getJoint** (uint32_t index)
- void clearMaterials ()

  geometry materials
- void **reserveMaterials** (uint32_t num_materials)
- uint32_t **addMaterial** (MeshMaterial &material, bool check=true)
- uint32_t **addMaterial** (MeshMaterial &material, MeshIndices &indices, bool check=true)
- bool **removeMaterial** (MeshMaterial &material)
- bool **replaceMaterial** (MeshMaterial &old_material, MeshMaterial &material)
- uint32_t **findMaterial** (const MeshMaterial &material) const
- uint32_t **findMaterial** (const char *name) const

- uint32_t **getNumMaterials** () const
- const Array< MeshMaterial > **getMaterials** () const
- Array< MeshMaterial > **getMaterials** ()
- const MeshMaterial **getMaterial** (uint32_t index) const
- MeshMaterial **getMaterial** (uint32_t index)
- void setBoundBox (const BoundBoxf &box)

  geometry bound box
- const BoundBoxf & **getBoundBox** () const
- void setBoundSphere (const BoundSpheref &sphere)

  geometry bound sphere
- const BoundSpheref & **getBoundSphere** () const
- bool setTransform (const Vector3f &scale)

  geometry transform
- bool **setTransform** (const Matrix4x3f &transform, bool apply=false)
- const Matrix4x3f & **getTransform** () const
- void setJointITransform (const Matrix4x3f &itransform)

  geometry inverse joint transform
- const Matrix4x3f & **getJointITransform** () const
- void setMinVisibility (float32_t distance)

  visibility range
- void **setMaxVisibility** (float32_t distance)
- void **setVisibilityRange** (float32_t min, float32_t max)
- float32_t **getMinVisibility** () const
- float32_t **getMaxVisibility** () const
- bool **hasVisibilityRange** () const
- void setVisibilityError (float32_t error)

  visibility error
- float32_t **getVisibilityError** () const
- bool createBounds (bool force=false, uint32_t position=Maxu32)
- uint32_t createBasis (bool force=false, uint32_t position=Maxu32, uint32_t normal=Maxu32, uint32_t tangent=Maxu32, bool append=false)
- uint32_t **createBasis** (float32_t angle, bool force=false, uint32_t position=Maxu32, uint32_t normal=Maxu32, uint32_t tangent=Maxu32, bool append=false)
- uint32_t createNormals (bool force=false, uint32_t position=Maxu32, bool append=false)
- uint32_t **createNormals** (float32_t angle, bool force=false, uint32_t position=Maxu32, bool append=false)
- uint32_t createTangents (bool force=false, uint32_t position=Maxu32, uint32_t normal=Maxu32, uint32_t texcoord=Maxu32, bool append=false)
- uint32_t createIslands (uint32_t max_attributes, uint32_t max_primitives, bool force=false, uint32_t index=Maxu32, uint32_t position=Maxu32, bool append=false)
- bool optimizeIndices (uint32_t cache=32, bool transparent=false, uint32_t index=Maxu32, uint32_t position=Maxu32)
- bool optimizeAttributes (uint32_t material=Maxu32)
- void optimizeMaterials ()

  optimize materials remove duplicates
- bool packAttributes (bool remove=true)

  pack attributes (morph targets, texture coordinates and vertex colors)
- bool **unpackAttributes** (bool remove=true)
- int32_t compare (const MeshGeometry &geometry, const Matrix4x3f &transform=Matrix4x3f::identity, float32_t threshold=1e-6f, bool spatial=true) const

  compare geometries
- bool isOptimized () const

  optimized geometry flag (if geometry contains single indices).
- bool validate () const

  validate geometry
- size_t getMemory () const

  memory usage

Friends

- class **Mesh**

5.193.1 Detailed Description

The MeshGeometry class represents the geometric structure of a 3D mesh, including its vertices, indices, attributes, joints, and materials. It allows for defining and manipulating the geometry hierarchical relationships, such as setting parent-child relationships between geometries, with support for two parent geometries to create seamless Level of Detail (LOD) transitions. This structure facilitates smooth blending of multiple geometric details based on distance or camera view, ensuring efficient rendering by dynamically adjusting the complexity of the mesh. The class provides methods for managing attributes like positions, normals, tangents, and texcoords, along with supporting operations for creating normals, tangents, and tangent basis. It also includes functionality for creating, optimizing, and managing materials, bounds, and visibility ranges.

5.193.2 Member Function Documentation

5.193.2.1 createBounds()

```
bool Tellusim::MeshGeometry::createBounds (
             bool force = false,
             uint32_t position = Maxu32 )
```

create bounds

Parameters

force	Force bounding creation.
position	Position attribute index.

5.193.2.2 createBasis()

```
uint32_t Tellusim::MeshGeometry::createBasis (
             bool force = false,
             uint32_t position = Maxu32,
             uint32_t normal = Maxu32,
             uint32_t tangent = Maxu32,
             bool append = false )
```

create tangent basis

Parameters

force	Force basis creation.
position	Position attribute index.
normal	Normal attribute index.
tangent	Tangent attribute index.
append	Append new basis attribute.

Returns

Basis attribute index.

5.193.2.3 createNormals()

```
uint32_t Tellusim::MeshGeometry::createNormals (
             bool force = false,
             uint32_t position = Maxu32,
             bool append = false )
```

create normals

Parameters

force	Force normals creation.
position	Position attribute index.
angle	Smoothing angle in degrees.
append	Append new normal attribute.

Returns

Normal attribute index.

5.193.2.4 createTangents()

```
uint32_t Tellusim::MeshGeometry::createTangents (
             bool force = false,
             uint32_t position = Maxu32,
             uint32_t normal = Maxu32,
             uint32_t texcoord = Maxu32,
             bool append = false )
```

create tangents

Parameters

force	Force tangents creation.
position	Position attribute index.
normal	Normal attribute index.
texcoord	TexCoord attribute index.
append	Append new tangent attribute.

Returns

Tangent attribute index.

5.193.2.5 createIslands()

```
uint32_t Tellusim::MeshGeometry::createIslands (
             uint32_t max_attributes,
             uint32_t max_primitives,
             bool force = false,
             uint32_t index = Maxu32,
             uint32_t position = Maxu32,
             bool append = false )
```

create islands

Parameters

max_attributes	Maximum number of attributes per island.
max_primitives	Maximum number of primitives per island.
index	Indices index to update.
position	Position attribute index.
append	Append new island indices.

Returns

Island indices index.

5.193.2.6 optimizeIndices()

```
bool Tellusim::MeshGeometry::optimizeIndices (
             uint32_t cache = 32,
             bool transparent = false,
             uint32_t index = Maxu32,
             uint32_t position = Maxu32 )
```

optimize indices

Parameters

cache	Vertex cache size.
transparent	Optimize for transparency.
index	Indices index to optimize.
position	Position attribute index.

5.193.2.7 optimizeAttributes()

```
bool Tellusim::MeshGeometry::optimizeAttributes (
             uint32_t material = Maxu32 )
```

optimize attributes and make single indices.

Parameters

material	material index.

5.194 Tellusim::MeshIndices Class Reference

```
#include <format/TellusimMesh.h>
```

Public Member Functions

- **MeshIndices** (const char *name=nullptr)
- **MeshIndices** (Type type, Format format, const char *name=nullptr)
- **MeshIndices** (Type type, Format format, uint32_t size, const char *name=nullptr)
- void clear ()

  clear indices
- void setName (const char *name)

  indices name
- String **getName** () const
- void create (Type type, Format format, uint32_t size=0)

  create indices
- Type getType () const

  indices type
- const char * **getTypeName** () const
- bool **isUnknown** () const
- bool **isPoint** () const
- bool **isLine** () const
- bool **isTriangle** () const
- bool **isQuadrilateral** () const
- bool **isTetrahedron** () const
- bool **isPrimitive** () const
- bool **isSolid** () const
- bool **isVolume** () const
- bool **isMaterial** () const
- bool **isGroup** () const
- bool **isJoint** () const
- bool **isEdge** () const
- uint32_t **getPrimitiveSize** () const
- Format getFormat () const

  indices format
- const char * **getFormatName** () const
- void setGeometry (MeshGeometry &geometry, bool check=true)

  indices geometry
- const MeshGeometry **getGeometry** () const
- MeshGeometry **getGeometry** ()
- void setSize (uint32_t size, bool discard=true, bool clear=false)

  indices size
- uint32_t **getSize** () const
- uint32_t **getStride** () const
- size_t **getBytes** () const
- void setData (uint32_t value, uint32_t size=0, uint32_t offset=0)

indices data

- void **setData** (const void *src, Format format=FormatUnknown, uint32_t size=0, uint32_t repeat=1)
- void **getData** (void *dest, Format format=FormatUnknown, uint32_t size=0, uint32_t repeat=1) const
- const void * **getData** () const
- void * **getData** ()
- void set (uint32_t index, uint32_t value)

  indices values
- void **set** (uint32_t index, uint32_t value_0, uint32_t value_1)
- void **set** (uint32_t index, uint32_t value_0, uint32_t value_1, uint32_t value_2)
- void **set** (uint32_t index, uint32_t value_0, uint32_t value_1, uint32_t value_2, uint32_t value_3)
- uint32_t **get** (uint32_t index) const
- void **get** (uint32_t index, uint32_t &value_0, uint32_t &value_1) const
- void **get** (uint32_t index, uint32_t &value_0, uint32_t &value_1, uint32_t &value_2) const
- void **get** (uint32_t index, uint32_t &value_0, uint32_t &value_1, uint32_t &value_2, uint32_t &value_3) const
- const void * getPtr (uint32_t index) const

  indices pointers
- void * **getPtr** (uint32_t index)
- bool isDirect () const

  direct indices flag
- bool isUniform () const

  uniform indices flag
- uint32_t getMinIndex () const

  indices range
- uint32_t **getMaxIndex** () const
- int32_t compare (const MeshIndices &indices) const

  compare indices
- void addIndices (const MeshIndices &indices, uint32_t offset, bool expand=false)

  add indices
- MeshIndices toFormat (Format format) const

  convert indices to format
- MeshIndices toType (Type type) const

  convert indices to type
- MeshIndices **toType** (Type type, const MeshAttribute &position_attribute) const
- size_t getMemory () const

  memory usage

Static Public Member Functions

- static const char * **getTypeName** (Type type)

Friends

- class **MeshGeometry**

5.194.1 Detailed Description

The MeshIndices class encapsulates index data used by mesh geometries to define how vertices are connected into primitives such as points, lines, triangles, and more complex topologies like quadrilaterals and tetrahedrons. Each MeshIndices instance has a specific type and format, allowing for flexible representation of both renderable geometry and auxiliary information like materials, groups, joints, and edges.

5.195 Tellusim::MeshJoint Class Reference

`#include <format/TellusimMesh.h>`

Public Member Functions

- **MeshJoint** (const char *name=nullptr)
- **MeshJoint** (MeshGeometry &geometry, const char *name=nullptr)
- void clear ()
  clear joint
- void setName (const char *name)
  joint name
- String **getName** () const
- void setNode (MeshNode &node)
  joint node
- const MeshNode **getNode** () const
- MeshNode **getNode** ()
- uint32_t **getNodeIndex** () const
- const Matrix4x3d & **getLocalTransform** () const
- const Matrix4x3d & **getGlobalTransform** () const
- void setIndices (MeshIndices &indices)
  joint indices
- const MeshIndices **getIndices** () const
- MeshIndices **getIndices** ()
- void setGeometry (MeshGeometry &geometry, bool check=true)
  joint geometry
- const MeshGeometry **getGeometry** () const
- MeshGeometry **getGeometry** ()
- void setBoundBox (const BoundBoxf &box)
  joint bound box
- const BoundBoxf & **getBoundBox** () const
- void setBoundSphere (const BoundSpheref &sphere)
  joint bound sphere
- const BoundSpheref & **getBoundSphere** ()
- void setITransform (const Matrix4x3f &itransform)
  inverse joint transform
- const Matrix4x3f & **getITransform** () const
- int32_t compare (const MeshJoint &joint) const
  compare joints
- size_t getMemory () const
  memory usage

Friends

- class **MeshGeometry**

5.195.1 Detailed Description

The MeshJoint class represents a skeletal joint within a mesh geometry, used for skeletal animation and skinning. It encapsulates joint metadata, including its name, associated node, transformation matrices (local/global), inverse bind transform, and optional bounding volumes.

5.196 Tellusim::MeshModel::Meshlet Struct Reference

Public Attributes

- uint32_t **num_primitives**
- uint32_t **num_vertices**
- uint32_t **base_index**
- uint32_t **base_vertex**
- float32_t **bound_sphere** [4]
- float32_t **normal_angle** [4]

5.197 Tellusim::MeshMaterial Class Reference

`#include <format/TellusimMesh.h>`

Public Types

- enum Flags {
  FlagNone = 0,
  FlagBool = (1 << 0),
  FlagScalarf32 = (1 << 1),
  FlagVector4f = (1 << 2),
  FlagMatrix3x2f = (1 << 3),
  FlagColor = (1 << 4),
  FlagName = (1 << 5),
  FlagLayout = (1 << 6),
  FlagBlob = (1 << 7),
  FlagImage = (1 << 8),
  FlagTexture = (FlagName | FlagBlob | FlagImage) }

  Material flags.

Public Member Functions

- **MeshMaterial** (const char *name=nullptr)
- **MeshMaterial** (MeshGeometry &geometry, const char *name=nullptr)
- void clear ()

  clear material
- void setName (const char *name)

  material name
- String **getName** () const
- uint32_t getIndex () const

  material index
- void setIndices (MeshIndices &indices)

  material indices
- const MeshIndices **getIndices** () const
- MeshIndices **getIndices** ()
- void setGeometry (MeshGeometry &geometry, bool check=true)

  material geometry
- const MeshGeometry **getGeometry** () const
- MeshGeometry **getGeometry** ()
- void setData (const char *data)

material data

- void **setData** (const String &data)
- String **getData** () const
- void clearParameters ()

material parameters

- bool **removeParameter** (const char *type)
- void **copyParameters** (const MeshMaterial &material)
- uint32_t **findParameter** (const char *type) const
- bool **hasParameter** (const char *type) const
- uint32_t **getNumParameters** () const
- String **getParameterType** (uint32_t index) const
- void addParameter (const char *type, bool value)

add material parameters

- void **addParameter** (const char *type, float32_t value)
- void **addParameter** (const char *type, const Vector4f &value)
- void **addParameter** (const char *type, const Matrix3x2f &value)
- void **addParameter** (const char *type, const Color &color)
- void **addParameter** (const char *type, const char *name, const char *layout=nullptr)
- void **addParameter** (const char *type, const String &name, const char *layout=nullptr)
- void **addParameter** (const char *type, const Image &image, const char *layout=nullptr)
- void **addParameter** (const char *type, Blob &blob, const char *layout=nullptr)
- Flags getParameterFlags (uint32_t index) const

get material parameter by index

- bool **hasParameterFlag** (uint32_t index, Flags flags) const
- bool **hasParameterFlags** (uint32_t index, Flags flags) const
- bool **getParameterBool** (uint32_t index, bool value=false) const
- float32_t **getParameterScalarf32** (uint32_t index, float32_t value=0.0f) const
- const Vector4f & **getParameterVector4f** (uint32_t index, const Vector4f &vector=Vector4f::zero) const
- const Matrix3x2f & **getParameterMatrix3x2f** (uint32_t index, const Matrix3x2f &matrix=Matrix3x2f::identity) const
- const Color & **getParameterColor** (uint32_t index, const Color &color=Color::white) const
- String **getParameterName** (uint32_t index, const String &name=String::null) const
- String **getParameterLayout** (uint32_t index, const String &layout=String::null) const
- const Image **getParameterImage** (uint32_t index) const
- Blob **getParameterBlob** (uint32_t index) const
- Image **getParameterImage** (uint32_t index)
- Blob **getParameterBlob** (uint32_t index)
- Flags getParameterFlags (const char *type) const

get material parameter by type

- bool **hasParameterFlag** (const char *type, Flags flags) const
- bool **hasParameterFlags** (const char *type, Flags flags) const
- bool **getParameterBool** (const char *type, bool value=false) const
- float32_t **getParameterScalarf32** (const char *type, float32_t value=0.0f) const
- const Vector4f & **getParameterVector4f** (const char *type, const Vector4f &vector=Vector4f::zero) const
- const Matrix3x2f & **getParameterMatrix3x2f** (const char *type, const Matrix3x2f &matrix=Matrix3x2f::identity) const
- const Color & **getParameterColor** (const char *type, const Color &color=Color::white) const
- String **getParameterName** (const char *type, const String &name=String::null) const
- String **getParameterLayout** (const char *type, const String &layout=String::null) const
- int32_t compare (const MeshMaterial &material) const

compare materials

- size_t getMemory () const

memory usage

Friends

- class **MeshGeometry**

5.197.1 Detailed Description

The MeshMaterial class represents a material used in a MeshGeometry object, supporting a flexible system of parameters for defining surface properties such as textures, colors, scalars, vectors, and transformation matrices. Materials can be named, assigned to geometries, and include multiple parameters identified by type strings. These parameters can store booleans, floats, vectors, matrices, colors, strings, images, and binary blobs.

5.198 Tellusim::MeshModel Class Reference

`#include <graphics/TellusimMeshModel.h>`

Classes

- struct Meshlet

Public Types

- enum Flags {
  FlagNone = 0,
  FlagDirect = (1 << 0),
  FlagVerbose = (1 << 1),
  FlagOptimize = (1 << 2),
  FlagMaterials = (1 << 3),
  FlagIndices10 = (1 << 4),
  FlagIndices16 = (1 << 5),
  FlagIndices32 = (1 << 6),
  FlagMeshlet64x84 = (1 << 7),
  FlagMeshlet64x126 = (1 << 8),
  FlagMeshlet96x169 = (1 << 9),
  FlagMeshlet128x212 = (1 << 10),
  FlagBufferWrite = (1 << 11),
  FlagBufferSource = (1 << 12),
  FlagBufferStorage = (1 << 13),
  FlagBufferTracing = (1 << 14),
  FlagBufferAddress = (1 << 15),
  FlagBufferTexel = (1 << 16),
  FlagMeshlets = (FlagMeshlet64x84 | FlagMeshlet64x126 | FlagMeshlet96x169 | FlagMeshlet128x212),
  DefaultFlags = (FlagVerbose | FlagMaterials),
  NumFlags = 17 }

  Model flags.
- using CreateCallback = Function< bool(const void *src, size_t size, bool owner)>

  create buffer callbacks

Public Member Functions

- void clear ()
  clear model
- bool isCreated () const
  check model
- Flags getFlags () const
  model flags
- bool **hasFlag** (Flags flags) const
- bool **hasFlags** (Flags flags) const
- bool load (const Device &device, const Pipeline &pipeline, const char *name, Flags flags=DefaultFlags, Async *async=nullptr)
  load model
- bool **load** (const Device &device, const Pipeline &pipeline, Stream &stream, Flags flags=DefaultFlags, Async *async=nullptr)
- bool create (const Device &device, const Pipeline &pipeline, const char *name, Flags flags=DefaultFlags)
  create model
- bool **create** (const Device &device, const Pipeline &pipeline, const Mesh &mesh, Flags flags=DefaultFlags)
- bool **create** (const Device &device, const Pipeline &pipeline, const MeshGeometry &geometry, Flags flags=DefaultFlags)
- bool **create** (const Device &device, const Pipeline &pipeline, const Array< MeshGeometry > &geometries, Flags flags=DefaultFlags)
- void **setVertexBufferCallback** (const CreateCallback &func)
- void **setIndexBufferCallback** (const CreateCallback &func)
- void **setMeshBufferCallback** (const CreateCallback &func)
- void setBuffers (Command &command, uint32_t index=0, const Pipeline *pipeline=nullptr) const
  set model buffers
- void draw (Command &command) const
  draw model
- void **draw** (Command &command, uint32_t geometry) const
- void **draw** (Command &command, uint32_t geometry, uint32_t material) const
- void drawInstanced (Command &command, uint32_t geometry, uint32_t num_instances, uint32_t base_instance=0) const
  draw instanced model
- void **drawInstanced** (Command &command, uint32_t geometry, uint32_t material, uint32_t num_instances, uint32_t base_instance) const
- uint32_t getNumVertices () const
  vertices buffer
- uint32_t **getNumVertexBuffers** () const
- uint32_t **getVertexBufferStride** (uint32_t index) const
- size_t **getVertexBufferOffset** (uint32_t index) const
- Buffer **getVertexBuffer** () const
- uint32_t getNumIndices () const
  indices buffer
- Format **getIndexFormat** () const
- Buffer **getIndexBuffer** () const
- uint32_t getNumMeshlets () const
  meshlets buffer
- Buffer **getMeshletBuffer** () const
- uint32_t getNumGeometries () const
  geometries
- uint32_t **getNumGeometryIndices** (uint32_t geometry) const
- uint32_t **getNumGeometryVertices** (uint32_t geometry) const

- uint32_t **getNumGeometryMeshlets** (uint32_t geometry) const
- uint32_t **getGeometryBaseIndex** (uint32_t geometry) const
- uint32_t **getGeometryBaseVertex** (uint32_t geometry) const
- uint32_t **getGeometryBaseMeshlet** (uint32_t geometry) const
- uint32_t getNumMaterials (uint32_t geometry) const

  geometry materials
- uint32_t **getNumMaterialIndices** (uint32_t geometry, uint32_t material) const
- uint32_t **getNumMaterialVertices** (uint32_t geometry, uint32_t material) const
- uint32_t **getNumMaterialMeshlets** (uint32_t geometry, uint32_t material) const
- uint32_t **getMaterialBaseIndex** (uint32_t geometry, uint32_t material) const
- uint32_t **getMaterialBaseVertex** (uint32_t geometry, uint32_t material) const
- uint32_t **getMaterialBaseMeshlet** (uint32_t geometry, uint32_t material) const
- size_t getMemory () const

  memory usage

Protected Member Functions

- virtual bool create_vertex_buffer (const Device &device, const void *src, size_t size, bool owner)
- virtual bool create_index_buffer (const Device &device, const void *src, size_t size, bool owner)
- virtual bool create_meshlet_buffer (const Device &device, const void *src, size_t size, bool owner)

5.198.1 Detailed Description

The MeshModel class represents a 3D model used in graphics rendering, offering various methods for creating, loading, and managing model data. It provides support for different model flags, such as optimizing indices, creating materials, and specifying buffer types. The class facilitates the setup of vertex and index buffers, meshlet buffers, and materials, with flexible configurations for geometry and material handling. It also supports the ability to draw the model in various ways, including instancing for efficient rendering. Additionally, the class offers tools for managing the memory usage of the model and interacting with the underlying device and pipeline for custom rendering operations.

5.198.2 Member Function Documentation

5.198.2.1 create_vertex_buffer()

```
virtual bool Tellusim::MeshModel::create_vertex_buffer (
            const Device & device,
            const void * src,
            size_t size,
            bool owner )  [protected], [virtual]
```

create vertex buffer

Parameters

src	Vertex data.
size	Vertex data size.
owner	If true, the vertex data must be freed with the Allocator class.

5.198.2.2 create_index_buffer()

```
virtual bool Tellusim::MeshModel::create_index_buffer (
             const Device & device,
             const void * src,
             size_t size,
             bool owner )  [protected], [virtual]
```

create index buffer

Parameters

src	Index data.
size	Index data size.
owner	If true, the index data must be freed with the Allocator class.

5.198.2.3 create_meshlet_buffer()

```
virtual bool Tellusim::MeshModel::create_meshlet_buffer (
             const Device & device,
             const void * src,
             size_t size,
             bool owner )  [protected], [virtual]
```

create meshlet buffer

Parameters

src	Meshlet data.
size	Meshlet data size.
owner	If true, the meshlet data must be freed with the Allocator class.

5.199 Tellusim::MeshNode Class Reference

```
#include <format/TellusimMesh.h>
```

Public Member Functions

- **MeshNode** (const char *name=nullptr)
- **MeshNode** (Mesh &mesh, const char *name=nullptr)
- **MeshNode** (MeshNode *parent, const char *name=nullptr)
- **MeshNode** (Mesh &mesh, MeshNode *parent, const char *name=nullptr)
- void clear ()

  clear node
- MeshNode clone (Mesh &mesh) const

  clone node

- void setName (const char *name)

  node name
- String **getName** () const
- uint32_t getIndex () const

  node index
- void setMesh (Mesh &mesh, bool check=true)

  node mesh
- const Mesh **getMesh** () const
- Mesh **getMesh** ()
- uint32_t setParent (MeshNode &parent, bool check=true)

  node parent
- const MeshNode **getParent** () const
- MeshNode **getParent** ()
- bool **isRoot** () const
- void reserveChildren (uint32_t num_children)

  node children
- uint32_t **addChild** (MeshNode &child, bool check=true)
- bool **removeChild** (MeshNode &child)
- void **releaseChildren** ()
- uint32_t **findChild** (const MeshNode &child) const
- uint32_t **findChild** (const char *name) const
- uint32_t **getNumChildren** () const
- const Array< MeshNode > **getChildren** () const
- Array< MeshNode > **getChildren** ()
- const MeshNode **getChild** (uint32_t index) const
- const MeshNode **getChild** (const char *name) const
- MeshNode **getChild** (uint32_t index)
- MeshNode **getChild** (const char *name)
- void clearGeometries ()

  node geometries
- void **reserveGeometries** (uint32_t num_geometries)
- uint32_t **addGeometry** (MeshGeometry &geometry, bool check=true)
- bool **removeGeometry** (MeshGeometry &geometry)
- bool **replaceGeometry** (MeshGeometry &old_geometry, MeshGeometry &geometry)
- uint32_t **findGeometry** (const MeshGeometry &geometry) const
- uint32_t **getNumGeometries** () const
- const Array< MeshGeometry > **getGeometries** () const
- Array< MeshGeometry > **getGeometries** ()
- const MeshGeometry **getGeometry** (uint32_t index) const
- MeshGeometry **getGeometry** (uint32_t index)
- void clearAttachments ()

  node attachments
- void **reserveAttachments** (uint32_t num_attachments)
- uint32_t **addAttachment** (MeshAttachment &attachment, bool check=true)
- bool **removeAttachment** (MeshAttachment &attachment)
- bool **replaceAttachment** (MeshAttachment &old_attachment, MeshAttachment &attachment)
- uint32_t **findAttachment** (const MeshAttachment &attachment) const
- uint32_t **findAttachment** (const char *name) const
- uint32_t **getNumAttachments** () const
- const Array< MeshAttachment > **getAttachments** () const
- Array< MeshAttachment > **getAttachments** ()
- const MeshAttachment **getAttachment** (uint32_t index) const
- MeshAttachment **getAttachment** (uint32_t index)

- void setLocalTransform (const Matrix4x3d &transform)

  local transform
- const Matrix4x3d & **getLocalTransform** () const
- void setGlobalTransform (const Matrix4x3d &transform)

  global transform
- const Matrix4x3d & **getGlobalTransform** () const
- void setPivotTransform (const Matrix4x3d &transform)

  pivot transform
- const Matrix4x3d & **getPivotTransform** () const
- void setMorphTransform (const Vector4f &transform)

  morph transform
- const Vector4f & **getMorphTransform** () const
- void createLocalTransforms (const Matrix4x3d &itransform=Matrix4x3d::identity)

  create transforms
- void **createGlobalTransforms** (const Matrix4x3d &transform=Matrix4x3d::identity)
- void setTransform (const Vector3d &scale)

  apply transform
- size_t getMemory () const

  memory usage

Friends

- class **Mesh**

5.199.1 Detailed Description

The MeshNode class represents a single node in a 3D mesh hierarchy and serves as a container for geometries, attachments, and transformation data. It supports hierarchical relationships by allowing parent-child connections between nodes, enabling complex scene graph structures. Each node can be associated with a mesh, named for identification, and optionally contain geometries.

5.200 Tellusim::MeshStream Class Reference

```
#include <format/TellusimMesh.h>
```

Public Member Functions

- virtual Mesh::Basis **getBasis** () const
- virtual bool **info** (Stream &stream, Mesh &mesh, Mesh::Flags flags, Async *async)
- virtual bool **load** (Stream &stream, Mesh &mesh, Mesh::Flags flags, Async *async)
- virtual bool **save** (Stream &stream, const Mesh &mesh, Mesh::Flags flags)

Static Public Member Functions

- static bool check (const String &name, uint32_t magic=0)

  mesh stream formats
- static String getLoadFormats ()

  list of supported formats
- static String **getSaveFormats** ()

Protected Types

- enum **Flags** {
  FlagNone = 0,
  FlagLoad = (1 << 0),
  FlagSave = (1 << 1),
  FlagLoadSave = (FlagLoad | FlagSave) }

Protected Member Functions

- **MeshStream** (Flags flags, const char *name, uint32_t magic=0)
- **MeshStream** (Flags flags, const InitializerList< const char *> &names, uint32_t magic=0)
- **MeshStream** (Flags flags, const InitializerList< const char *> &names, const InitializerList< uint32_t > &magics)

5.200.1 Detailed Description

The MeshStream class is a base class designed for creating custom mesh stream formats, providing virtual methods for loading and saving meshes through streams. It supports handling different mesh types and formats, serving as a foundation for implementing specific mesh stream formats and enabling the customization and extension of mesh handling functionality. The class also includes static methods for checking supported formats and retrieving lists of compatible load and save formats, offering flexibility in managing and working with various mesh stream formats.

5.201 Tellusim::MeshTransform Class Reference

```
#include <format/TellusimMesh.h>
```

Classes

- struct KeyData
  transform data

Public Types

- using TranslateKeys = Array< KeyData< Vector3d > >
  translation keys
- using RotateKeys = Array< KeyData< Quaternionf > >
  rotation keys
- using ScaleKeys = Array< KeyData< Vector3f > >
  scaling keys
- using MorphKeys = Array< KeyData< Vector4f > >
  morphing keys

Public Member Functions

- void clear ()

  clear transform
- float64_t getMinTime () const

  time range
- float64_t **getMaxTime** () const
- void setTransform (float64_t time, const Matrix4x3d &transform, float32_t threshold=1e-6f)

  set transform
- void **setTranslate** (float64_t time, const Vector3d &translate, float32_t threshold=1e-6f)
- void **setRotate** (float64_t time, const Quaternionf &rotate, float32_t threshold=1e-6f)
- void **setScale** (float64_t time, const Vector3f &scale, float32_t threshold=1e-6f)
- void **setMorph** (float64_t time, const Vector4f &morph, float32_t threshold=1e-6f)
- Matrix4x3d getTransform (float64_t time) const

  get transform
- Vector3d **getTranslate** (float64_t time) const
- Quaternionf **getRotate** (float64_t time) const
- Vector3f **getScale** (float64_t time) const
- Vector4f **getMorph** (float64_t time) const
- bool hasTransformKeys () const

  transform keys
- void **setTranslateKeys** (const TranslateKeys &keys, float64_t scale=1.0)
- TranslateKeys **getTranslateKeys** () const
- bool **hasTranslateKeys** () const
- void **setRotateKeys** (const RotateKeys &keys)
- RotateKeys **getRotateKeys** () const
- bool **hasRotateKeys** () const
- void **setScaleKeys** (const ScaleKeys &keys)
- ScaleKeys **getScaleKeys** () const
- bool **hasScaleKeys** () const
- void **setMorphKeys** (const MorphKeys &keys)
- MorphKeys **getMorphKeys** () const
- bool **hasMorphKeys** () const
- void setTransform (const Vector3d &scale)

  apply transform
- void **setTransform** (const Matrix4x3d &transform)
- size_t getMemory () const

  memory usage

5.201.1 Detailed Description

The MeshTransform class provides functionality for defining and manipulating transformations applied to a 3D mesh over time. It supports transformations such as translation, rotation, scale, and morphing, allowing each transformation to be set and retrieved for specific time intervals. The class includes methods for managing keyframes for each type of transformation, enabling smooth animation and interpolation of transformations between time points.

5.202 Tellusim::Mipmap Struct Reference

```
#include <TellusimTypes.h>
```

Public Member Functions

- **Mipmap** (uint32_t base)
- **Mipmap** (uint32_t base, uint32_t size)

Public Attributes

- uint32_t **base** = 0
- uint32_t **size** = 1

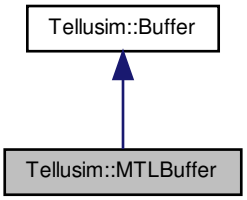
5.202.1 Detailed Description

The Mipmap struct represents a level of a mipmap chain. It has two members: base, indicating the base level of the mipmap, and size, which defines the number of mipmap levels. Mipmaps are precomputed textures at various levels of detail used in 3D rendering to improve performance and visual quality.

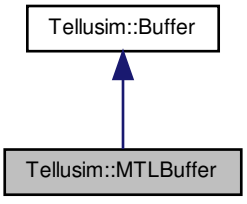
5.203 Tellusim::MTLBuffer Class Reference

```
#include <platform/TellusimBuffer.h>
```

Inheritance diagram for Tellusim::MTLBuffer:



Collaboration diagram for Tellusim::MTLBuffer:



Public Member Functions

- bool create (Flags flags, void *buffer)

  create external buffer
- void * **getMTLBuffer** () const
- void * **getSharedPtr** () const

Additional Inherited Members

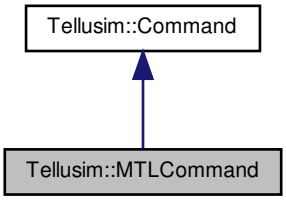
5.203.1 Detailed Description

The MTLBuffer class is a Metal-specific implementation of the Buffer class, providing access to internal resources in Metal-based applications. It supports the creation of external buffers from raw pointer data and offers methods to retrieve the Metal buffer and shared pointer for interop with other systems. The class also inherits the create method from the Buffer class to allow for flexible buffer creation and management.

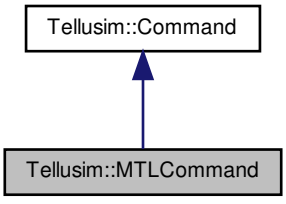
5.204 Tellusim::MTLCommand Class Reference

```
#include <platform/TellusimCommand.h>
```

Inheritance diagram for Tellusim::MTLCommand:



Collaboration diagram for Tellusim::MTLCommand:



Public Member Functions

- void * getEncoder () const

  command context
- void flush (void *encoder, bool enqueue=false)

  end encoding
- void **flush** (bool create=false, bool enqueue=false)
- void update ()

  update resources

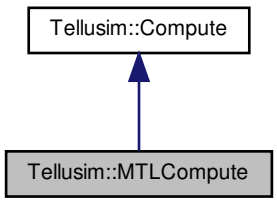
5.204.1 Detailed Description

The MTLCommand class is a Metal-specific implementation of the Command class, providing access to the command encoder.

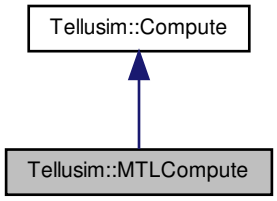
5.205 Tellusim::MTLCompute Class Reference

```
#include <platform/TellusimCompute.h>
```

Inheritance diagram for Tellusim::MTLCompute:



Collaboration diagram for Tellusim::MTLCompute:



Public Member Functions

- void * getEncoder () const

  command context
- void flush (void *encoder, bool enqueue=false)

  end encoding
- void **flush** (bool create=false, bool enqueue=false)
- void update ()

  update resources

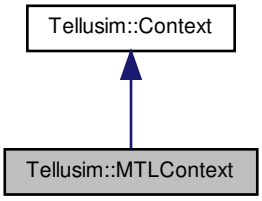
5.205.1 Detailed Description

The MTLCompute class is a Metal-specific implementation of the Compute class, providing access to the command encoder.

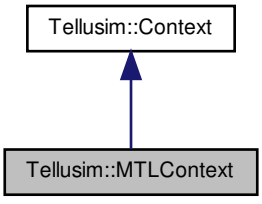
5.206 Tellusim::MTLContext Class Reference

```
#include <platform/TellusimContext.h>
```

Inheritance diagram for Tellusim::MTLContext:



Collaboration diagram for Tellusim::MTLContext:



Public Member Functions

- bool create (void *device, void *queue)

  create context
- void * getDevice () const

  current device
- void * **getQueue** () const
- void * **getCommand** () const
- void * getEncoder () const

  command encoder
- void * **getRenderEncoder** (void *descriptor) const
- void * **getComputeEncoder** () const
- void * **getTracingEncoder** () const
- void * **getBlitEncoder** () const
- void **endEncoder** () const

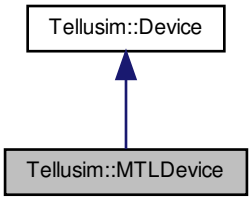
5.206.1 Detailed Description

The MTLContext class is a Metal-specific implementation of the Context class. It allows initializing the rendering context using externally provided Metal device and command queue pointers. The class provides access to the underlying device, command queue, command buffer, and Metal command encoders.

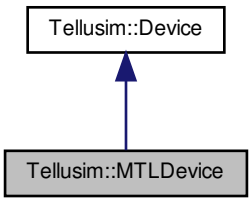
5.207 Tellusim::MTLDevice Class Reference

```
#include <platform/TellusimDevice.h>
```

Inheritance diagram for Tellusim::MTLDevice:



Collaboration diagram for Tellusim::MTLDevice:



Public Member Functions

- **MTLDevice** (Context &context)
- **MTLDevice** (Surface &surface)
- **MTLDevice** (Window &window)
- void * getMTLDevice () const

  command context
- void * **getQueue** () const
- void * **getCommand** () const
- void * getEncoder () const

  command encoder
- void * **getRenderEncoder** (void *descriptor) const
- void * **getComputeEncoder** () const
- void * **getTracingEncoder** () const
- void * **getBlitEncoder** () const
- void **endEncoder** () const

5.207.1 Detailed Description

The MTLDevice class extends the Device class to provide Metal-specific functionality for managing a rendering device. It offers access to the Metal device, command queue, and command context, enabling efficient rendering workflows on macOS and iOS platforms.

5.208 Tellusim::MTLTracing::MTLInstance Struct Reference

tracing instance

```
#include <platform/TellusimTracing.h>
```

Public Attributes

- float32_t **transform** [12]
- uint32_t **options**
- uint32_t **mask**
- uint32_t **offset**
- uint32_t **index**

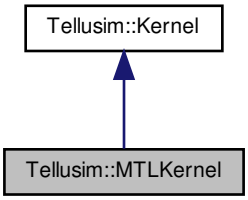
5.208.1 Detailed Description

tracing instance

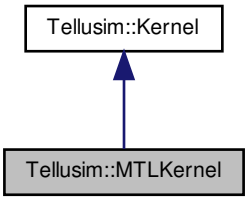
5.209 Tellusim::MTLKernel Class Reference

`#include <platform/TellusimKernel.h>`

Inheritance diagram for Tellusim::MTLKernel:



Collaboration diagram for Tellusim::MTLKernel:



Public Member Functions

- void setIndirect (bool enabled)
  indirect command buffer
- bool **isIndirect** () const
- void * **getComputeFunction** () const

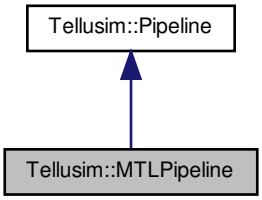
5.209.1 Detailed Description

The MTLKernel class is a compute kernel designed for Metal, inheriting from the Kernel class. It allows enabling or disabling indirect command buffers and provides access to the compute function.

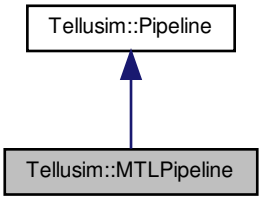
5.210 Tellusim::MTLPipeline Class Reference

`#include <platform/TellusimPipeline.h>`

Inheritance diagram for Tellusim::MTLPipeline:



Collaboration diagram for Tellusim::MTLPipeline:



Public Member Functions

- void setIndirect (bool enabled)

  indirect command buffer
- bool **isIndirect** () const
- void * **getVertexFunction** () const
- void * **getFragmentFunction** () const

Additional Inherited Members

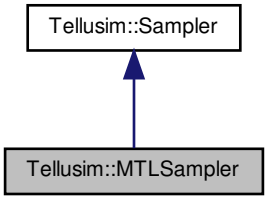
5.210.1 Detailed Description

The MTLPipeline class is a graphics pipeline implementation for Metal, inheriting from the Pipeline class. It supports indirect command buffer execution and provides access to the underlying Metal vertex and fragment functions.

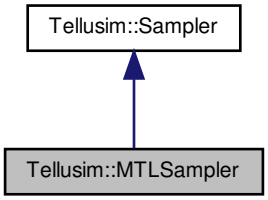
5.211 Tellusim::MTLSampler Class Reference

`#include <platform/TellusimSampler.h>`

Inheritance diagram for Tellusim::MTLSampler:



Collaboration diagram for Tellusim::MTLSampler:



Public Member Functions

- void setIndirect (bool enabled)

  indirect command buffer
- bool **isIndirect** () const

Additional Inherited Members

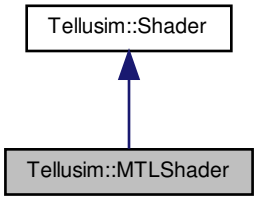
5.211.1 Detailed Description

The MTLSampler class extends the Sampler class and provides additional functionality specific to Metal, including support for indirect command buffers. It allows enabling or disabling the indirect command feature, which is useful for optimizing rendering commands and reducing CPU overhead.

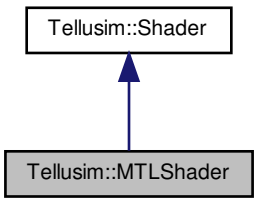
5.212 Tellusim::MTLShader Class Reference

`#include <platform/TellusimShader.h>`

Inheritance diagram for Tellusim::MTLShader:



Collaboration diagram for Tellusim::MTLShader:



Public Member Functions

- void setIndirect (bool enabled)

  indirect command buffer
- bool **isIndirect** () const
- void * **getLibrary** () const
- void * **getFunction** () const

Additional Inherited Members

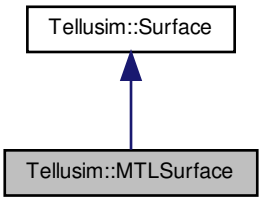
5.212.1 Detailed Description

The MTLShader class extends the Shader class to specialize in managing shaders for Metal. It provides methods to retrieve the underlying Metal shader library and function, enabling integration with Metal.

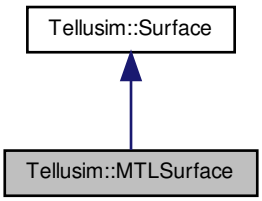
5.213 Tellusim::MTLSurface Class Reference

`#include <platform/TellusimSurface.h>`

Inheritance diagram for Tellusim::MTLSurface:



Collaboration diagram for Tellusim::MTLSurface:



Public Member Functions

- **MTLSurface** (MTLContext &context)
- void * getDevice () const

  current device
- void * **getQueue** () const
- void * **getCommand** () const
- void setDescriptor (void *descriptor)

  render pass descriptor
- void * **getDescriptor** () const
- uint32_t getColorPixelFormat () const

  surface formats
- uint32_t **getDepthPixelFormat** () const

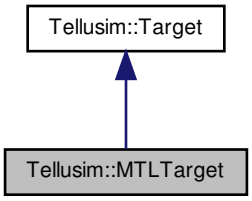
5.213.1 Detailed Description

The MTLSurface class extends the Surface class to provide Metal-specific functionality for managing a rendering surface. It includes methods for interacting with Metal devices, queues, and command buffers, enabling rendering operations in the context of Metal. The class supports managing render pass descriptors and surface formats, which are essential for rendering operations.

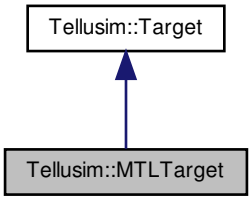
5.214 Tellusim::MTLTarget Class Reference

`#include <platform/TellusimTarget.h>`

Inheritance diagram for Tellusim::MTLTarget:



Collaboration diagram for Tellusim::MTLTarget:



Public Member Functions

- void * **getDescriptor** () const

Additional Inherited Members

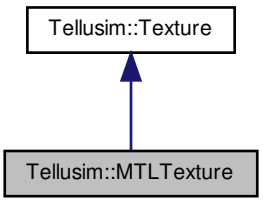
5.214.1 Detailed Description

The MTLTarget class is a Metal-specific implementation of the Target class, providing functionality for interacting with Metal render and depth-stencil targets. It offers access to a descriptor object, which encapsulates the configuration of a Metal rendering target.

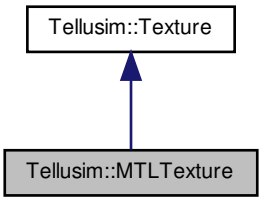
5.215 Tellusim::MTLTexture Class Reference

`#include <platform/TellusimTexture.h>`

Inheritance diagram for Tellusim::MTLTexture:



Collaboration diagram for Tellusim::MTLTexture:



Public Member Functions

- bool create (void *texture, Flags flags=DefaultFlags, Format format=FormatUnknown)
  create external texture
- uint32_t **getPixelFormat** () const
- uint32_t **getTextureType** () const
- void * **getMTLTexture** () const
- void * **getMTLBuffer** () const
- void * **getSharedPtr** () const

Additional Inherited Members

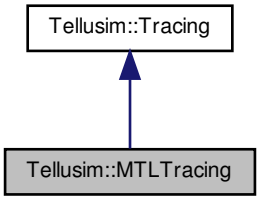
5.215.1 Detailed Description

The MTLTexture class is a Metal-specific implementation of the Texture class, offering access to Metal resources and enabling texture management in Metal applications. It supports external texture creation through raw pointer data and provides methods to retrieve the Metal texture, buffer, and shared pointer for interoperability. The class inherits the create method from the Texture class, facilitating flexible texture creation and management.

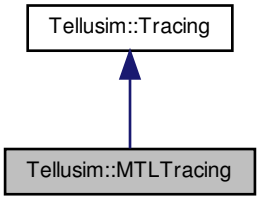
5.216 Tellusim::MTLTracing Class Reference

`#include <platform/TellusimTracing.h>`

Inheritance diagram for Tellusim::MTLTracing:



Collaboration diagram for Tellusim::MTLTracing:



Classes

- struct MTLInstance
  tracing instance

Public Member Functions

- void * **getGeometryDesc** (uint32_t index) const
- void * **getPrimitiveDesc** () const
- void * **getInstanceDesc** () const
- void * **getAccelerationStructure** () const

Additional Inherited Members

5.216.1 Detailed Description

The MTLTracing class is a Metal-specific implementation of the Tracing class. It provides methods and structures for managing ray-tracing acceleration structures within the Metal API.

5.217 Tellusim::Spatial::Node< Type, Size > Struct Template Reference

Spatial Node.

`#include <geometry/TellusimSpatial.h>`

Public Types

- enum { **Axes** = Size }
- using **Bound** = Type

Public Attributes

- Bound **bound**
- uint32_t **left**
- uint32_t **right**
- uint32_t **parent**
- uint32_t **spatial**

5.217.1 Detailed Description

template<class Type, uint32_t Size>
struct Tellusim::Spatial::Node< Type, Size >

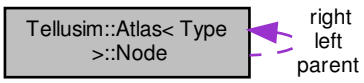
Spatial Node.

5.218 Tellusim::Atlas< Type >::Node Struct Reference

Atlas Node.

`#include <geometry/TellusimAtlas.h>`

Collaboration diagram for Tellusim::Atlas< Type >::Node:



Public Attributes

- Bound **bound**
- Node * **left** = nullptr
- Node * **right** = nullptr
- Node * **parent** = nullptr
- uint32_t **axis** = Maxu32

5.218.1 Detailed Description

template<class Type>
struct Tellusim::Atlas< Type >::Node

Atlas Node.

5.219 Tellusim::SpatialTree::Node Struct Reference

Public Attributes

- float32_t **bound_min** [3]
- float32_t **is_enabled**
- float32_t **bound_max** [3]
- uint32_t **data**
- uint32_t **left**
- uint32_t **right**
- uint32_t **parent**
- uint32_t **spatial**

5.220 Tellusim::Origin Struct Reference

```
#include <TellusimTypes.h>
```

Public Member Functions

- **Origin** (uint32_t x, uint32_t y)
- **Origin** (uint32_t x, uint32_t y, uint32_t z)

Public Attributes

- uint32_t **x** = 0
- uint32_t **y** = 0
- uint32_t **z** = 0

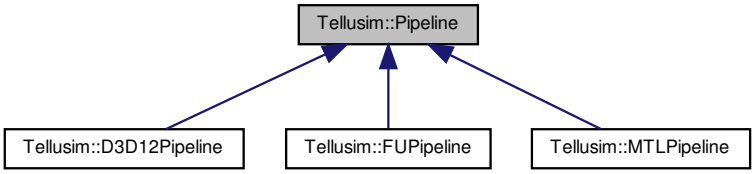
5.220.1 Detailed Description

The Origin struct represents a point in 2D or 3D space with integer coordinates. It provides constructors to initialize the point in either 2D (with x and y coordinates) or 3D (with x, y, and z coordinates) space. By default, the x, y, and z values are set to zero.

5.221 Tellusim::Pipeline Class Reference

`#include <platform/TellusimPipeline.h>`

Inheritance diagram for Tellusim::Pipeline:



Public Types

- enum Primitive {
  PrimitivePoint = 0,
  PrimitivePointPatch,
  PrimitiveLine,
  PrimitiveLineAdj,
  PrimitiveLineStrip,
  PrimitiveLinePatch,
  PrimitiveTriangle,
  PrimitiveTriangleAdj,
  PrimitiveTriangleStrip,
  PrimitiveTrianglePatch,
  PrimitiveQuadrilateralPatch,
  NumPrimitiveTypes }

  Primitive types.
- enum Attribute {
  AttributePosition = 0,
  AttributeBasis,
  AttributeNormal,
  AttributeTangent,
  AttributeBinormal,
  AttributeTexCoord,
  AttributeWeights,
  AttributeJoints,
  AttributeColor,
  AttributeIndex,
  NumAttributeTypes }

  Attribute types.
- enum FillMode {
  FillModeLine = 0,
  FillModeSolid,
  NumFillModes }

  Filling modes.

- enum CullMode {
  CullModeNone = 0,
  CullModeBack,
  CullModeFront,
  NumCullModes }

  Culling modes.
- enum FrontMode {
  FrontModeCCW = 0,
  FrontModeCW,
  NumFrontModes }

  Front modes.
- enum BlendOp {
  BlendOpAdd = 0,
  BlendOpSub,
  BlendOpMin,
  BlendOpMax,
  NumBlendOperations }

  Blending operations.
- enum BlendFunc {
  BlendFuncNone = 0,
  BlendFuncZero,
  BlendFuncOne,
  BlendFuncSrcColor,
  BlendFuncSrcAlpha,
  BlendFuncSrc1Color,
  BlendFuncSrc1Alpha,
  BlendFuncDestColor,
  BlendFuncDestAlpha,
  BlendFuncFactorColor,
  BlendFuncFactorAlpha,
  BlendFuncInvSrcColor,
  BlendFuncInvSrcAlpha,
  BlendFuncInvSrc1Color,
  BlendFuncInvSrc1Alpha,
  BlendFuncInvDestColor,
  BlendFuncInvDestAlpha,
  BlendFuncInvFactorColor,
  BlendFuncInvFactorAlpha,
  NumBlendFunctions }

  Blending functions.
- enum ColorMask {
  ColorMaskNone = 0,
  ColorMaskR = (1 << 0),
  ColorMaskG = (1 << 1),
  ColorMaskB = (1 << 2),
  ColorMaskA = (1 << 3),
  ColorMaskUnknown = (1 << 4),
  ColorMaskRGB = (ColorMaskR | ColorMaskG | ColorMaskB),
  ColorMaskAll = (ColorMaskRGB | ColorMaskA) }

  Color masks.
- enum DepthMask {
  DepthMaskNone = 0,
  DepthMaskRead,
  DepthMaskWrite,
  NumDepthMasks }

  Depth masks.

- enum DepthFunc {
  DepthFuncNone = 0,
  DepthFuncNever,
  DepthFuncAlways,
  DepthFuncEqual,
  DepthFuncLess,
  DepthFuncGreater,
  DepthFuncNotEqual,
  DepthFuncLessEqual,
  DepthFuncGreaterEqual,
  NumDepthFunctions }

  Depth functions.
- enum StencilOp {
  StencilOpKeep = 0,
  StencilOpInvert,
  StencilOpReplace,
  StencilOpIncrWrap,
  StencilOpDecrWrap,
  StencilOpIncrSat,
  StencilOpDecrSat,
  NumStencilOperations }

  Stencil operations.
- enum StencilFunc {
  StencilFuncNone = 0,
  StencilFuncNever,
  StencilFuncAlways,
  StencilFuncEqual,
  StencilFuncLess,
  StencilFuncGreater,
  StencilFuncNotEqual,
  StencilFuncLessEqual,
  StencilFuncGreaterEqual,
  NumStencilFunctions }

  Stencil functions.

Public Member Functions

- Platform getPlatform () const

  pipeline platform
- const char * **getPlatformName** () const
- uint32_t getIndex () const

  pipeline device index
- void clear ()

  clear pipeline
- bool isCreated () const

  check pipeline
- void setName (const char *name)

  pipeline name
- String **getName** () const
- bool create ()

  create pipeline
- void setParameters (const Pipeline &pipeline)

  pipeline parameters
- bool **saveState** (Stream &stream) const

- void addShader (Shader &shader, bool owner=false)

  shader pointers
- Shader **getVertexShader** () const
- Shader **getControlShader** () const
- Shader **getEvaluateShader** () const
- Shader **getGeometryShader** () const
- Shader **getFragmentShader** () const
- Shader **getTaskShader** () const
- Shader **getMeshShader** () const
- bool loadShader (Shader::Type type, const char *name, const char *format,...) 1(4

  load shaders
- bool bool **loadShaderGLSL** (Shader::Type type, const char *name, const char *format,...) 1(4
- bool bool bool **loadShader** (Shader::Type type, const char *name, const String &macros=String::null, const char **includes=nullptr, uint32_t size=0)
- bool **loadShaderGLSL** (Shader::Type type, const char *name, const String &macros=String::null, const char **includes=nullptr, uint32_t size=0)
- bool **loadShaderSPIRV** (Shader::Type type, const char *name)
- bool createShader (Shader::Type type, const char *src, const char *format,...) 1(4

  create shaders
- bool bool **createShaderGLSL** (Shader::Type type, const char *src, const char *format,...) 1(4
- bool bool bool **createShader** (Shader::Type type, const char *src, const String &macros=String::null, const char **includes=nullptr, uint32_t size=0)
- bool **createShaderGLSL** (Shader::Type type, const char *src, const String &macros=String::null, const char **includes=nullptr, uint32_t size=0)
- bool **createShaderSPIRV** (Shader::Type type, const Array< uint32_t > &data)
- uint32_t addSampler (Shader::Mask mask)

  sampler parameters
- uint32_t **getNumSamplers** () const
- Pipeline & **setSamplerOffset** (uint32_t offset)
- uint32_t **getSamplerOffset** () const
- Pipeline & **setSamplerMask** (uint32_t index, Shader::Mask mask)
- Shader::Mask **getSamplerMask** (uint32_t index) const
- Pipeline & **setSamplerMasks** (uint32_t index, uint32_t num, Shader::Mask mask, bool array=false)
- Shader::Mask **getSamplerMasks** (uint32_t index, uint32_t num) const
- Pipeline & **setSamplerArray** (uint32_t index, uint32_t num, bool array)
- uint32_t **getSamplerArray** (uint32_t index) const
- uint32_t addTexture (Shader::Mask mask)

  texture parameters
- uint32_t **getNumTextures** () const
- Pipeline & **setTextureOffset** (uint32_t offset)
- uint32_t **getTextureOffset** () const
- Pipeline & **setTextureMask** (uint32_t index, Shader::Mask mask)
- Shader::Mask **getTextureMask** (uint32_t index) const
- Pipeline & **setTextureMasks** (uint32_t index, uint32_t num, Shader::Mask mask, bool array=false)
- Shader::Mask **getTextureMasks** (uint32_t index, uint32_t num) const
- Pipeline & **setTextureArray** (uint32_t index, uint32_t num, bool array)
- uint32_t **getTextureArray** (uint32_t index) const
- uint32_t addSurface (Shader::Mask mask)

  surface parameters
- uint32_t **getNumSurfaces** () const
- Pipeline & **setSurfaceOffset** (uint32_t offset)
- uint32_t **getSurfaceOffset** () const
- Pipeline & **setSurfaceMask** (uint32_t index, Shader::Mask mask)
- Shader::Mask **getSurfaceMask** (uint32_t index) const

- Pipeline & **setSurfaceMasks** (uint32_t index, uint32_t num, Shader::Mask mask, bool array=false)
- Shader::Mask **getSurfaceMasks** (uint32_t index, uint32_t num) const
- Pipeline & **setSurfaceArray** (uint32_t index, uint32_t num, bool array)
- uint32_t **getSurfaceArray** (uint32_t index) const
- uint32_t addUniform (Shader::Mask mask, BindFlags flags=BindFlagNone)

  uniform parameters
- uint32_t **getNumUniforms** () const
- Pipeline & **setUniformOffset** (uint32_t offset)
- uint32_t **getUniformOffset** () const
- Pipeline & **setUniformMask** (uint32_t index, Shader::Mask mask, BindFlags flags=BindFlagNone)
- Shader::Mask **getUniformMask** (uint32_t index) const
- Pipeline & **setUniformMasks** (uint32_t index, uint32_t num, Shader::Mask mask, BindFlags flags=Bind↩ FlagNone)
- Shader::Mask **getUniformMasks** (uint32_t index, uint32_t num) const
- Pipeline & **setUniformFlags** (uint32_t index, BindFlags flags)
- BindFlags **getUniformFlags** (uint32_t index) const
- uint32_t addStorage (Shader::Mask mask, BindFlags flags=BindFlagNone)

  storage parameters
- uint32_t **getNumStorages** () const
- Pipeline & **setStorageOffset** (uint32_t offset)
- uint32_t **getStorageOffset** () const
- Pipeline & **setStorageMask** (uint32_t index, Shader::Mask mask, BindFlags flags=BindFlagNone)
- Shader::Mask **getStorageMask** (uint32_t index) const
- Pipeline & **setStorageMasks** (uint32_t index, uint32_t num, Shader::Mask mask, BindFlags flags=Bind↩ FlagNone)
- Shader::Mask **getStorageMasks** (uint32_t index, uint32_t num) const
- Pipeline & **setStorageFlags** (uint32_t index, BindFlags flags)
- BindFlags **getStorageFlags** (uint32_t index) const
- uint32_t addTracing (Shader::Mask mask)

  tracing parameters
- uint32_t **getNumTracings** () const
- Pipeline & **setTracingOffset** (uint32_t offset)
- uint32_t **getTracingOffset** () const
- Pipeline & **setTracingMask** (uint32_t index, Shader::Mask mask)
- Shader::Mask **getTracingMask** (uint32_t index) const
- Pipeline & **setTracingMasks** (uint32_t index, uint32_t num, Shader::Mask mask)
- Shader::Mask **getTracingMasks** (uint32_t index, uint32_t num) const
- uint32_t addTexel (Shader::Mask mask)

  texel parameters
- uint32_t **getNumTexels** () const
- Pipeline & **setTexelOffset** (uint32_t offset)
- uint32_t **getTexelOffset** () const
- Pipeline & **setTexelMask** (uint32_t index, Shader::Mask mask)
- Shader::Mask **getTexelMask** (uint32_t index) const
- Pipeline & **setTexelMasks** (uint32_t index, uint32_t num, Shader::Mask mask)
- Shader::Mask **getTexelMasks** (uint32_t index, uint32_t num) const
- uint32_t addTable (TableType type, uint32_t size, Shader::Mask mask, BindFlags flags=BindFlagNone)

  table parameters
- uint32_t **getNumTables** () const
- Pipeline & **setTableOffset** (uint32_t offset)
- uint32_t **getTableOffset** () const
- Pipeline & **setTableType** (uint32_t index, TableType type, uint32_t size, Shader::Mask mask, BindFlags flags=BindFlagNone)
- TableType **getTableType** (uint32_t index) const

- uint32_t **getTableSize** (uint32_t index) const
- Pipeline & **setTableMask** (uint32_t index, Shader::Mask mask, BindFlags flags=BindFlagNone)
- Shader::Mask **getTableMask** (uint32_t index) const
- Pipeline & **setTableFlags** (uint32_t index, BindFlags flags)
- BindFlags **getTableFlags** (uint32_t index) const
- uint32_t getNumVertices () const

  vertex parameters
- uint32_t **getVertexStride** (uint32_t index) const
- uint32_t **getVertexRate** (uint32_t index) const
- uint32_t addAttribute (Attribute attribute, Format format, uint32_t vertex, size_t offset, size_t stride, uint32_t rate=0)

  vertex attributes
- Pipeline & **setAttribute** (uint32_t index, Attribute attribute, Format format, uint32_t vertex, size_t offset, size_t stride, uint32_t rate=0)
- Pipeline & **setAttributeType** (uint32_t index, Attribute attribute)
- Pipeline & **setAttributeFormat** (uint32_t index, Format format)
- Pipeline & **setAttributeVertex** (uint32_t index, uint32_t vertex)
- Pipeline & **setAttributeOffset** (uint32_t index, size_t offset)
- Pipeline & **setAttributeStride** (uint32_t index, size_t stride)
- Pipeline & **setAttributeRate** (uint32_t index, uint32_t rate)
- uint32_t **getNumAttributes** () const
- Attribute **getAttributeType** (uint32_t index) const
- Format **getAttributeFormat** (uint32_t index) const
- uint32_t **getAttributeVertex** (uint32_t index) const
- uint32_t **getAttributeOffset** (uint32_t index) const
- uint32_t **getAttributeStride** (uint32_t index) const
- uint32_t **getAttributeRate** (uint32_t index) const
- void setPrimitive (Primitive primitive)

  rasterization parameters
- Primitive **getPrimitive** () const
- void **setFillMode** (FillMode mode)
- FillMode **getFillMode** () const
- void **setCullMode** (CullMode mode)
- CullMode **getCullMode** () const
- void **setFrontMode** (FrontMode mode)
- FrontMode **getFrontMode** () const
- void **setDepthBias** (float32_t bias, float32_t slope, float32_t clamp=0.0f)
- float32_t **getDepthBias** () const
- float32_t **getDepthSlope** () const
- float32_t **getDepthClamp** () const
- void **setMultisample** (uint32_t multisample)
- uint32_t **getMultisample** () const
- void **setSampleMask** (uint32_t sample_mask)
- uint32_t **getSampleMask** () const
- void **setDepthClip** (bool enabled)
- bool **getDepthClip** () const
- void **setDepthReplace** (bool enabled)
- bool **getDepthReplace** () const
- void **setScissorTest** (bool enabled)
- bool **getScissorTest** () const
- void **setRasterDiscard** (bool enabled)
- bool **getRasterDiscard** () const
- void **setSampleShading** (bool enabled)
- bool **getSampleShading** () const

- void **setAlphaToCoverage** (bool enabled)
- bool **getAlphaToCoverage** () const
- void **setMultisampleRaster** (bool enabled)
- bool **getMultisampleRaster** () const
- void **setConservativeRaster** (bool enabled)
- bool **getConservativeRaster** () const
- void **setNumViewports** (uint32_t num_viewports)
- uint32_t **getNumTargets** () const
- uint32_t **getNumViewports** () const
- void **setNumClipDistances** (uint32_t num_distances)
- uint32_t **getNumClipDistances** () const
- void setBlend (BlendOp op, BlendFunc src, BlendFunc dest)

  blending parameters
- void **setBlendColor** (BlendOp op, BlendFunc src, BlendFunc dest)
- void **setBlendAlpha** (BlendOp op, BlendFunc src, BlendFunc dest)
- void **setBlend** (uint32_t index, BlendOp op, BlendFunc src, BlendFunc dest)
- void **setBlendColor** (uint32_t index, BlendOp op, BlendFunc src, BlendFunc dest)
- void **setBlendAlpha** (uint32_t index, BlendOp op, BlendFunc src, BlendFunc dest)
- BlendOp **getBlendColorOp** (uint32_t index) const
- BlendOp **getBlendAlphaOp** (uint32_t index) const
- BlendFunc **getBlendSrcColorFunc** (uint32_t index) const
- BlendFunc **getBlendSrcAlphaFunc** (uint32_t index) const
- BlendFunc **getBlendDestColorFunc** (uint32_t index) const
- BlendFunc **getBlendDestAlphaFunc** (uint32_t index) const
- void setColorMask (ColorMask mask)

  color parameters
- void **setColorMask** (uint32_t index, ColorMask mask)
- void **setColorFormat** (uint32_t index, Format format)
- void **setColorFormat** (Format format, uint32_t num=1)
- ColorMask **getColorMask** (uint32_t index) const
- Format **getColorFormat** (uint32_t index) const
- void setDepthMask (DepthMask mask)

  depth parameters
- void **setDepthFunc** (DepthFunc func)
- void **setDepthFormat** (Format format)
- DepthMask **getDepthMask** () const
- DepthFunc **getDepthFunc** () const
- Format **getDepthFormat** () const
- void setStencilMask (uint32_t mask)

  stencil parameters
- void **setStencilBackMask** (uint32_t mask)
- void **setStencilFrontMask** (uint32_t mask)
- void **setStencilFunc** (StencilFunc func, StencilOp dpass_op)
- void **setStencilBackFunc** (StencilFunc func, StencilOp dpass_op)
- void **setStencilFrontFunc** (StencilFunc func, StencilOp dpass_op)
- void **setStencilFunc** (StencilFunc func, StencilOp fail_op, StencilOp dfail_op, StencilOp dpass_op)
- void **setStencilBackFunc** (StencilFunc func, StencilOp fail_op, StencilOp dfail_op, StencilOp dpass_op)
- void **setStencilFrontFunc** (StencilFunc func, StencilOp fail_op, StencilOp dfail_op, StencilOp dpass_op)
- uint32_t **getStencilBackMask** () const
- StencilFunc **getStencilBackFunc** () const
- StencilOp **getStencilBackFailOp** () const
- StencilOp **getStencilBackDepthFailOp** () const
- StencilOp **getStencilBackDepthPassOp** () const
- uint32_t **getStencilFrontMask** () const
- StencilFunc **getStencilFrontFunc** () const
- StencilOp **getStencilFrontFailOp** () const
- StencilOp **getStencilFrontDepthFailOp** () const
- StencilOp **getStencilFrontDepthPassOp** () const

5.221.1 Detailed Description

The Pipeline class manages the configuration of a graphics pipeline, offering control over shader stages, primitive types, blending, depth and stencil testing, and rasterization. It allows users to configure and retrieve various pipeline parameters, including vertex attributes, texture and surface bindings, sampling configurations, and memory offsets. The class supports shader creation, loading, and compilation in multiple formats such as native, GLSL, and SPIRV, providing fine-grained control over pipeline states.

5.222 Tellusim::Polygon< Capacity > Struct Template Reference

`#include <geometry/TellusimPolygon.h>`

Static Public Member Functions

- template<class Type >
  static Vector3< Type > normal (const Vector3< Type > *vertices, uint32_t num_vertices)
  3D polygon normal
- template<class Type , class Index >
  static uint32_t triangulate (const Vector3< Type > *vertices, uint32_t num_vertices, Index *indices)
  triangulate 3D polygon
- template<class Type , class Index >
  static uint32_t triangulate (const Vector3< Type > *vertices, uint32_t num_vertices, const Vector3< Type > &normal, Index *indices)
  triangulate 3D polygon with specified normal
- template<class Type , class Index >
  static uint32_t triangulate (const Vector2< Type > *vertices, uint32_t num_vertices, Index *indices)
  triangulate 2D polygon

5.222.1 Detailed Description

template<uint32_t Capacity = 256>
struct Tellusim::Polygon< Capacity >

Polygon utils

5.222.2 Member Function Documentation

5.222.2.1 normal()

```
template<uint32_t Capacity = 256>
template<class Type >
static Vector3<Type> Tellusim::Polygon< Capacity >::normal (
            const Vector3< Type > * vertices,
            uint32_t num_vertices )  [inline], [static]
```

3D polygon normal

vertex bounds

maximum axis

maximum area

5.222.2.2 triangulate() [1/3]

```
template<uint32_t Capacity = 256>
template<class Type , class Index >
static uint32_t Tellusim::Polygon< Capacity >::triangulate (
            const Vector3< Type > * vertices,
            uint32_t num_vertices,
            Index * indices )  [inline], [static]
```

triangulate 3D polygon

polygon normal

triangulate polygon

5.222.2.3 triangulate() [2/3]

```
template<uint32_t Capacity = 256>
template<class Type , class Index >
static uint32_t Tellusim::Polygon< Capacity >::triangulate (
            const Vector3< Type > * vertices,
            uint32_t num_vertices,
            const Vector3< Type > & normal,
            Index * indices )  [inline], [static]
```

triangulate 3D polygon with specified normal

polygon normal basis

project polygon vertices

triangulate polygon

5.222.2.4 triangulate() [3/3]

```
template<uint32_t Capacity = 256>
template<class Type , class Index >
static uint32_t Tellusim::Polygon< Capacity >::triangulate (
            const Vector2< Type > * vertices,
            uint32_t num_vertices,
            Index * indices )  [inline], [static]
```

triangulate 2D polygon

triangle polygon

signed polygon area

polygon winding

vertex angles

triangulate polygon

find next vertex

next vertex

update angles

next triangle

5.223 Tellusim::PrefixScan Class Reference

`#include <parallel/TellusimPrefixScan.h>`

Classes

- struct DispatchParameters

Public Types

- enum Mode {
  ModeSingle = 0,
  ModeMultiple,
  NumModes }

  Scan modes.
- enum Flags {
  FlagNone = 0,
  FlagSingle = (1 << ModeSingle),
  FlagMultiple = (1 << ModeMultiple),
  FlagIndirect = (1 << (NumModes + 0)),
  FlagRepeat = (1 << (NumModes + 1)),
  FlagsAll = (FlagSingle | FlagMultiple | FlagIndirect) }

  Scan flags.

Public Member Functions

- void clear ()

  clear scan
- bool isCreated (Flags flags) const

  check scan
- uint32_t getGroupSize () const

  scan parameters
- uint32_t **getScanElements** () const
- uint32_t **getMaxElements** () const
- uint32_t **getMaxRegions** () const
- bool create (const Device &device, Mode mode, uint32_t groups=256, uint32_t regions=1, Async *async=nullptr)
- bool **create** (const Device &device, Flags flags, uint32_t groups=256, uint32_t regions=1, Async *async=nullptr)
- bool dispatch (Compute &compute, Buffer &data, uint32_t offset, uint32_t size)
- bool dispatch (Compute &compute, Buffer &data, uint32_t count, const uint32_t *offsets, const uint32_t *sizes, Flags flags=FlagNone)
- bool dispatchIndirect (Compute &compute, Buffer &data, Buffer &dispatch, uint32_t offset, Flags flags=FlagNone, uint32_t max_size=Maxu32)
- bool dispatchIndirect (Compute &compute, Buffer &data, uint32_t count, Buffer &dispatch, uint32_t offset, Flags flags=FlagNone, uint32_t max_size=Maxu32)
- bool dispatchIndirect (Compute &compute, Buffer &data, Buffer &count, Buffer &dispatch, uint32_t count_offset, uint32_t dispatch_offset, Flags flags=FlagNone, uint32_t max_size=Maxu32)

5.223.1 Detailed Description

The PrefixScan class provides an efficient implementation of the prefix scan (sum) algorithm, supporting both single and multiple array scanning. It is highly configurable and optimized for parallel computing, commonly used in tasks such as data aggregation and parallel processing. The class supports various modes and flags, allowing for flexible control over the scanning process, including the ability to perform indirect scans using dispatch buffers. Its methods enable in-place scanning for both single and multiple datasets, with support for large-scale data processing across multiple regions.

5.223.2 Member Function Documentation

5.223.2.1 create()

```
bool Tellusim::PrefixScan::create (
             const Device & device,
             Mode mode,
             uint32_t groups = 256,
             uint32_t regions = 1,
             Async * async = nullptr )
```

create prefix scan

Parameters

groups	Prefix scan group size.
regions	Maximum number of multiple regions.

5.223.2.2 dispatch() [1/2]

```
bool Tellusim::PrefixScan::dispatch (
             Compute & compute,
             Buffer & data,
             uint32_t offset,
             uint32_t size )
```

dispatch single in-place prefix scan

Parameters

data	Buffer of uint32_t elements to scan.
offset	Elements offset index (4 aligned).
size	Number of uint32_t elements to scan.

5.223.2.3 dispatch() [2/2]

```
bool Tellusim::PrefixScan::dispatch (
```

```
            Compute & compute,
            Buffer & data,
            uint32_t count,
            const uint32_t * offsets,
            const uint32_t * sizes,
            Flags flags = FlagNone )
```

dispatch multiple in-place prefix scans

Parameters

data	Buffer of uint32_t elements to scan.
count	Number of regions to scan.
offsets	Elements offset index (4 aligned).
sizes	Number of uint32_t elements to scan.

5.223.2.4 dispatchIndirect() [1/3]

```
bool Tellusim::PrefixScan::dispatchIndirect (
            Compute & compute,
            Buffer & data,
            Buffer & dispatch,
            uint32_t offset,
            Flags flags = FlagNone,
            uint32_t max_size = Maxu32 )
```

dispatch single in-place indirect prefix scan

Parameters

data	Buffer of uint32_t elements to scan.
dispatch	Dispatch indirect buffer.
offset	Dispatch indirect buffer offset.
max_size	Maximum number of elements to scan.

5.223.2.5 dispatchIndirect() [2/3]

```
bool Tellusim::PrefixScan::dispatchIndirect (
            Compute & compute,
            Buffer & data,
            uint32_t count,
            Buffer & dispatch,
            uint32_t offset,
            Flags flags = FlagNone,
            uint32_t max_size = Maxu32 )
```

dispatch multiple in-place indirect prefix scans

Parameters

data	Buffer of uint32_t elements to scan.
count	Number of regions to scan.
dispatch	Dispatch indirect buffer.
offset	Dispatch indirect buffer offset.
max_size	Maximum number of elements to scan.

5.223.2.6 dispatchIndirect() [3/3]

```
bool Tellusim::PrefixScan::dispatchIndirect (
              Compute & compute,
              Buffer & data,
              Buffer & count,
              Buffer & dispatch,
              uint32_t count_offset,
              uint32_t dispatch_offset,
              Flags flags = FlagNone,
              uint32_t max_size = Maxu32 )
```

dispatch multiple in-place indirect prefix scans

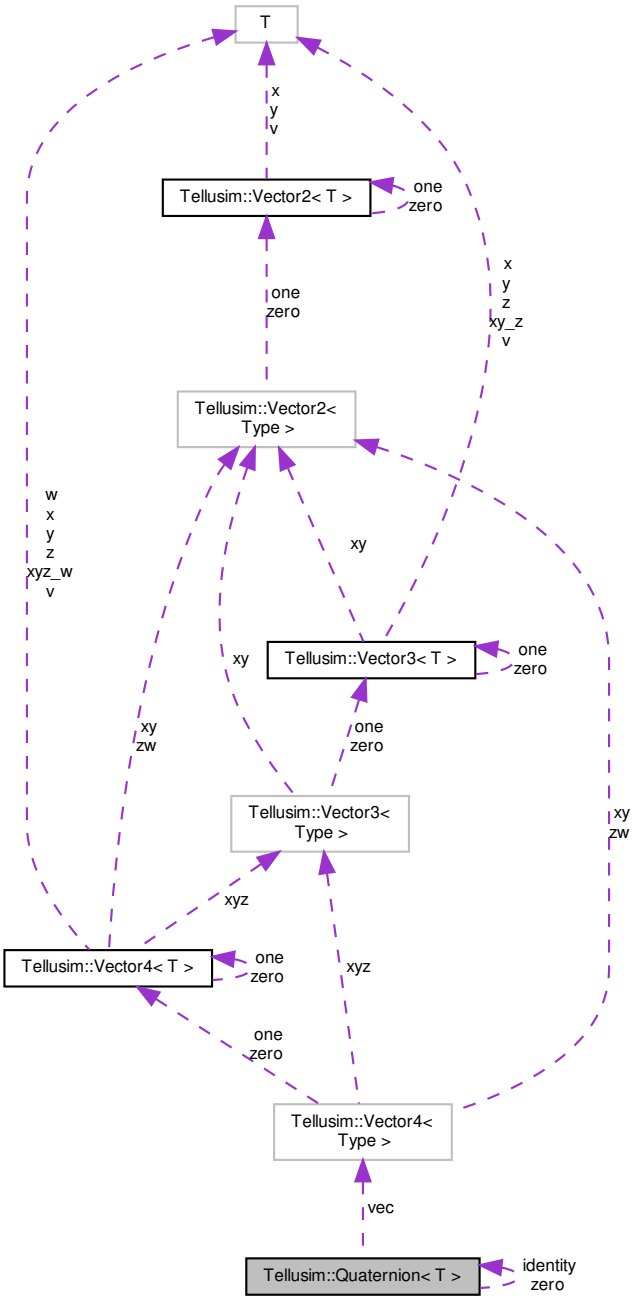
Parameters

data	Buffer of uint32_t elements to scan.
count	Count indirect buffer.
dispatch	Dispatch indirect buffer.
count_offset	Count indirect buffer offset.
dispatch_offset	Dispatch indirect buffer offset.
max_size	Maximum number of elements to scan.

5.224 Tellusim::Quaternion< T > Struct Template Reference

```
#include <math/TellusimQuaternion.h>
```

Collaboration diagram for Tellusim::Quaternion< T >:



Public Types

- enum { **Size** = 4 }
- using **Vector3** = Tellusim::Vector3< Type >
- using **Vector4** = Tellusim::Vector4< Type >
- using **Matrix4x3** = Tellusim::Matrix4x3< Type >
- using **Matrix4x4** = Tellusim::Matrix4x4< Type >

Public Member Functions

- **Quaternion** (const Quaternion &q)
- **Quaternion** (Type x, Type y, Type z, Type w)
- **Quaternion** (const Vector3 &axis, Type angle)
- **Quaternion** (const Type *q)
- **Quaternion** (const Matrix4x3 &m)
- **Quaternion** (const Matrix4x4 &m)
- **Quaternion** (const Vector4 &vector)
- template<class CType >
  Quaternion (const Quaternion< CType > &q)
- void set (Type X, Type Y, Type Z, Type W)

  update quaternion data
- void **set** (const Type *1 q)
- void **get** (Type *1 q) const
- Quaternion & operator*= (const Quaternion &q1)

  quaternion to quaternion multiplication
- Quaternion & operator+= (const Quaternion &q)

  quaternion to quaternion operators
- Quaternion & **operator-=** (const Quaternion &q)
- void setIdentity ()

  identity quaternion
- bool **isIdentity** () const
- void setRotateX (Type angle)

  rotation quaternion
- void **setRotateY** (Type angle)
- void **setRotateZ** (Type angle)
- void **setRotateXYZ** (const Vector3 &angles)
- void **setRotateZYX** (const Vector3 &angles)
- void **setRotate** (const Vector3 &axis, Type angle)
- Type **getRotateX** () const
- Type **getRotateY** () const
- Type **getRotateZ** () const
- Vector3 **getRotateXYZ** () const
- Vector3 **getRotateZYX** () const
- Vector3 **getRotate** () const
- void **getRotate** (Vector3 &axis, Type &angle) const
- void **setRotate** (Type x, Type y, Type z, Type angle)
- void set (const Vector4 &1 row_0, const Vector4 &1 row_1, const Vector4 &1 row_2)

  quaternion to/from matrix
- void **get** (Vector4 &1 row_0, Vector4 &1 row_1, Vector4 &1 row_2) const
- const Type & operator[ ] (uint32_t index) const

  quaternion data
- Type & **operator[ ]** (uint32_t index)

Static Public Member Functions

- static Quaternion **rotateX** (Type angle)
- static Quaternion **rotateY** (Type angle)
- static Quaternion **rotateZ** (Type angle)
- static Quaternion **rotateXYZ** (const Vector3 &angles)
- static Quaternion **rotateZYX** (const Vector3 &angles)
- static Quaternion **rotate** (const Vector3 &axis, Type angle)
- static Quaternion **rotateXYZ** (Type angle_x, Type angle_y, Type angle_z)
- static Quaternion **rotateZYX** (Type angle_x, Type angle_y, Type angle_z)
- static Quaternion **rotate** (Type axis_x, Type axis_y, Type axis_z, Type angle)

Public Attributes

- union {
  struct {
  Type **x**
  Type **y**
  Type **z**
  Type **w**
  }
  Vector4 **vec**
  Type **q** [Size]
  };

Static Public Attributes

- static const Quaternion zero
  default quaternions
- static const Quaternion **identity**

5.224.1 Detailed Description

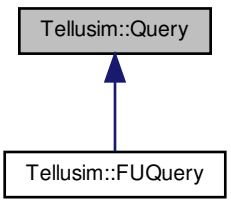
template<class T>
struct Tellusim::Quaternion< T >

Quaternion class

5.225 Tellusim::Query Class Reference

`#include <platform/TellusimQuery.h>`

Inheritance diagram for Tellusim::Query:



Classes

- struct Statistics
  statistics query

Public Member Functions

- Platform getPlatform () const
  query platform
- const char * **getPlatformName** () const
- uint32_t getIndex () const
  query device index
- void clear ()
  clear query
- bool isCreated () const
  check query
- bool **isAvailable** () const
- bool **isBegan** () const
- bool **isEnded** () const
- bool create (Type type)
  create query
- Type getType () const
  query type
- const char * **getTypeName** () const
- size_t **getTypeSize** () const
- bool **isTime** () const
- bool **isClock** () const
- bool **isSamples** () const
- bool **isSamples1** () const
- bool **isStatistics** () const
- bool **isTimeType** () const
- bool **isSamplesType** () const
- bool get (void *dest, size_t size, bool wait=true) const
  get query data
- uint64_t **getTime** (bool wait=true, bool *status=nullptr) const
- uint32_t **getSamples** (bool wait=true, bool *status=nullptr) const
- Statistics **getStatistics** (bool wait=true, bool *status=nullptr) const

Static Public Member Functions

- static const char * **getTypeName** (Type type)

5.225.1 Detailed Description

The Query class provides functionality for querying various types of data, such as time, clock, sample counts, and statistics, in a platform-specific manner. It allows for creating and managing different types of queries, checking the state of the query, and retrieving the results of the query.

5.226 Tellusim::RadixCompare< Type > Struct Template Reference

```
#include <core/TellusimSort.h>
```

5.226.1 Detailed Description

template<class Type>
struct Tellusim::RadixCompare< Type >

radixSort default compare

5.227 Tellusim::RadixCompare< float32_t > Struct Template Reference

Public Types

- enum { **Bits** = 32 }
- using **Radix** = uint32_t

Public Member Functions

- uint32_t **operator()** (float32_t value)

5.228 Tellusim::RadixCompare< int32_t > Struct Template Reference

Public Types

- enum { **Bits** = 32 }
- using **Radix** = uint32_t

Public Member Functions

- uint32_t **operator()** (int32_t value)

5.229 Tellusim::RadixCompare< uint32_t > Struct Template Reference

Public Types

- enum { **Bits** = 32 }
- using **Radix** = uint32_t

Public Member Functions

- uint32_t **operator()** (uint32_t value)

5.230 Tellusim::RadixMap< Key, Type, Size > Class Template Reference

```
#include <core/TellusimRadix.h>
```

Classes

- class ConstIterator
  Constant iterator.
- class Iterator
  Iterator.

Public Member Functions

- RadixMap ()
  constructors
- **RadixMap** (const RadixMap &)=delete
- **RadixMap** (RadixMap &&map)
- void clear ()
  clear map
- void swap (RadixMap &map)
  swap maps
- void move (RadixMap &&map)
  move map
- RadixMap & operator= (const RadixMap &)=delete
  assignment operators
- RadixMap & **operator=** (RadixMap &&map)
- Iterator append (Key value)
  append value
- Iterator **append** (Key value, const Type &data)
- bool remove (uint32_t value)
  remove value
- bool empty () const
  map info
- **operator bool** () const
- size_t **memory** () const
- uint32_t **size** () const
- Iterator find (Key value)
  map data
- ConstIterator **find** (Key value) const
- const Type & **operator[ ]** (Key value) const
- Type & **operator[ ]** (Key value)
- const Type & **get** (Key value) const
- Type & **get** (Key value)
- Iterator begin ()
  map iterators
- Iterator **back** ()
- Iterator **end** ()
- ConstIterator **begin** () const
- ConstIterator **back** () const
- ConstIterator **end** () const

5.230.1 Detailed Description

template<class Key, class Type, uint32_t Size = 32>
class Tellusim::RadixMap< Key, Type, Size >

RadixMap container

5.231 Tellusim::RadixSort Class Reference

`#include <parallel/TellusimRadixSort.h>`

Classes

- struct DispatchParameters

Public Types

- enum Mode {
  ModeSingle = 0,
  ModeMultiple,
  NumModes }
  Sort modes.
- enum Flags {
  FlagNone = 0,
  FlagSingle = (1 << ModeSingle),
  FlagMultiple = (1 << ModeMultiple),
  FlagIndirect = (1 << (NumModes + 0)),
  FlagOrder = (1 << (NumModes + 1)),
  FlagTracing = (1 << (NumModes + 2)),
  FlagScratch = (1 << (NumModes + 3)),
  FlagsAll = (FlagSingle | FlagMultiple | FlagIndirect | FlagOrder) }
  Sort flags.

Public Member Functions

- void clear ()
  clear sort
- bool isCreated (Flags flags) const
  check sort
- uint32_t getDataSize () const
  sort parameters
- uint32_t **getGroupSize** () const
- uint32_t **getSortElements** () const
- uint32_t **getUpdateElements** () const
- uint32_t **getMaxElements** () const
- uint32_t **getMaxRegions** () const
- PrefixScan **getPrefixScan** () const
- Buffer **getDataBuffer** () const
- bool create (const Device &device, Mode mode, PrefixScan &scan, uint32_t size, uint32_t groups=256, uint32_t regions=1, Async *async=nullptr)
- bool **create** (const Device &device, Flags flags, PrefixScan &scan, uint32_t size, uint32_t groups=256, uint32_t regions=1, Async *async=nullptr)
- bool dispatch (Compute &compute, Buffer &data, uint32_t keys_offset, uint32_t data_offset, uint32_t size, Flags flags=FlagNone, uint32_t bits=32)
- bool dispatch (Compute &compute, Buffer &data, uint32_t count, const uint32_t *keys_offsets, const uint32_t *data_offsets, const uint32_t *sizes, Flags flags=FlagNone, uint32_t bits=32)
- bool dispatchIndirect (Compute &compute, Buffer &data, Buffer &dispatch, uint32_t offset, Flags flags=FlagNone, uint32_t bits=32, uint32_t max_size=Maxu32)
- bool dispatchIndirect (Compute &compute, Buffer &data, uint32_t count, Buffer &dispatch, uint32_t offset, Flags flags=FlagNone, uint32_t bits=32, uint32_t max_size=Maxu32)
- bool dispatchIndirect (Compute &compute, Buffer &data, Buffer &count, Buffer &dispatch, uint32_t count_offset, uint32_t dispatch_offset, Flags flags=FlagNone, uint32_t bits=32, uint32_t max_size=Maxu32)

5.231.1 Detailed Description

The RadixSort class offers a versatile and efficient implementation of the radix sort algorithm, supporting both single and multiple array sorting, indirect dispatch modes, and auto-key sorting. It is designed for high-performance computing tasks, including sorting large datasets in simulations, rendering, and physics engines. With a range of modes and flags, the class is highly customizable to meet various sorting requirements. Its dispatch methods enable flexible and scalable sorting in complex data processing scenarios.

5.231.2 Member Function Documentation

5.231.2.1 create()

```
bool Tellusim::RadixSort::create (
              const Device & device,
              Mode mode,
              PrefixScan & scan,
              uint32_t size,
              uint32_t groups = 256,
              uint32_t regions = 1,
              Async * async = nullptr )
```

create radix sort

Parameters

scan	Prefix scan.
size	Radix sort data size.
groups	Radix sort group size.
regions	Maximum number of multiple regions.

5.231.2.2 dispatch() [1/2]

```
bool Tellusim::RadixSort::dispatch (
              Compute & compute,
              Buffer & data,
              uint32_t keys_offset,
              uint32_t data_offset,
              uint32_t size,
              Flags flags = FlagNone,
              uint32_t bits = 32 )
```

dispatch single in-place radix sort

Parameters

data	Buffer of uint32_t data elements to sort.
keys_offset	Keys elements offset index (4 aligned).
data_offset	Data elements offset index (4 aligned).
size	Number of uint32_t elements to sort.
bits	Number of key bits to sort.

5.231.2.3 dispatch() [2/2]

```
bool Tellusim::RadixSort::dispatch (
            Compute & compute,
            Buffer & data,
            uint32_t count,
            const uint32_t * keys_offsets,
            const uint32_t * data_offsets,
            const uint32_t * sizes,
            Flags flags = FlagNone,
            uint32_t bits = 32 )
```

dispatch multiple in-place radix sorts

Parameters

data	Buffer of uint32_t data elements to sort.
count	Number of regions to sort.
keys_offsets	Keys elements offset index (4 aligned).
data_offsets	Data elements offset index (4 aligned).
sizes	Number of uint32_t elements to sort.
bits	Number of key bits to sort.

5.231.2.4 dispatchIndirect() [1/3]

```
bool Tellusim::RadixSort::dispatchIndirect (
            Compute & compute,
            Buffer & data,
            Buffer & dispatch,
            uint32_t offset,
            Flags flags = FlagNone,
            uint32_t bits = 32,
            uint32_t max_size = Maxu32 )
```

dispatch single in-place indirect radix sort

Parameters

data	Buffer of uint32_t data elements to sort.
dispatch	Dispatch indirect buffer.
offset	Dispatch indirect buffer offset.
bits	Number of key bits to sort.
max_size	Maximum number of elements to sort.

5.231.2.5 dispatchIndirect() [2/3]

```
bool Tellusim::RadixSort::dispatchIndirect (
```

```
            Compute & compute,
            Buffer & data,
            uint32_t count,
            Buffer & dispatch,
            uint32_t offset,
            Flags flags = FlagNone,
            uint32_t bits = 32,
            uint32_t max_size = Maxu32 )
```

dispatch multiple in-place indirect radix sorts

Parameters

data	Buffer of uint32_t data elements to sort.
count	Number of regions to sort.
dispatch	Dispatch indirect buffer.
offset	Dispatch indirect buffer offset.
bits	Number of key bits to sort.
max_size	Maximum number of elements to sort.

5.231.2.6 dispatchIndirect() [3/3]

```
bool Tellusim::RadixSort::dispatchIndirect (
            Compute & compute,
            Buffer & data,
            Buffer & count,
            Buffer & dispatch,
            uint32_t count_offset,
            uint32_t dispatch_offset,
            Flags flags = FlagNone,
            uint32_t bits = 32,
            uint32_t max_size = Maxu32 )
```

dispatch multiple in-place indirect radix sorts

Parameters

data	Buffer of uint32_t data elements to sort.
count	Count indirect buffer.
dispatch	Dispatch indirect buffer.
count_offset	Count indirect buffer offset.
dispatch_offset	Dispatch indirect buffer offset.
bits	Number of key bits to sort.
max_size	Maximum number of elements to sort.

5.232 Tellusim::Random< Integer, Float > Struct Template Reference

```
#include <math/TellusimRandom.h>
```

Public Types

- enum { **MaxValue** = 0x0fffffff }

Public Member Functions

- Random ()

  constructor
- **Random** (const Integer &s)
- void init (const Integer &s)

  initialize random
- Integer geti32 (int32_t mask=MaxValue)

  returns an integer number
- Integer geti32 (const Integer &min, const Integer &max)

  returns an integer in [min-max] range
- Float getf32 ()

  returns a floating-point number in [0-1] range
- Float getf32 (const Float &min, const Float &max)

  returns a floating-point number in [min-max] range

Public Attributes

- Integer **seed_0**
- Integer **seed_1**

5.232.1 Detailed Description

template<class Integer = int32_t, class Float = float32_t>
struct Tellusim::Random< Integer, Float >

Linear congruential random generator

5.233 Tellusim::Rect Struct Reference

```
#include <interface/TellusimTypes.h>
```

Public Member Functions

- **Rect** (float32_t value)
- **Rect** (float32_t horizontal, float32_t vertical)
- **Rect** (float32_t left, float32_t right, float32_t bottom, float32_t top)
- bool isValid () const

  check rectangle
- **operator bool** () const
- Rect & expand (float32_t x, float32_t y)

  expand by point
- Rect & expand (const Rect &rect)

  expand by rectangle
- Rect & shrink (const Rect &rect)

  shrink by rectangle
- bool inside (float32_t x, float32_t y) const

  inside point
- bool **inside** (const Vector2f &v) const
- bool **inside** (const Rect &rect) const
- Vector2f getSize () const

  rectangle size
- float32_t **getWidth** () const
- float32_t **getHeight** () const
- Vector2f getCenter () const

  center of rectangle
- float32_t **getCenterX** () const
- float32_t **getCenterY** () const
- Rect & operator+= (const Vector2f &v)

  rectangle to vector operators
- Rect & **operator-=** (const Vector2f &v)
- Rect & operator+= (const Rect &r)

  rectangle to rectangle operators
- Rect & **operator-=** (const Rect &r)

Public Attributes

-
```
union {
  struct {
    float32_t left
    float32_t right
    float32_t bottom
    float32_t top
  }
  float32_t rect [4]
};
```

5.233.1 Detailed Description

The Rect struct defines a rectangle with properties for the left, right, bottom, and top coordinates, and provides various methods to manipulate and query the rectangle.

5.234 Tellusim::Region Struct Reference

`#include <TellusimTypes.h>`

Public Member Functions

- **Region** (uint32_t width, uint32_t height)
- **Region** (uint32_t width, uint32_t height, uint32_t depth)
- **Region** (uint32_t x, uint32_t y, uint32_t width, uint32_t height)
- **Region** (uint32_t x, uint32_t y, uint32_t z, uint32_t width, uint32_t height, uint32_t depth)
- **Region** (const Origin &origin, const Size &size)
- **Region** (const Size &size)
- Origin getOrigin () const

  region parameters
- Size **getSize** () const

Public Attributes

- uint32_t **x** = 0
- uint32_t **y** = 0
- uint32_t **z** = 0
- uint32_t **width** = 0
- uint32_t **height** = 0
- uint32_t **depth** = 0

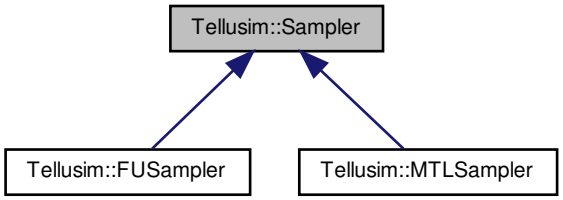
5.234.1 Detailed Description

The Region struct defines a 2D or 3D region in space, typically used to represent areas or volumes within a larger structure, such as an image, texture, or 3D object. It provides several constructors to initialize the region with specific coordinates (x, y, z) and dimensions (width, height, depth). The Region can be initialized using a variety of combinations, such as specifying just the size, the origin and size, or the full region in 3D space. Additionally, it includes methods to retrieve the Origin (the starting point of the region) and the Size (the dimensions of the region). By default, all values are set to zero.

5.235 Tellusim::Sampler Class Reference

`#include <platform/TellusimSampler.h>`

Inheritance diagram for Tellusim::Sampler:



Public Types

- enum Filter {
  FilterPoint = 0,
  FilterLinear,
  FilterBipoint,
  FilterBilinear,
  FilterTrilinear,
  NumFilters }

  Filter types.
- enum {
  MinAnisotropy = 1,
  MaxAnisotropy = 16 }

  Anisotropy range.
- enum WrapMode {
  WrapModeClamp = 0,
  WrapModeRepeat,
  WrapModeMirror,
  WrapModeBorder,
  NumWrapModes }

  Wrap modes.
- enum CompareFunc {
  CompareFuncNone = 0,
  CompareFuncEqual,
  CompareFuncLess,
  CompareFuncGreater,
  CompareFuncNotEqual,
  CompareFuncLessEqual,
  CompareFuncGreaterEqual,
  NumCompareFunctions }

  Compare functions.
- enum ReductionMode {
  ReductionModeAverage = 0,
  ReductionModeMin,
  ReductionModeMax,
  NumReductionModes }

  Reduction modes.

Public Member Functions

- Platform getPlatform () const

  sampler platform
- const char * **getPlatformName** () const
- uint32_t getIndex () const

  sampler device index
- void clear ()

  clear sampler
- bool isCreated () const

  check sampler
- bool create ()

  create sampler
- void setParameters (const Sampler &sampler)

  sampler parameters
- void setFilter (Filter filter)

filter type

- Filter **getFilter** () const
- bool **isPointFilter** () const
- void setAnisotropy (uint32_t anisotropy)

  anisotropy level
- uint32_t **getAnisotropy** () const
- bool **hasAnisotropy** () const
- void setWrapMode (WrapMode mode)

  wrapping mode
- void **setWrapMode** (WrapMode mode_s, WrapMode mode_t, WrapMode mode_r)
- void **setWrapModeS** (WrapMode mode)
- WrapMode **getWrapModeS** () const
- void **setWrapModeT** (WrapMode mode)
- WrapMode **getWrapModeT** () const
- void **setWrapModeR** (WrapMode mode)
- WrapMode **getWrapModeR** () const
- void setLod (float32_t min, float32_t max, float32_t bias)

  level of detail
- void **setLodMin** (float32_t min)
- float32_t **getLodMin** () const
- void **setLodMax** (float32_t max)
- float32_t **getLodMax** () const
- void **setLodBias** (float32_t bias)
- float32_t **getLodBias** () const
- void setBorderColor (const Color &color)

  border color
- void **setBorderColor** (float32_t r, float32_t g, float32_t b, float32_t a)
- const Color & **getBorderColor** () const
- void setCompareFunc (CompareFunc func)

  compare func
- CompareFunc **getCompareFunc** () const
- void setReductionMode (ReductionMode mode)

  reduction mode
- ReductionMode **getReductionMode** () const

5.235.1 Detailed Description

The Sampler class manages texture sampling parameters such as filter types, anisotropy, wrapping modes, level of detail (LOD), and border colors for texture mapping. It provides methods for setting and retrieving various sampling properties, including different filtering and wrapping strategies, anisotropic filtering, comparison functions, and reduction modes. The class also offers functionality to clear or create a sampler and check its creation status, which is essential for controlling the behavior of textures in rendering pipelines across different platforms and devices.

5.236 Tellusim::Scissor Struct Reference

```
#include <TellusimTypes.h>
```

Public Member Functions

- **Scissor** (int32_t width, int32_t height)
- **Scissor** (int32_t x, int32_t y, int32_t width, int32_t height)

Public Attributes

- int32_t **x** = 0
- int32_t **y** = 0
- int32_t **width** = Maxi16
- int32_t **height** = Maxi16

5.236.1 Detailed Description

The Scissor struct is used to define a clipping region, ensuring that rendering operations only affect the specified area within the defined boundaries. It is used to optimize rendering or to focus on specific regions of the screen.

5.237 Tellusim::SeparableFilter Class Reference

`#include <graphics/TellusimSeparableFilter.h>`

Public Types

- enum Mode {
  ModeHorizontal = 0,
  ModeVertical,
  NumModes }

  Filter modes.
- enum Flags {
  FlagNone = 0,
  FlagRepeat = (1 << 0),
  FlagZero = (1 << 1),
  DefaultFlags = FlagNone }

  Filter flags.

Public Member Functions

- void clear ()

  clear filter
- bool isCreated (Format format, uint32_t size) const

  check filter
- void setInputSource (Mode mode, const char *src)
- String **getInputSource** (Mode mode) const
- void setOutputSource (Mode mode, const char *src)
- String **getOutputSource** (Mode mode) const
- bool create (const Device &device, Format format, uint32_t size, Flags flags=DefaultFlags)
- void setWeights (Mode mode, const Array< Vector4f > &weights, bool normalize=false)

  filter weights
- void **setWeights** (Mode mode, const Array< float32_t > &weights, bool normalize=false)
- void setGaussianWeights (uint32_t size, const Vector4f &sigma)

  Gaussian filter weights.
- void **setGaussianWeights** (uint32_t size, float32_t sigma)
- void setSobelXWeights (uint32_t size)

  Sobel filter weights.
- void **setSobelYWeights** (uint32_t size)

- void setBoxWeights (uint32_t size)

  box filter weights
- bool dispatch (Compute &compute, Mode mode, uint32_t size, Texture &dest, Texture &src, const Slice &dest_slice, const Slice &src_slice, const Vector4f &values=Vector4f::zero) const
- bool **dispatch** (Compute &compute, Mode mode, uint32_t size, Texture &dest, Texture &src, const Slice &src_slice, const Vector4f &values=Vector4f::zero) const
- bool **dispatch** (Compute &compute, Mode mode, uint32_t size, Texture &dest, Texture &src, const Vector4f &values=Vector4f::zero) const

5.237.1 Detailed Description

The SeparableFilter class implements a flexible GPU-based filtering system that applies separable convolution operations in horizontal and vertical directions using compute shaders. It allows customization of shader code for input and output stages, supports various filter types including Gaussian, Sobel, and box filters, and enables users to define or normalize custom weights. Filters can be configured with different border handling modes and dispatched efficiently on textures with optional slice and parameter control.

5.237.2 Member Function Documentation

5.237.2.1 setInputSource()

```
void Tellusim::SeparableFilter::setInputSource (
            Mode mode,
            const char * src )
```

input shader source

Parameters

src	Shader source.

5.237.2.2 setOutputSource()

```
void Tellusim::SeparableFilter::setOutputSource (
            Mode mode,
            const char * src )
```

output shader source

Parameters

src	Shader source.

5.237.2.3 create()

```
bool Tellusim::SeparableFilter::create (
             const Device & device,
             Format format,
             uint32_t size,
             Flags flags = DefaultFlags )
```

create filter

Parameters

format	Texture format.
size	Filter size in pixels, the actual filter size is (size ∗ 2 + 1).

5.237.2.4 dispatch()

```
bool Tellusim::SeparableFilter::dispatch (
             Compute & compute,
             Mode mode,
             uint32_t size,
             Texture & dest,
             Texture & src,
             const Slice & dest_slice,
             const Slice & src_slice,
             const Vector4f & values = Vector4f::zero ) const
```

dispatch separable filter

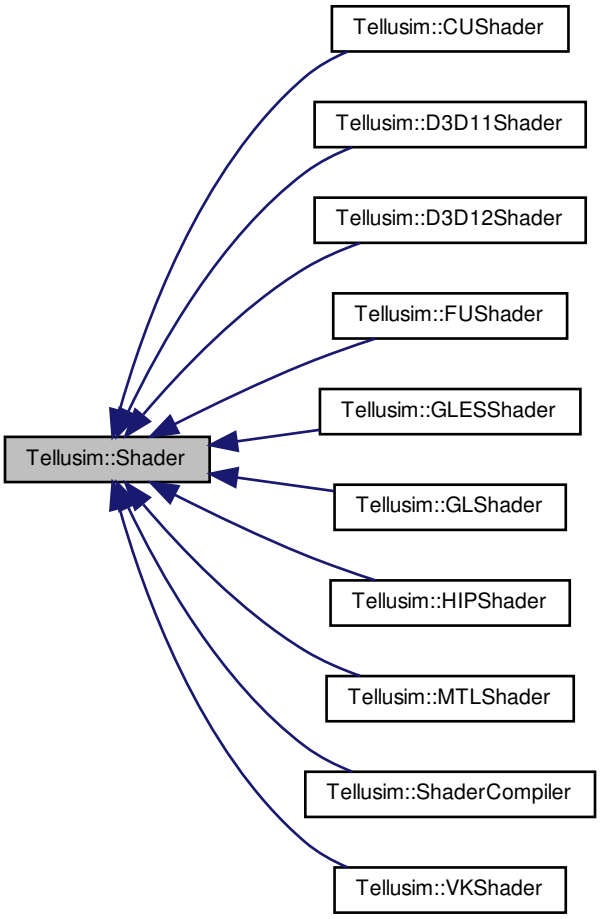
Parameters

dest	Destination texture.
src	Source texture.
dest_slice	Destination texture slice.
src_slice	Source texture slice.
values	Filter parameters available for the source blocks.

5.238 Tellusim::Shader Class Reference

```
#include <platform/TellusimShader.h>
```

Inheritance diagram for Tellusim::Shader:



Public Types

- enum Mask {
  MaskNone = 0,
  MaskVertex = (1 << TypeVertex),
  MaskControl = (1 << TypeControl),
  MaskEvaluate = (1 << TypeEvaluate),
  MaskGeometry = (1 << TypeGeometry),
  MaskFragment = (1 << TypeFragment),
  MaskCompute = (1 << TypeCompute),
  MaskTask = (1 << TypeTask),
  MaskMesh = (1 << TypeMesh),
  MaskRayGen = (1 << TypeRayGen),
  MaskRayMiss = (1 << TypeRayMiss),
  MaskClosest = (1 << TypeClosest),
  MaskFirstHit = (1 << TypeFirstHit),
  MaskIntersection = (1 << TypeIntersection),
  MaskCallable = (1 << TypeCallable),

MaskVertexFragment = (MaskVertex | MaskFragment),
MaskGraphics = (MaskVertex | MaskControl | MaskEvaluate | MaskGeometry | MaskFragment),
MaskTracing = (MaskRayGen | MaskRayMiss | MaskClosest | MaskFirstHit | MaskIntersection | MaskCallable),
MaskAll = (MaskGraphics | MaskCompute | MaskTask | MaskMesh | MaskTracing) }

Shader masks.

Public Member Functions

- Platform getPlatform () const
  shader platform
- const char * **getPlatformName** () const
- uint32_t getIndex () const
  shader device index
- void clear ()
  clear shader
- bool isCreated () const
  check shader
- bool saveState (Stream &stream) const
  shader parameters
- Type getType () const
  shader type
- const char * **getTypeName** () const
- bool **isVertex** () const
- bool **isControl** () const
- bool **isEvaluate** () const
- bool **isGeometry** () const
- bool **isFragment** () const
- bool **isCompute** () const
- bool **isTask** () const
- bool **isMesh** () const
- bool **isRayGen** () const
- bool **isRayMiss** () const
- bool **isClosest** () const
- bool **isFirstHit** () const
- bool **isIntersection** () const
- bool **isCallable** () const
- bool **isGraphicsType** () const
- bool **isTessellationType** () const
- bool **isTracingType** () const
- bool **isMeshType** () const
- String getName () const
  shader name
- String **getMacros** () const
- void setSamplerOffset (int32_t offset)
  shader samplers
- int32_t **getSamplerOffset** () const
- void setTextureOffset (int32_t offset)
  shader textures
- int32_t **getTextureOffset** () const
- void setSurfaceOffset (int32_t offset)
  shader surfaces

- int32_t **getSurfaceOffset** () const
- void setUniformOffset (int32_t offset)

  shader uniforms
- int32_t **getUniformOffset** () const
- void setStorageOffset (int32_t offset)

  shader storages
- int32_t **getStorageOffset** () const
- void setTracingOffset (int32_t offset)

  shader tracings
- int32_t **getTracingOffset** () const
- void setTexelOffset (int32_t offset)

  shader texels
- int32_t **getTexelOffset** () const
- void setTableOffset (int32_t offset)

  shader tables
- int32_t **getTableOffset** () const
- void setPatchSize (uint32_t size)

  shader patch size
- uint32_t **getPatchSize** () const
- void setInputSize (uint32_t size)

  shader input size
- uint32_t **getInputSize** () const
- void setOutputSize (uint32_t size)

  shader output size
- uint32_t **getOutputSize** () const
- bool load (Type type, const char ∗name, const char ∗format,...) 1(4

  create native shader
- bool bool **create** (Type type, const char ∗src, const char ∗format,...) 1(4
- bool bool bool **load** (Type type, const char ∗name, const String &macros=String::null, const char ∗∗includes=nullptr, uint32_t size=0)
- bool **create** (Type type, const char ∗src, const String &macros=String::null, const char ∗∗includes=nullptr, uint32_t size=0)
- bool loadGLSL (Type type, const char ∗name, const char ∗format,...) 1(4

  create GLSL shader
- bool bool **createGLSL** (Type type, const char ∗src, const char ∗format,...) 1(4
- bool bool bool **loadGLSL** (Type type, const char ∗name, const String &macros=String::null, const char ∗∗includes=nullptr, uint32_t size=0)
- bool **createGLSL** (Type type, const char ∗src, const String &macros=String::null, const char ∗∗includes=nullptr, uint32_t size=0)
- bool loadSPIRV (Type type, const char ∗name)

  create SPIR-V shader
- bool **createSPIRV** (Type type, const Array< uint32_t > &data)

Static Public Member Functions

- static const char ∗ **getTypeName** (Type type)
- static bool hasCache ()

  global shader cache
- static bool **setCache** (const char ∗name)
- static bool **loadCache** (const String &hash, Stream &stream)
- static bool **saveCache** (const String &hash, Stream &stream)
- static void **clearCache** ()

- static bool isMacro (const char *name)

  global macro definitions
- static bool **setMacro** (const char *name, int32_t value)
- static bool **setMacro** (const char *name, uint32_t value)
- static bool **setMacro** (const char *name, const char *value=nullptr)
- static bool **setMacros** (const char *macros)
- static bool **removeMacro** (const char *name)
- static void **clearMacros** ()
- static bool isInclude (const char *name)

  global include definitions
- static bool **setInclude** (const char *name, const String &src)
- static bool **removeInclude** (const char *name)
- static void **clearIncludes** ()
- static String preprocessor (const char *src, const char *format,...) 1(2

  global macro preprocessor
- static String static String **preprocessor** (const char *src, const String &macros=String::null, const char **includes=nullptr, uint32_t size=0)

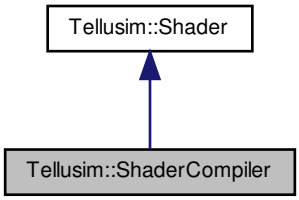
5.238.1 Detailed Description

The Shader class represents a shader program, enabling the creation, loading, and management of shaders in various types and formats like native, GLSL, and SPIR-V. It provides functionality to configure and retrieve properties such as shader type, name, macros, and resource offsets for textures, surfaces, and uniforms. The class supports shader creation and compilation, management of global shader caches, and handling global macros and includes.

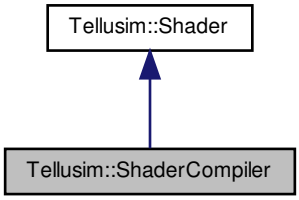
5.239 Tellusim::ShaderCompiler Class Reference

```
#include <platform/TellusimShader.h>
```

Inheritance diagram for Tellusim::ShaderCompiler:



Collaboration diagram for Tellusim::ShaderCompiler:



Public Types

- enum **Flags** {
  FlagNone = 0,
  FlagMSLIndirect = (1 << 0) }

Public Member Functions

- void setFlags (Flags flags)
  shader flags
- Flags **getFlags** () const
- bool getBinary (Stream &stream, Platform platform=PlatformUnknown) const
  shader binary
- String getSource (Platform platform=PlatformUnknown) const
  shader source

Additional Inherited Members

5.239.1 Detailed Description

The ShaderCompiler class extends the Shader class to provide additional functionality for compiling and managing shaders. The class supports retrieving the compiled shader binary for a specific platform and obtaining the shader source code for the given platform.

5.240 Tellusim::Size Struct Reference

`#include <TellusimTypes.h>`

Public Member Functions

- **Size** (uint32_t size)
- **Size** (uint32_t width, uint32_t height)
- **Size** (uint32_t width, uint32_t height, uint32_t depth)

Public Attributes

- uint32_t **width** = 0
- uint32_t **height** = 0
- uint32_t **depth** = 0

5.240.1 Detailed Description

The Size struct represents a 2D or 3D size, commonly used for defining dimensions such as width, height, and depth. It provides multiple constructors to initialize the size in 2D (with width and height), 3D (with width, height, and depth), or as a uniform size for all dimensions (when a single size value is provided for width and height). By default, all values are set to zero.

5.241 Tellusim::Slice Struct Reference

`#include <TellusimTypes.h>`

Public Member Functions

- **Slice** (const Face &f)
- **Slice** (const Layer &l)
- **Slice** (const Mipmap &m)
- **Slice** (const Layer &l, const Face &f)
- **Slice** (const Face &f, const Mipmap &m)
- **Slice** (const Layer &l, const Mipmap &m)
- **Slice** (const Layer &l, const Face &f, const Mipmap &m)
- bool hasFaces () const

  slice parameters
- bool **hasLayers** () const
- bool **hasMipmaps** () const
- Face **getFace** () const
- Layer **getLayer** () const
- Mipmap **getMipmap** () const
- Slice setFace (uint32_t base, uint32_t size=1) const

  set slice parameters
- Slice **setLayer** (uint32_t base, uint32_t size=1) const
- Slice **setMipmap** (uint32_t base, uint32_t size=1) const
- Slice setSize (const Slice &s) const

  replace slice sizes
- Slice addBase (const Slice &s) const

  increment slice bases

Public Attributes

- uint32_t **face** = 0
- uint32_t **faces** = 1
- uint32_t **layer** = 0
- uint32_t **layers** = 1
- uint32_t **mipmap** = 0
- uint32_t **mipmaps** = 1

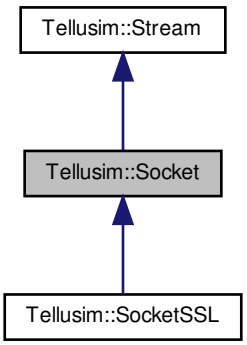
5.241.1 Detailed Description

The Slice struct defines a specific subset of a texture or image, using faces, layers, and mipmaps. It can be configured to represent a single face, layer, or mipmap level of an image. The struct includes methods for checking if the slice has multiple faces, layers, or mipmaps, and for adjusting or replacing slice sizes and bases.

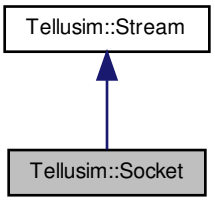
5.242 Tellusim::Socket Class Reference

`#include <core/TellusimSocket.h>`

Inheritance diagram for Tellusim::Socket:



Collaboration diagram for Tellusim::Socket:



Public Member Functions

- **Socket** (Type type=TypeStream)
- bool open (uint16_t port, uint16_t num=32)

  open/close socket

- bool **open** (const char *name, uint16_t port)
- bool **open** (const String &name, uint16_t port)
- void **close** ()
- bool connect (uint32_t sec, uint32_t usec=0)

  stream socket
- virtual bool **accept** (Socket &socket)
- bool select (uint32_t sec, uint32_t usec=0)

  socket operations
- bool setTimeout (uint32_t sec)

  socket timeout
- uint32_t **getTimeout** () const
- bool setBlock (bool block)

  socket blocking
- bool **getBlock** () const
- bool setDelay (bool delay)

  sockey delay
- bool **getDelay** () const
- void setName (const char *name)

  socket parameters
- void **setName** (const String &name)
- uint16_t **getPort** () const
- int32_t **getFD** () const
- Type **getType** () const

Static Public Member Functions

- static String getAddress (const char *delimiter=nullptr)

  socket utils

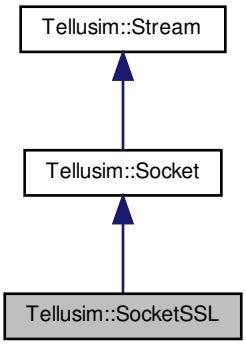
5.242.1 Detailed Description

The Socket class extends the Stream class and provides an abstraction for working with network sockets, supporting both stream and datagram socket types. It offers functionality for opening and closing sockets, connecting to remote addresses, and accepting incoming connections. The class supports socket-specific operations such as selecting sockets for activity, setting timeouts, and configuring blocking and delay behaviors.

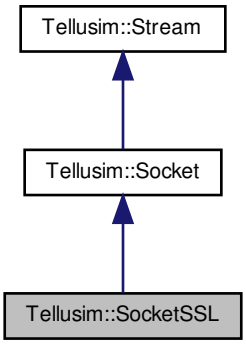
5.243 Tellusim::SocketSSL Class Reference

```
#include <core/TellusimSocket.h>
```

Inheritance diagram for Tellusim::SocketSSL:



Collaboration diagram for Tellusim::SocketSSL:



Public Member Functions

- bool handshake (const char *name=nullptr)

  stream socket
- virtual bool **accept** (SocketSSL &socket)
- virtual bool **accept** (Socket &socket)
- bool load (Stream &stream)

  socket certificate
- bool **load** (const char *name)
- bool **load** (const String &name)
- bool isConnected () const

  socket status

Additional Inherited Members

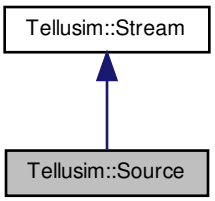
5.243.1 Detailed Description

The SocketSSL class extends the Socket class and provides additional functionality for handling secure SSL/T↩ LS connections. It offers methods for performing the SSL handshake, which establishes a secure connection, as well as accepting SSL-based socket connections. The class includes capabilities for loading SSL certificates from streams or file names, enabling secure communication through encryption.

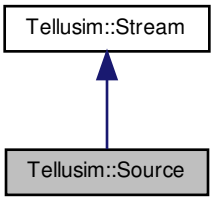
5.244 Tellusim::Source Class Reference

`#include <core/TellusimSource.h>`

Inheritance diagram for Tellusim::Source:



Collaboration diagram for Tellusim::Source:



Public Types

- using IsCallback = bool(const char ∗name, void ∗data)

  source callback
- using **OpenCallback** = Stream(const char ∗name, void ∗data)

Public Member Functions

- **Source** (const uint8_t *data, size_t size, const char *name=nullptr)
- bool open (const char *name, bool callback=true, bool write=false)

  open/close source
- bool **open** (const String &name, bool callback=true, bool write=false)
- void **close** ()
- void setName (const char *name, size_t offset, size_t size)

  source name
- void **setName** (const String &name, size_t offset, size_t size)
- void setData (const uint8_t *data, size_t size, const char *name=nullptr)

  source data

Static Public Member Functions

- static bool isSource (const char *name)

  source utils
- static bool **isSource** (const String &name)
- static uint64_t **getMTime** (const char *name)
- static size_t **getSize** (const char *name)
- static void **setCallback** (OpenCallback *open_func, void *data=nullptr)
- static void **setCallback** (OpenCallback *open_func, IsCallback *is_func, void *data=nullptr)
- static OpenCallback * **getOpenCallback** ()
- static IsCallback * **getIsCallback** ()
- static void * **getCallbackData** ()

5.244.1 Detailed Description

The Source class extends the Stream class and provides functionality for managing data from memory-mapped files or raw memory pointers. It supports operations for opening and closing sources, setting data, and querying the source name and size. The class also allows defining custom callbacks for opening and validating user-defined sources.

5.245 Tellusim::SpatialGrid Class Reference

```
#include <parallel/TellusimSpatialGrid.h>
```

Classes

- struct DispatchParameters

Public Member Functions

- void clear ()

  clear grid
- bool isCreated () const

  check grid
- uint32_t getGroupSize () const

  grid parameters
- RadixSort **getRadixSort** () const
- bool create (const Device &device, RadixSort &sort, uint32_t groups=256)

  create spatial grid
- bool dispatch (Compute &compute, Buffer &data, uint32_t offset, uint32_t size, uint32_t bits=32)
- bool dispatchIndirect (Compute &compute, Buffer &data, Buffer &dispatch, uint32_t offset, uint32_t max_↩ size=Maxu32)

5.245.1 Detailed Description

The SpatialGrid class provides a framework for managing and processing spatial data using a grid-based approach. It supports efficient creation, dispatch, and manipulation of the spatial grid, utilizing radix sorting and customizable dispatch operations. With its methods for grid creation, dispatching grid generation, and managing the spatial elements, the class is well-suited for applications that require fast, scalable spatial partitioning and computation, such as in graphics rendering, simulations, and physics-based applications.

5.245.2 Member Function Documentation

5.245.2.1 dispatch()

```
bool Tellusim::SpatialGrid::dispatch (
             Compute & compute,
             Buffer & data,
             uint32_t offset,
             uint32_t size,
             uint32_t bits = 32 )
```

dispatch spatial grid generation

Parameters

data	Spatial grid of uint32_t elements.
offset	Spatial grid offset index.
size	Number of spatial elements.
bits	Number of hash bits to sort.

5.245.2.2 dispatchIndirect()

```
bool Tellusim::SpatialGrid::dispatchIndirect (
             Compute & compute,
             Buffer & data,
             Buffer & dispatch,
             uint32_t offset,
             uint32_t max_size = Maxu32 )
```

dispatch spatial tree generation

Parameters

data	Spatial grid of uint32_t elements.
dispatch	Dispatch indirect buffer.
offset	Dispatch indirect buffer offset.
max_size	Maximum number of spatial elements.

5.246 Tellusim::SpatialTree Class Reference

`#include <parallel/TellusimSpatialTree.h>`

Classes

- struct DispatchParameters
- struct LeafNodef16
- struct Node

Public Types

- enum Mode {
  ModeSingle = 0,
  ModeMultiple,
  NumModes }

  Tree modes.
- enum Hash {
  HashXYZ10,
  HashXYZ9,
  HashXYZ8,
  HashXY15,
  HashXY14,
  HashXY12,
  HashXY10 }

  Hash modes.
- enum Flags {
  FlagNone = 0,
  FlagSingle = (1 << ModeSingle),
  FlagMultiple = (1 << ModeMultiple),
  FlagUpdate = (1 << (NumModes + 0)),
  FlagOptimize = (1 << (NumModes + 1)),
  FlagTracing = (1 << (NumModes + 2)),
  FlagScratch = (1 << (NumModes + 3)),
  FlagAtomic = (1 << (NumModes + 4)),
  FlagLeafNodef16 = (1 << (NumModes + 5)),
  FlagSpatialData = (1 << (NumModes + 6)),
  FlagSingleUpdate = (FlagSingle | FlagUpdate),
  FlagMultipleUpdate = (FlagMultiple | FlagUpdate),
  FlagSingleOptimize = (FlagSingle | FlagOptimize),
  FlagMultipleOptimize = (FlagMultiple | FlagOptimize),
  FlagsAll = (FlagSingle | FlagMultiple),
  FlagsAllOptimize = (FlagsAll | FlagOptimize) }

  Tree flags.

Public Member Functions

- void clear ()

  clear tree
- bool isCreated (Flags flags) const

  check tree
- uint32_t getGroupSize () const

tree parameters

- uint32_t **getBoundsNodes** () const
- uint32_t **getMaxNodes** () const
- uint32_t **getMaxRegions** () const
- RadixSort **getRadixSort** () const
- Buffer **getHashBuffer** () const
- Buffer **getParentsBuffer** () const
- Buffer **getCounterBuffer** () const
- bool create (const Device &device, Mode mode, RadixSort &sort, uint32_t size, uint32_t groups=256, uint32_t regions=1, Async *async=nullptr)
- bool **create** (const Device &device, Flags flags, RadixSort &sort, uint32_t size, uint32_t groups=256, uint32_t regions=1, Async *async=nullptr)
- bool dispatch (Compute &compute, Hash hash, Buffer &nodes, uint32_t offset, uint32_t size, Flags flags=FlagNone)
- bool dispatch (Compute &compute, Hash hash, Buffer &nodes, uint32_t count, const uint32_t *offsets, const uint32_t *sizes, Flags flags=FlagNone)
- bool dispatchIndirect (Compute &compute, Hash hash, Buffer &nodes, Buffer &dispatch, uint32_t offset, uint32_t max_size=Maxu32, Flags flags=FlagNone)
- bool dispatchIndirect (Compute &compute, Hash hash, Buffer &nodes, uint32_t count, Buffer &dispatch, uint32_t offset, uint32_t max_size=Maxu32, Flags flags=FlagNone)

5.246.1 Detailed Description

The SpatialTree class offers a powerful framework for creating and managing spatial data structures like bounding volume hierarchies (BVH). It supports flexible modes for handling single and multiple trees, different hash modes for efficient spatial partitioning, and various flags to control optimizations, memory management, and operation behaviors. Its methods allow for the creation, dispatch, and manipulation of trees, making it well-suited for high-performance applications such as graphics rendering, physics simulations, and spatial data processing.

5.246.2 Member Function Documentation

5.246.2.1 create()

```
bool Tellusim::SpatialTree::create (
            const Device & device,
            Mode mode,
            RadixSort & sort,
            uint32_t size,
            uint32_t groups = 256,
            uint32_t regions = 1,
            Async * async = nullptr )
```

create spatial tree

Parameters

sort	Radix sort.
size	Spatial tree data size.
groups	Spatial tree group size.
regions	Maximum number of multiple regions.

5.246.2.2 dispatch() [1/2]

```
bool Tellusim::SpatialTree::dispatch (
            Compute & compute,
            Hash hash,
            Buffer & nodes,
            uint32_t offset,
            uint32_t size,
            Flags flags = FlagNone )
```

dispatch single in-place spatial tree generation

Parameters

hash	Spatial tree hash mode.
nodes	Buffer of spatial tree nodes.
offset	Spatial tree nodes offset index.
size	Number of spatial elements.

5.246.2.3 dispatch() [2/2]

```
bool Tellusim::SpatialTree::dispatch (
            Compute & compute,
            Hash hash,
            Buffer & nodes,
            uint32_t count,
            const uint32_t * offsets,
            const uint32_t * sizes,
            Flags flags = FlagNone )
```

dispatch multiple in-place spatial tree generation

Parameters

hash	Spatial tree hash mode.
nodes	Buffer of spatial tree nodes.
count	Number of regions to create.
offsets	Spatial tree nodes offset index.
sizes	Number of spatial elements.

5.246.2.4 dispatchIndirect() [1/2]

```
bool Tellusim::SpatialTree::dispatchIndirect (
            Compute & compute,
            Hash hash,
```

```
            Buffer & nodes,
            Buffer & dispatch,
            uint32_t offset,
            uint32_t max_size = Maxu32,
            Flags flags = FlagNone )
```

dispatch single in-place spatial tree generation

Parameters

hash	Spatial tree hash mode.
nodes	Buffer of spatial tree nodes.
dispatch	Dispatch indirect buffer.
offset	Dispatch indirect buffer offset.
max_size	Maximum number of spatial elements.

5.246.2.5 dispatchIndirect() [2/2]

```
bool Tellusim::SpatialTree::dispatchIndirect (
            Compute & compute,
            Hash hash,
            Buffer & nodes,
            uint32_t count,
            Buffer & dispatch,
            uint32_t offset,
            uint32_t max_size = Maxu32,
            Flags flags = FlagNone )
```

dispatch multiple in-place spatial tree generation

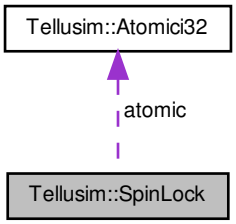
Parameters

hash	Spatial tree hash mode.
nodes	Buffer of spatial tree nodes.
count	Number of regions to create.
dispatch	Dispatch indirect buffer.
offset	Dispatch indirect buffer offset.
max_size	Maximum number of spatial elements.

5.247 Tellusim::SpinLock Struct Reference

```
#include <core/TellusimAtomic.h>
```

Collaboration diagram for Tellusim::SpinLock:



Public Member Functions

- **operator bool** ()
- void **clear** ()
- void **signal** ()
- void **lock** ()
- void **unlock** ()
- bool **check** ()
- int32_t **get** ()

Public Attributes

- Atomici32 **atomic**

5.247.1 Detailed Description

SpinLock class

5.248 Tellusim::Query::Statistics Struct Reference

statistics query

`#include <platform/TellusimQuery.h>`

Public Attributes

- uint64_t **num_vertices**
- uint64_t **num_primitives**
- uint64_t **vertex_invocations**
- uint64_t **control_invocations**
- uint64_t **evaluate_invocations**
- uint64_t **geometry_invocations**
- uint64_t **geometry_primitives**
- uint64_t **fragment_invocations**
- uint64_t **compute_invocations**
- uint64_t **clipping_invocations**
- uint64_t **clipping_primitives**

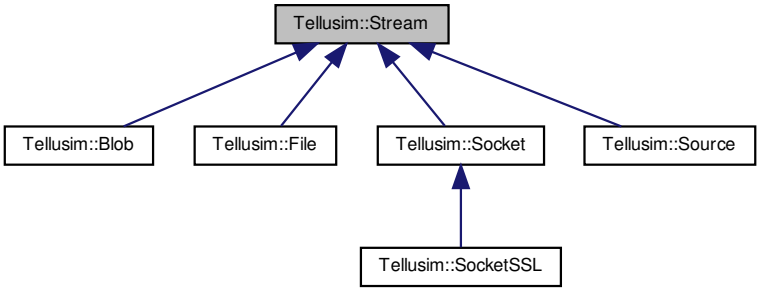
5.248.1 Detailed Description

statistics query

5.249 Tellusim::Stream Class Reference

`#include <core/TellusimStream.h>`

Inheritance diagram for Tellusim::Stream:



Public Member Functions

- Stream move ()
  move stream
- bool isOpened () const
  stream status
- bool **isMapped** () const
- bool **isAvailable** () const
- size_t **getSize** () const
- String **getName** () const
- size_t tell ()
  stream position
- bool **seek** (size_t offset)
- bool **seekBack** (size_t offset)
- bool **seekCur** (int64_t offset)
- const uint8_t * getData () const
  stream data
- size_t read (void *dest, size_t size)
  read/write stream
- size_t **write** (const void *src, size_t size)
- bool **flush** ()
- bool puts (const char *str)
  unterminated strings
- bool **puts** (const String &str)
- bool **vprintf** (const char *format, va_list args)

- bool **printf** (const char *format,...) 1(2
- template<class... List>
  bool bool **tprintf** (const char *format, List... Args)
- String **gets** (bool *status=nullptr)
- int8_t readi8 (bool *status=nullptr)

  8-bit integer numbers
- bool **writei8** (int8_t value)
- uint8_t **readu8** (bool *status=nullptr)
- bool **writeu8** (uint8_t value)
- int16_t readi16 (bool *status=nullptr)

  16-bit integer numbers
- bool **writei16** (int16_t value)
- uint16_t **readu16** (bool *status=nullptr)
- bool **writeu16** (uint16_t value)
- int32_t readi32 (bool *status=nullptr)

  32-bit integer numbers
- bool **writei32** (int32_t value)
- uint32_t **readu32** (bool *status=nullptr)
- bool **writeu32** (uint32_t value)
- int64_t readi64 (bool *status=nullptr)

  64-bit integer numbers
- bool **writei64** (int64_t value)
- uint64_t **readu64** (bool *status=nullptr)
- bool **writeu64** (uint64_t value)
- float32_t readf32 (bool *status=nullptr)

  32-bit floating-point numbers
- bool **writef32** (float32_t value)
- float64_t readf64 (bool *status=nullptr)

  64-bit floating-point numbers
- bool **writef64** (float64_t value)
- int32_t readi32e (bool *status=nullptr)

  encoded 32-bit integer numbers
- bool **writei32e** (int32_t value)
- uint32_t **readu32e** (bool *status=nullptr)
- bool **writeu32e** (uint32_t value)
- int64_t readi64e (bool *status=nullptr)

  encoded 64-bit integer numbers
- bool **writei64e** (int64_t value)
- uint64_t **readu64e** (bool *status=nullptr)
- bool **writeu64e** (uint64_t value)
- String readString (bool *status=nullptr)

  string with encoded 32-bit integer length
- bool **writeString** (const String &str)
- bool **writeString** (const char *str)
- String readString (char term, bool *status=nullptr, uint32_t size=Maxu32)

  string with terminated character
- bool **writeString** (const String &str, char term)
- bool **writeString** (const char *str, char term)
- String readToken (bool *status=nullptr)

  read token
- bool **readToken** (String &dest, bool clear=true)
- String readLine (bool *status=nullptr)

  read line

- bool **readLine** (String &dest, bool empty=false, bool clear=true)
- size_t readStream (Stream &dest, size_t size=0, bool *status=nullptr)

  copy streams
- size_t **writeStream** (Stream &src, size_t size=0, bool *status=nullptr)
- size_t readZip (void *dest, size_t size)

  zip streams with encoded 32-bit length
- size_t **writeZip** (const void *src, size_t size, int32_t level=-1)
- size_t **writeZipFast** (const void *src, size_t size)
- size_t **writeZipBest** (const void *src, size_t size)
- size_t readLz4 (void *dest, size_t size)

  lz4 streams with encoded 32-bit length
- size_t **writeLz4** (const void *src, size_t size, int32_t level=-1)
- size_t **writeLz4Fast** (const void *src, size_t size)
- size_t **writeLz4Best** (const void *src, size_t size)
- size_t decodeZip (Stream &src, size_t size=0, bool *status=nullptr, int32_t window=15)

  zip streams
- size_t **encodeZip** (Stream &dest, size_t size=0, bool *status=nullptr, int32_t level=-1)
- size_t **encodeZipFast** (Stream &dest, size_t size=0, bool *status=nullptr)
- size_t **encodeZipBest** (Stream &dest, size_t size=0, bool *status=nullptr)
- size_t decodeLz4 (Stream &src, size_t size=0, bool *status=nullptr)

  lz4 streams
- size_t **encodeLz4** (Stream &dest, size_t size=0, bool *status=nullptr, int32_t level=-1)
- size_t **encodeLz4Fast** (Stream &dest, size_t size=0, bool *status=nullptr)
- size_t **encodeLz4Best** (Stream &dest, size_t size=0, bool *status=nullptr)

5.249.1 Detailed Description

The Stream class provides an abstraction for reading from and writing to streams of data, supporting various formats and encoding methods. It offers a set of functionalities to manage the stream status, position, and data, including operations for checking if a stream is open, mapped, or available, and querying its size and name. The class supports a wide range of read and write operations for different data types, from basic integers and floating-point numbers to more complex encoded formats, and strings. Additionally, the Stream class supports compression and decompression of data using the ZIP and LZ4 algorithms, allowing for both reading and writing compressed streams with different encoding levels. Whether dealing with raw data, compressed streams, or formatted strings, this class offers flexible and efficient ways to manage data flow within applications.

5.250 Tellusim::String Class Reference

`#include <core/TellusimString.h>`

Collaboration diagram for Tellusim::String:

Public Member Functions

- **String** (uint32_t size, char c=0)
- **String** (const char *str, uint32_t length=Maxu32)
- **String** (const wchar_t *str, uint32_t length=Maxu32)
- **String** (const uint32_t *str, uint32_t length=Maxu32)
- String & reserve (uint32_t size, bool discard=false)

  resize string
- String & **resize** (uint32_t size, char c=0, bool reserve=false)
- void release ()

  clear string
- void **clear** ()
- void copy (const char *str, uint32_t length=Maxu32)

  copy string
- void **copy** (const wchar_t *str, uint32_t length=Maxu32)
- void **copy** (const uint32_t *str, uint32_t length=Maxu32)
- void **copy** (const String &string, uint32_t length=Maxu32)
- String & **operator=** (const char *str)
- String & append (char c)

  append string
- String & **append** (const char *str, uint32_t length=Maxu32)
- String & **append** (const String &string, uint32_t length=Maxu32)
- String & **operator+=** (char c)
- String & **operator+=** (const char *str)
- String & **operator+=** (const String &string)
- String & insert (uint32_t pos, const char *str, uint32_t length=Maxu32)

  insert string
- String & **insert** (uint32_t pos, const String &string, uint32_t length=Maxu32)
- String & removeBack (uint32_t length=1)

  remove string
- String & **remove** (uint32_t pos, uint32_t length=1)
- String & reverse (uint32_t pos=0, uint32_t length=Maxu32)

  reverse string
- uint32_t size () const

  string info
- bool **empty** () const
- **operator bool** () const
- char * get ()

  string data
- const char * **get** () const
- char & **get** (uint32_t index)
- char **get** (uint32_t index) const
- char & **operator[ ]** (uint32_t index)
- char **operator[ ]** (uint32_t index) const
- uint32_t find (char c, uint32_t pos=0) const

  find ascii character
- uint32_t **rfind** (char c, uint32_t pos=Maxu32) const
- uint32_t **count** (char c, uint32_t pos=0) const
- uint32_t find (const char *str, uint32_t pos=0) const

  find part of the string
- uint32_t **rfind** (const char *str, uint32_t pos=Maxu32) const
- uint32_t **count** (const char *str, uint32_t pos=0) const
- bool begins (const char *str, uint32_t length=Maxu32, uint32_t pos=0) const

compare begin of the string

- bool contains (const char *str, uint32_t length=Maxu32, uint32_t pos=0) const

  contains the string
- bool match (const char *str, uint32_t length=Maxu32, uint32_t pos=0) const

  matches the pattern
- int32_t compare (const char *str, uint32_t pos=0) const

  compare strings
- uint32_t distance (const char *str, bool scan=false, uint32_t pos=0) const

  distance between strings
- const char * begin () const

  string iterators
- const char * **end** () const
- char front (uint32_t index=0) const

  string elements
- char **back** (uint32_t index=0) const
- char & **front** (uint32_t index=0)
- char & **back** (uint32_t index=0)
- String substring (uint32_t pos, uint32_t length=Maxu32) const

  return part of the string
- String replace (char before, char after, uint32_t pos=0) const

  replace part of the string
- String **replace** (const char *before, const char *after, uint32_t pos=0) const
- String **replace** (const String &before, const String &after, uint32_t pos=0) const
- const Array< String > split (const char *delimiters, uint32_t length=Maxu32) const

  split string
- const Array< String > **split** (const String &delimiters, uint32_t length=Maxu32) const
- String extension (const char *extension) const

  file name utils
- String **extension** () const
- String **pathname** () const
- String **basename** () const
- String **dirname** () const
- String capitalize (const char *delimiters=nullptr, const char *spaces=nullptr) const

  convert string case
- String **lower** () const
- String **upper** () const
- uint32_t **toUtf16** (wchar_t *d, uint32_t length) const
- uint32_t **toUtf32** (uint32_t *d, uint32_t length) const
- uint32_t vscanf (const char *format, va_list args) const

  string scan function
- uint32_t **scanf** (const char *format,...) const 1(2
- uint32_t String & vprintf (const char *format, va_list args)

  string printf function
- String & **printf** (const char *format,...) 1(2
- template<class... List>
  String String & **tprintf** (const char *f, List... Args)
- int32_t **toi32** (uint32_t radix=10, uint32_t pos=0) const
- int64_t **toi64** (uint32_t radix=10, uint32_t pos=0) const
- uint32_t **tou32** (uint32_t radix=10, uint32_t pos=0) const
- uint64_t **tou64** (uint32_t radix=10, uint32_t pos=0) const
- float32_t **tof32** (uint32_t pos=0) const
- float64_t **tof64** (uint32_t pos=0) const
- uint32_t toHashu32 (uint32_t pos=0) const

string to hash value

- uint64_t **toHashu64** (uint32_t pos=0) const
- uint32_t toRGBAu8 (uint32_t pos=0) const

string to color value

- uint64_t **toBytes** (uint32_t pos=0, uint32_t *size=nullptr) const
- uint64_t **toNumber** (uint32_t pos=0, uint32_t *size=nullptr) const
- uint64_t **toFrequency** (uint32_t pos=0, uint32_t *size=nullptr) const
- float64_t **toLength** (uint32_t pos=0, uint32_t *size=nullptr) const

Static Public Member Functions

- static String relname (const char *path, const char *str)

relative file name

- static String **relname** (const String &path, const String &str)
- static uint32_t toUtf32 (const char *str, uint32_t &code)

string to unicode

- static uint32_t fromUtf32 (String &d, uint32_t code)

unicode to string

- static String **fromUtf16** (const wchar_t *str, uint32_t length=Maxu32)
- static String **fromUtf32** (const uint32_t *str, uint32_t length=Maxu32)
- static String fromUrl (const char *str, uint32_t length=Maxu32)

url to string

- static String **fromUrl** (const String &string, uint32_t length=Maxu32)
- static String vformat (const char *format, va_list args)

string format function

- static String **format** (const char *format,...) 1(1
- template<class... List>
  static String static String **tformat** (const char *format, List... Args)
- static String & fromi32 (String &d, int32_t value, uint32_t radix=10)

value to string

- static String & **fromi64** (String &d, int64_t value, uint32_t radix=10)
- static String & **fromu32** (String &d, uint32_t value, uint32_t radix=10)
- static String & **fromu64** (String &d, uint64_t value, uint32_t radix=10)
- static String & **fromf32** (String &d, float32_t value, uint32_t digits=6, bool compact=false, bool exponent=false)
- static String & **fromf64** (String &d, float64_t value, uint32_t digits=12, bool compact=false, bool exponent=false)
- static String **fromi32** (int32_t value, uint32_t radix=10)
- static String **fromi64** (int64_t value, uint32_t radix=10)
- static String **fromu32** (uint32_t value, uint32_t radix=10)
- static String **fromu64** (uint64_t value, uint32_t radix=10)
- static String **fromf32** (float32_t value, uint32_t digits=6, bool compact=false, bool exponent=false)
- static String **fromf64** (float64_t value, uint32_t digits=12, bool compact=false, bool exponent=false)
- static int32_t toi32 (const char *str, uint32_t radix=10, uint32_t *size=nullptr)

string to value

- static int64_t **toi64** (const char *str, uint32_t radix=10, uint32_t *size=nullptr)
- static uint32_t **tou32** (const char *str, uint32_t radix=10, uint32_t *size=nullptr)
- static uint64_t **tou64** (const char *str, uint32_t radix=10, uint32_t *size=nullptr)
- static int32_t **toi32** (const char *str, uint32_t *size)
- static int64_t **toi64** (const char *str, uint32_t *size)
- static uint32_t **tou32** (const char *str, uint32_t *size)
- static uint64_t **tou64** (const char *str, uint32_t *size)
- static float32_t **tof32** (const char *str, uint32_t *size=nullptr)

- static float64_t **tof64** (const char *str, uint32_t *size=nullptr)
- static String fromTime (uint64_t usec, uint32_t digits=2)

  value convertors
- static String **fromBytes** (uint64_t bytes, uint32_t digits=2)
- static String **fromNumber** (uint64_t value, uint32_t digits=2)
- static String **fromFrequency** (uint64_t hz, uint32_t digits=2)
- static String **fromLength** (float64_t distance, uint32_t digits=2)
- static String **fromAngle** (float64_t angle, uint32_t digits=2)
- static uint64_t **toBytes** (const char *str, uint32_t *size=nullptr)
- static uint64_t **toNumber** (const char *str, uint32_t *size=nullptr)
- static uint64_t **toFrequency** (const char *str, uint32_t *size=nullptr)
- static float64_t **toLength** (const char *str, uint32_t *size=nullptr)

Static Public Attributes

- static const String null

  empty string

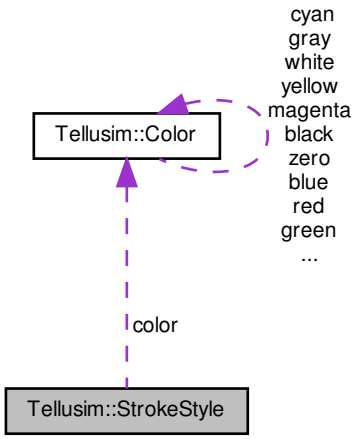
5.250.1 Detailed Description

The String class provides a comprehensive set of functionalities for managing and manipulating strings. It supports operations such as resizing, clearing, copying, appending, inserting, removing, and reversing strings. The class also provides utility functions for string comparison, search, format conversion, and encoding/decoding. Additionally, it includes functions for working with file paths, string case manipulation, and string formatting using variadic templates. The String class can handle different types of string data, including ASCII, UTF-16, and UTF-32 encoded data. It also provides static functions to convert between strings and various data types (e.g., integers, floating-point values, and sizes).

5.251 Tellusim::StrokeStyle Struct Reference

```
#include <interface/TellusimTypes.h>
```

Collaboration diagram for Tellusim::StrokeStyle:



Public Member Functions

- **StrokeStyle** (float32_t width)
- **StrokeStyle** (const Color &color)
- **StrokeStyle** (float32_t width, float32_t offset)
- **StrokeStyle** (float32_t width, const Color &color)
- **StrokeStyle** (float32_t width, float32_t offset, const Color &color)
- bool isValid () const

  check style
- **operator bool** () const

Public Attributes

- float32_t **width** = 0.0f
- float32_t **offset** = 0.0f
- Color **color** = Color::white

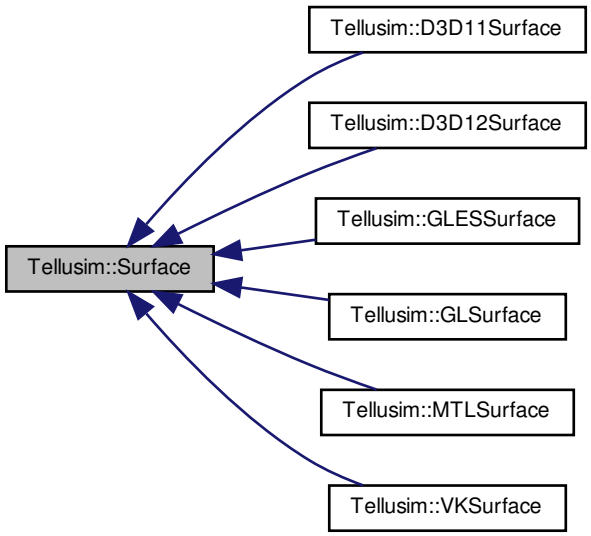
5.251.1 Detailed Description

The StrokeStyle struct represents the properties of a stroke used in rendering, such as the width, offset, and color.

5.252 Tellusim::Surface Class Reference

`#include <platform/TellusimSurface.h>`

Inheritance diagram for Tellusim::Surface:



Public Member Functions

- **Surface** (Context &context)
- **Surface** (Platform platform)
- Platform getPlatform () const

  context platform
- const char * **getPlatformName** () const
- uint32_t getIndex () const

  surface device index
- void setSize (uint32_t width, uint32_t height)

  surface size
- uint32_t **getWidth** () const
- uint32_t **getHeight** () const
- void setMultisample (uint32_t multisample)

  surface multisample
- uint32_t **getMultisample** () const
- bool **hasMultisample** () const
- void setColorLayer (uint32_t layer, uint32_t layers)

  surface layers
- void **setDepthLayer** (uint32_t layer, uint32_t layers)
- uint32_t **getColorLayer** () const
- uint32_t **getDepthLayer** () const
- uint32_t **getColorLayers** () const
- uint32_t **getDepthLayers** () const
- bool **hasColorLayers** () const
- bool **hasDepthLayers** () const
- void setColorFormat (Format format)

  surface formats
- void **setDepthFormat** (Format format)
- Format **getColorFormat** () const
- Format **getDepthFormat** () const

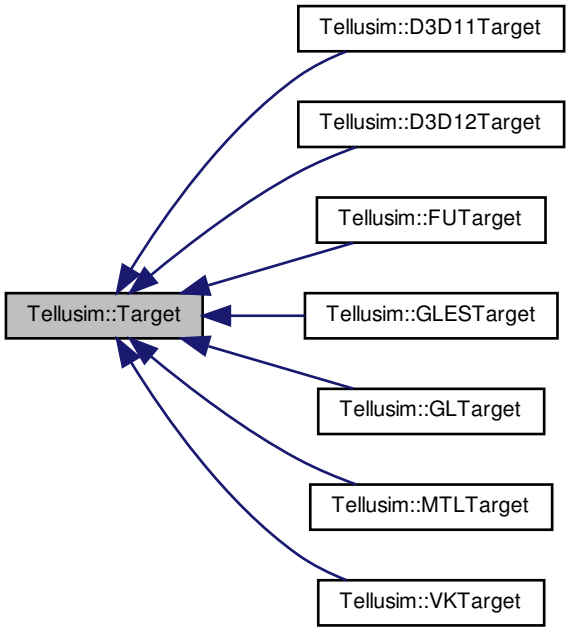
5.252.1 Detailed Description

The Surface class represents a rendering surface, encapsulating the properties and configurations related to a graphical context and its associated platform. It provides methods for querying and manipulating surface characteristics, such as the platform, size, multisampling options, layers, and formats. The class supports configuring the surface color and depth layers, as well as its associated formats, allowing for fine control over how rendering operations are performed on that surface. It also supports features like multisampling and layer management for advanced rendering scenarios.

5.253 Tellusim::Target Class Reference

```
#include <platform/TellusimTarget.h>
```

Inheritance diagram for Tellusim::Target:



Public Types

- enum Operation {
  BeginLoad = (1 << 0),
  BeginClear = (1 << 1),
  BeginDiscard = (1 << 2),
  BeginMask = (BeginLoad | BeginClear | BeginDiscard),
  EndStore = (1 << 3),
  EndResolve = (1 << 4),
  EndDiscard = (1 << 5),
  EndMask = (EndStore | EndResolve | EndDiscard),
  OpNone = 0,
  OpLoad = BeginLoad,
  OpLoadStore = (BeginLoad | EndStore),
  OpClearStore = (BeginClear | EndStore),
  OpClearDiscard = (BeginClear | EndDiscard),
  OpDefault = OpClearStore }

  Target operations.

Public Member Functions

- Platform getPlatform () const

  target platform
- const char * **getPlatformName** () const
- uint32_t getIndex () const

target device index

- bool begin (Fence &fence)

begin target

- bool **begin** ()
- void end (Fence &fence)

end target

- void **end** ()
- void swap (Surface &surface)

swap target

- bool isEnabled () const

check target

- Format getColorFormat (uint32_t index=0) const

target format

- Format **getDepthFormat** () const
- uint32_t **getMultisample** () const
- bool **hasMultisample** () const
- bool **isFlipped** () const
- bool **isAtomic** () const
- uint32_t getWidth () const

target dimension

- uint32_t **getHeight** () const
- uint32_t **getDepth** () const
- uint32_t **getFaces** () const
- uint32_t **getLayers** () const
- uint32_t **getMipmaps** () const
- void setClearColor (const Color &color)

color target

- void **setClearColor** (uint32_t index, const Color &color)
- void **setClearColor** (float32_t r, float32_t g, float32_t b, float32_t a)
- void **setClearColor** (uint32_t index, float32_t r, float32_t g, float32_t b, float32_t a)
- void **setColorTexture** (Texture &texture, Operation op=OpDefault, const Slice &slice=Slice())
- void **setColorTexture** (uint32_t index, Texture &texture, Operation op=OpDefault, const Slice &slice=Slice())
- void **setColorResolve** (Texture &texture, const Slice &slice=Slice())
- void **setColorResolve** (uint32_t index, Texture &texture, const Slice &slice=Slice())
- uint32_t **getNumTargets** () const
- const Color & **getClearColor** (uint32_t index=0) const
- Operation **getColorOp** (uint32_t index=0) const
- Texture **getColorTexture** (uint32_t index) const
- Texture **getColorResolve** (uint32_t index) const
- const Slice & **getColorTextureSlice** (uint32_t index) const
- const Slice & **getColorResolveSlice** (uint32_t index) const
- void setClearDepth (float32_t depth, uint32_t stencil=0)

depth target

- void **setDepthTexture** (Texture &texture, Operation op=OpDefault, const Slice &slice=Slice())
- void **setDepthResolve** (Texture &texture, const Slice &slice=Slice())
- float32_t **getClearDepth** () const
- uint32_t **getClearStencil** () const
- Operation **getDepthOp** () const
- Texture **getDepthTexture** () const
- Texture **getDepthResolve** () const
- const Slice & **getDepthTextureSlice** () const
- const Slice & **getDepthResolveSlice** () const

5.253.1 Detailed Description

The Target class provides functionality for managing rendering targets, including color and depth buffers, in a graphics pipeline. It supports operations such as loading, clearing, discarding, storing, and resolving targets, allowing for flexible rendering workflows. The class enables managing target properties such as format, dimension, multisampling, as well as configuring clear colors, textures, and slices for color and depth targets, making it suitable for complex rendering tasks.

5.254 Tellusim::Async::Task Class Reference

Task.

`#include <core/TellusimAsync.h>`

Public Member Functions

- bool operator== (const Task &task) const
  comparison operators
- bool **operator!=** (const Task &task) const
- bool empty () const
  task info
- **operator bool** () const
- void clear ()
  clear functions queue
- void cancel ()
  cancel functions queue
- uint32_t index ()
  queue thread index
- uint32_t size () const
  number of queued functions
- template<class Func >
  Task & run (const Func &func)
  run the function on the queue
- template<class Func , class A0 >
  Task & **run** (const Func &func, A0 a0)
- template<class Func , class A0 , class A1 >
  Task & **run** (const Func &func, A0 a0, A1 a1)
- template<class Func , class A0 , class A1 , class A2 >
  Task & **run** (const Func &func, A0 a0, A1 a1, A2 a2)
- template<class Func , class A0 , class A1 , class A2 , class A3 >
  Task & **run** (const Func &func, A0 a0, A1 a1, A2 a2, A3 a3)
- template<class Func , class A0 , class A1 , class A2 , class A3 , class A4 >
  Task & **run** (const Func &func, A0 a0, A1 a1, A2 a2, A3 a3, A4 a4)
- template<class Func , class A0 , class A1 , class A2 , class A3 , class A4 , class A5 >
  Task & **run** (const Func &func, A0 a0, A1 a1, A2 a2, A3 a3, A4 a4, A5 a5)
- template<class Func , class A0 , class A1 , class A2 , class A3 , class A4 , class A5 , class A6 >
  Task & **run** (const Func &func, A0 a0, A1 a1, A2 a2, A3 a3, A4 a4, A5 a5, A6 a6)
- template<class Func , class A0 , class A1 , class A2 , class A3 , class A4 , class A5 , class A6 , class A7 >
  Task & **run** (const Func &func, A0 a0, A1 a1, A2 a2, A3 a3, A4 a4, A5 a5, A6 a6, A7 a7)
- template<class Func >
  Task & **run** (const Function< Func > &func)
- bool check (uint32_t num=Maxu32) const

check task completion status

- bool wait (uint32_t num=Maxu32) const

  waiting for the task completion

- template<class Ret >
  const Ret & get (uint32_t num=0) const

  function result

- template<class Ret >
  Ret **getPtr** (uint32_t num=0) const
- bool **getBool** (uint32_t num=0) const

Friends

- class **Async**

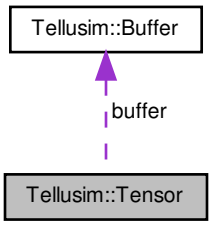
5.254.1 Detailed Description

Task.

5.255 Tellusim::Tensor Struct Reference

`#include <parallel/TellusimTensorGraph.h>`

Collaboration diagram for Tellusim::Tensor:



Public Member Functions

- **Tensor** (Buffer *buffer, size_t offset=0)
- **Tensor** (Format format, uint32_t width=0, uint32_t height=1, uint32_t depth=1, uint32_t layers=1)
- **Tensor** (Buffer *buffer, Format format, uint32_t width=0, uint32_t height=1, uint32_t depth=1, uint32_t layers=1)
- **Tensor** (const Tensor &t)
- **Tensor** (const Tensor &t, uint32_t width, uint32_t height=1, uint32_t depth=1, uint32_t layers=1)
- bool isValid () const

  check tensor

- **operator bool** () const

- uint32_t getSize () const

  tensor size
- size_t **getBytes** () const
- Tensor setAxis (uint32_t axis) const

  set operation parameters
- Tensor **setKernel** (uint32_t kernel) const
- Tensor **setStride** (uint32_t stride) const
- Tensor **setPadding** (uint32_t padding) const
- Tensor **setDilation** (uint32_t dilation) const
- Tensor setScaleBias (float32_t scale, float32_t bias) const

  set value parameters
- Tensor **setScale** (float32_t scale) const
- Tensor **setBias** (float32_t bias) const

Public Attributes

- Buffer * **buffer** = nullptr
- Format **format** = FormatUnknown
- size_t **offset** = 0
- uint32_t **axis** = 0
- uint32_t **kernel** = 2
- uint32_t **stride** = 1
- uint32_t **padding** = 0
- uint32_t **dilation** = 1
- float32_t **scale** = 1.0f
- float32_t **bias** = 0.0f
-

```
union {
  struct {
    uint32_t width
    uint32_t height
    uint32_t depth
    uint32_t layers
  }
  uint32_t size [4]
};
```

5.255.1 Detailed Description

The Tensor structure provides a comprehensive representation of a multi-dimensional array used in computational frameworks. It allows for flexible creation, parameter configuration (e.g., for convolution or pooling operations), and manipulation of tensors. By including additional features like scaling, biasing, and various operation parameters, it becomes a powerful tool for numerical computation and machine learning tasks.

5.256 Tellusim::TensorGraph Class Reference

`#include <parallel/TellusimTensorGraph.h>`

Public Types

- enum Operation {
  Clear = 0,
  Range,
  Copy,
  Cat,
  Transpose,
  MatMul,
  Mul,
  Mad,
  Div,
  Add,
  Conv,
  DeConv,
  BatchNorm,
  BatchMad,
  SoftMin,
  SoftMax,
  MaxPool,
  AvgPool,
  GELU,
  ReLU,
  SiLU,
  Sigm,
  Tanh,
  Sin,
  Cos,
  Exp,
  NumOperations }

  Graph operations.
- enum Flags {
  FlagNone = 0,
  FlagSizeQuery = (1 << 0),
  FlagFormatRf32 = (1 << 1),
  FlagFormatRf16 = (1 << 2),
  FlagTranspose = (1 << 3),
  FlagWrapClamp = (1 << 4),
  FlagWrapRepeat = (1 << 5),
  FlagReadScale = (1 << 6),
  FlagReadBias = (1 << 7),
  FlagConvert = (1 << 8),
  FlagKernel = (1 << 9),
  FlagGELU = (1 << 10),
  FlagReLU = (1 << 11),
  FlagSiLU = (1 << 12),
  FlagSigm = (1 << 13),
  FlagTanh = (1 << 14),
  FlagSin = (1 << 15),
  FlagCos = (1 << 16),
  FlagExp = (1 << 17),
  FlagFormat = FlagFormatRf32 | FlagFormatRf16,
  FlagWrap = FlagWrapClamp | FlagWrapRepeat,
  FlagRead = FlagReadScale | FlagReadBias,
  FlagUnit = FlagGELU | FlagReLU | FlagSiLU,
  FlagMath = FlagSigm | FlagTanh | FlagSin | FlagCos | FlagExp,
  FlagsAll = FlagFormat | FlagTranspose | FlagWrap | FlagRead | FlagConvert | FlagKernel | FlagUnit | Flag↩
  Math }

Graph flags.

- enum Masks {
  MaskNone = 0,
  MaskClear = (1 << Clear),
  MaskRange = (1 << Range),
  MaskCopy = (1 << Copy),
  MaskCat = (1 << Cat),
  MaskTranspose = (1 << Transpose),
  MaskMatMul = (1 << MatMul),
  MaskMul = (1 << Mul),
  MaskMad = (1 << Mad),
  MaskDiv = (1 << Div),
  MaskAdd = (1 << Add),
  MaskConv = (1 << Conv),
  MaskDeConv = (1 << DeConv),
  MaskBatchNorm = (1 << BatchNorm),
  MaskBatchMad = (1 << BatchMad),
  MaskSoftMin = (1 << SoftMin),
  MaskSoftMax = (1 << SoftMax),
  MaskMaxPool = (1 << MaxPool),
  MaskAvgPool = (1 << AvgPool),
  MaskGELU = (1 << GELU),
  MaskReLU = (1 << ReLU),
  MaskSiLU = (1 << SiLU),
  MaskSigm = (1 << Sigm),
  MaskTanh = (1 << Tanh),
  MaskSin = (1 << Sin),
  MaskCos = (1 << Cos),
  MaskExp = (1 << Exp),
  MasksAll = (1 << NumOperations) - 1 }

  Graph masks.

Public Member Functions

- void clear ()

  clear graph
- bool isCreated () const

  check graph
- bool create (const Device &device, Flags flags=FlagsAll, Masks masks=MasksAll, Async *async=nullptr)

  create graph
- bool dispatch (Compute &compute, Operation op, const Tensor &dest, Flags flags=FlagNone) const
- bool **dispatch** (Compute &compute, Operation op, Tensor &dest, const Tensor &src_0, Flags flags=Flag↩None) const
- bool **dispatch** (Compute &compute, Operation op, Tensor &dest, const Tensor &src_0, const Tensor &src_1, Flags flags=FlagNone) const
- bool **dispatch** (Compute &compute, Operation op, Tensor &dest, const Tensor &src_0, const Tensor &src_1, const Tensor &src_2, Flags flags=FlagNone) const
- bool dispatch (Compute &compute, const Tensor &dest, Texture &src, const Region &region, const Slice &slice=Slice()) const

  dispatch Texture to Tensor
- bool **dispatch** (Compute &compute, const Tensor &dest, Texture &src, const Slice &slice=Slice()) const
- bool dispatch (Compute &compute, Texture &dest, const Tensor &src, const Region &region, const Slice &slice=Slice()) const

  dispatch Tensor to Texture
- bool **dispatch** (Compute &compute, Texture &dest, const Tensor &src, const Slice &slice=Slice()) const

5.256.1 Detailed Description

The TensorGraph class is a powerful abstraction for managing and executing tensor operations within a computational graph. It provides an efficient mechanism for building, modifying, and dispatching operations on tensors with a high degree of flexibility and control over execution parameters, formats, and flags. Whether working with basic operations or advanced deep learning techniques, this class offers a robust interface for numerical computation.

5.256.2 Member Function Documentation

5.256.2.1 dispatch()

```
bool Tellusim::TensorGraph::dispatch (
            Compute & compute,
            Operation op,
            const Tensor & dest,
            Flags flags = FlagNone ) const
```

dispatch Tensor operation

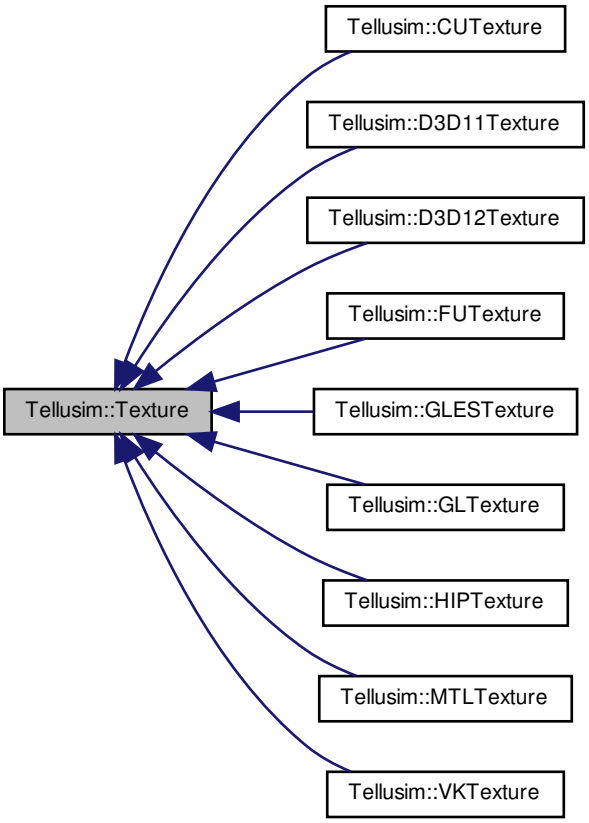
Parameters

op	Graph operation.
flags	Operation flags.
dest	Destination tensor.
src_0	Source tensors.

5.257 Tellusim::Texture Class Reference

```
#include <platform/TellusimTexture.h>
```

Inheritance diagram for Tellusim::Texture:



Public Types

- enum Flags {
  FlagNone = 0,
  FlagRead = (1 << 0),
  FlagWrite = (1 << 1),
  FlagTarget = (1 << 2),
  FlagBuffer = (1 << 3),
  FlagSource = (1 << 4),
  FlagSparse = (1 << 5),
  FlagShared = (1 << 6),
  FlagExtern = (1 << 7),
  FlagInterop = (1 << 8),
  FlagSurface = (1 << 9),
  FlagMutable = (1 << 10),
  FlagMipmaps = (1 << 11),
  FlagGenerate = (1 << 12),
  FlagFormatNorm = (1 << 13),
  FlagFormatSRGB = (1 << 14),
  FlagFormatSigned = (1 << 15),
  FlagMultisample2 = (1 << 16),

FlagMultisample4 = (1 << 17),
FlagMultisample8 = (1 << 18),
FlagClearOne = (1 << 19),
FlagClearZero = (1 << 20),
FlagClearNormal = (1 << 21),
FlagMultisample = (FlagMultisample2 | FlagMultisample4 | FlagMultisample8),
DefaultFlags = FlagNone,
NumFlags = 22 }

Texture flags.

Public Member Functions

- Platform getPlatform () const
  texture platform
- const char * **getPlatformName** () const
- uint32_t getIndex () const
  texture device index
- void clear ()
  clear texture
- bool isCreated () const
  check texture
- void setName (const char *name)
  texture name
- String **getName** () const
- bool create (Type type, Format format, const Size &size, uint32_t layers, Flags flags=DefaultFlags)
  create texture
- bool **create2D** (Format format, uint32_t size, Flags flags=DefaultFlags)
- bool **create3D** (Format format, uint32_t size, Flags flags=DefaultFlags)
- bool **createCube** (Format format, uint32_t size, Flags flags=DefaultFlags)
- bool **create2D** (Format format, uint32_t width, uint32_t height, Flags flags=DefaultFlags)
- bool **create3D** (Format format, uint32_t width, uint32_t height, uint32_t depth, Flags flags=DefaultFlags)
- bool **create2D** (Format format, uint32_t width, uint32_t height, uint32_t layers, Flags flags=DefaultFlags)
- bool **createCube** (Format format, uint32_t size, uint32_t layers, Flags flags=DefaultFlags)
- Type getType () const
  texture type
- const char * **getTypeName** () const
- bool **is2DType** () const
- bool **is3DType** () const
- bool **isCubeType** () const
- Format getFormat () const
  texture format
- const char * **getFormatName** () const
- bool **isColorFormat** () const
- bool **isDepthFormat** () const
- bool **isPixelFormat** () const
- bool **isPlainFormat** () const
- bool **isMixedFormat** () const
- bool **isBlockFormat** () const
- bool **isStencilFormat** () const
- bool **isNormFormat** () const
- bool **isSRGBFormat** () const
- bool **isFloatFormat** () const
- bool **isSignedFormat** () const

- bool **isUnsignedFormat** () const
- bool **isIntegerFormat** () const
- bool **isi8Format** () const
- bool **isu8Format** () const
- bool **is8BitFormat** () const
- bool **isi16Format** () const
- bool **isu16Format** () const
- bool **isf16Format** () const
- bool **is16BitFormat** () const
- bool **isi32Format** () const
- bool **isu32Format** () const
- bool **isf32Format** () const
- bool **is32BitFormat** () const
- bool **isi64Format** () const
- bool **isu64Format** () const
- bool **isf64Format** () const
- bool **is64BitFormat** () const
- bool **isBC15Format** () const
- bool **isBC67Format** () const
- bool **isETC2Format** () const
- bool **isASTCFormat** () const
- uint32_t **getComponents** () const
- uint32_t **getPixelSize** () const
- uint32_t **getBlockSize** () const
- uint32_t **getBlockWidth** () const
- uint32_t **getBlockHeight** () const
- Flags getFlags () const

  texture flags
- bool **hasFlag** (Flags flags) const
- bool **hasFlags** (Flags flags) const
- String **getFlagsName** () const
- uint32_t getMultisample () const

  texture multisample
- bool **hasMultisample** () const
- uint32_t getWidth () const

  texture dimension
- uint32_t **getHeight** () const
- uint32_t **getDepth** () const
- uint32_t **getFaces** () const
- uint32_t **getLayers** () const
- uint32_t **getMipmaps** () const
- uint32_t **findMipmap** (const Size &size) const
- uint32_t **getWidth** (uint32_t mipmap) const
- uint32_t **getHeight** (uint32_t mipmap) const
- uint32_t **getDepth** (uint32_t mipmap) const
- bool **hasFaces** () const
- bool **hasLayers** () const
- bool **hasMipmaps** () const
- Size **getSize** () const
- Region **getRegion** () const
- Slice **getSlice** () const
- Size **getSize** (uint32_t mipmap) const
- Region **getRegion** (uint32_t mipmap) const
- Slice **getSlice** (uint32_t mipmap) const

- uint32_t getTileWidth () const
  sparse texture dimension
- uint32_t **getTileHeight** () const
- uint32_t **getTileDepth** () const
- uint32_t **getTileMipmaps** () const
- Size **getTileSize** () const
- String getDescription () const
  texture description
- size_t getMemory () const
  memory usage

Static Public Member Functions

- static const char * **getTypeName** (Type type)

5.257.1 Detailed Description

The Texture class represents a GPU texture resource supporting 2D, 3D, and Cube types with configurable format, dimensions, layers, mipmaps, and multisampling. It includes flags for usage patterns such as read/write access, target rendering, sharing, and interop with external or sparse resources. The class provides functions to query texture properties, format characteristics, and memory usage for efficient texture management across rendering platforms.

5.258 Tellusim::TextureTable Class Reference

```
#include <platform/TellusimTexture.h>
```

Public Member Functions

- Platform getPlatform () const
  table platform
- const char * **getPlatformName** () const
- uint32_t getIndex () const
  table device index
- void clear ()
  clear table
- bool isCreated () const
  check table
- void setName (const char *name)
  table name
- String **getName** () const
- bool create (Texture::Type type, uint32_t size)
  create table
- Texture::Type getType () const
  table type
- const char * **getTypeName** () const
- uint32_t getSize () const
  table textures
- Texture **get** (uint32_t index) const
- bool **isOwner** (uint32_t index) const
- size_t getMemory () const
  memory usage

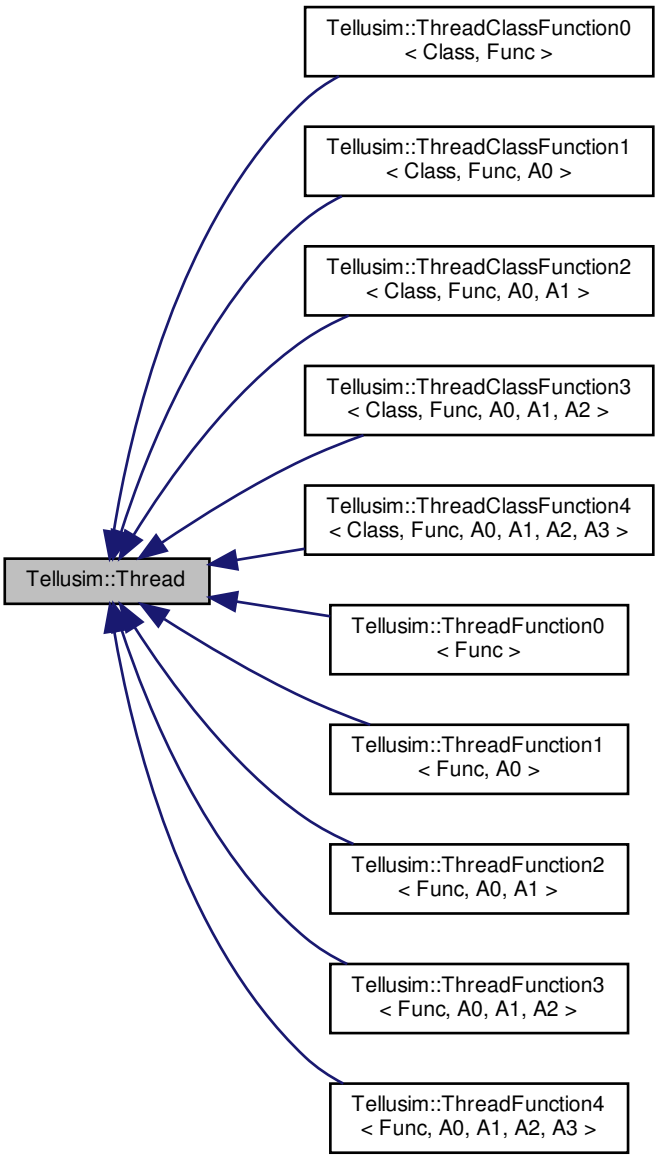
5.258.1 Detailed Description

The TextureTable class provides a container for managing multiple texture objects with support for bindless resource access. It enables efficient rendering workflows by organizing textures of a specific type, reducing the overhead of traditional binding operations. This class supports querying platform details, memory usage, and ownership of individual texture entries.

5.259 Tellusim::Thread Class Reference

`#include <core/TellusimThread.h>`

Inheritance diagram for Tellusim::Thread:



Public Member Functions

- bool run (uint32_t stack=1024 *1024)
  run the thread
- bool stop (bool wait=false)
  stop the thread
- bool wait ()
  wait for the signal
- bool signal ()
  wake up the thread
- bool terminate ()
  terminate the thread
- bool isRunning () const
  thread status
- bool **isStopped** () const
- bool **isWaiting** () const

Protected Member Functions

- virtual void process ()
  thread process

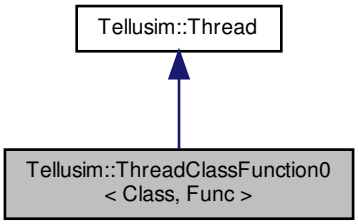
5.259.1 Detailed Description

Hardware thread

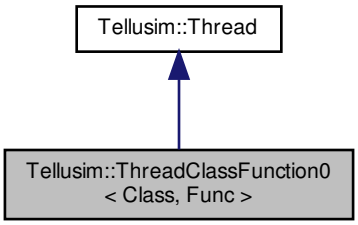
5.260 Tellusim::ThreadClassFunction0< Class, Func > Class Template Reference

```
#include <core/TellusimThread.h>
```

Inheritance diagram for Tellusim::ThreadClassFunction0< Class, Func >:



Collaboration diagram for Tellusim::ThreadClassFunction0< Class, Func >:



Public Member Functions

- **ThreadClassFunction0** (Class *c, const Func &func)

Protected Member Functions

- virtual void process ()

  thread process

Protected Attributes

- Class * **c**
- Func **func**

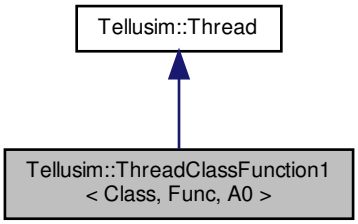
5.260.1 Detailed Description

template<class Class, class Func>
class Tellusim::ThreadClassFunction0< Class, Func >

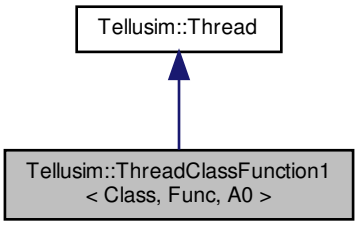
Thread class member function

5.261 Tellusim::ThreadClassFunction1< Class, Func, A0 > Class Template Reference

Inheritance diagram for Tellusim::ThreadClassFunction1< Class, Func, A0 >:



Collaboration diagram for Tellusim::ThreadClassFunction1< Class, Func, A0 >:



Public Member Functions

- **ThreadClassFunction1** (Class *c, const Func &func, A0 a0)

Protected Member Functions

- virtual void process ()

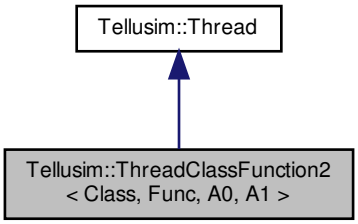
  thread process

Protected Attributes

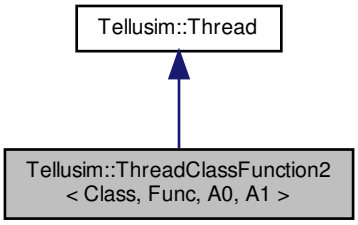
- Class * **c**
- Func **func**
- A0 **a0**

5.262 Tellusim::ThreadClassFunction2< Class, Func, A0, A1 > Class Template Reference

Inheritance diagram for Tellusim::ThreadClassFunction2< Class, Func, A0, A1 >:



Collaboration diagram for Tellusim::ThreadClassFunction2< Class, Func, A0, A1 >:



Public Member Functions

- **ThreadClassFunction2** (Class *c, const Func &func, A0 a0, A1 a1)

Protected Member Functions

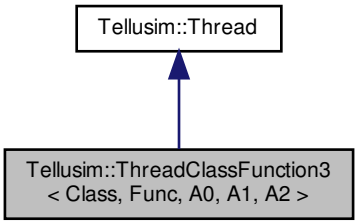
- virtual void process ()
  thread process

Protected Attributes

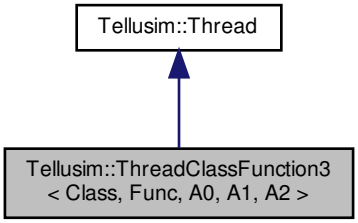
- Class * **c**
- Func **func**
- A0 **a0**
- A1 **a1**

5.263 Tellusim::ThreadClassFunction3< Class, Func, A0, A1, A2 > Class Template Reference

Inheritance diagram for Tellusim::ThreadClassFunction3< Class, Func, A0, A1, A2 >:



Collaboration diagram for Tellusim::ThreadClassFunction3< Class, Func, A0, A1, A2 >:



Public Member Functions

- **ThreadClassFunction3** (Class *c, const Func &func, A0 a0, A1 a1, A2 a2)

Protected Member Functions

- virtual void process ()

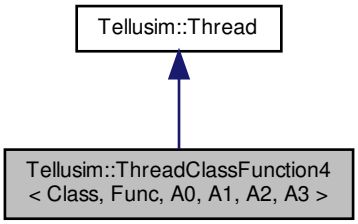
  thread process

Protected Attributes

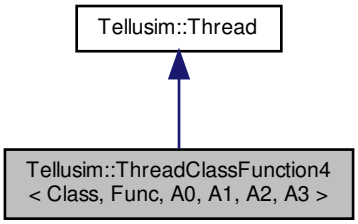
- Class * **c**
- Func **func**
- A0 **a0**
- A1 **a1**
- A2 **a2**

5.264 Tellusim::ThreadClassFunction4< Class, Func, A0, A1, A2, A3 > Class Template Reference

Inheritance diagram for Tellusim::ThreadClassFunction4< Class, Func, A0, A1, A2, A3 >:



Collaboration diagram for Tellusim::ThreadClassFunction4< Class, Func, A0, A1, A2, A3 >:



Public Member Functions

- **ThreadClassFunction4** (Class *c, const Func &func, A0 a0, A1 a1, A2 a2, A3 a3)

Protected Member Functions

- virtual void process ()

  thread process

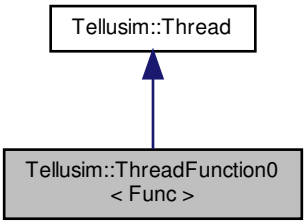
Protected Attributes

- Class * **c**
- Func **func**
- A0 **a0**
- A1 **a1**
- A2 **a2**
- A3 **a3**

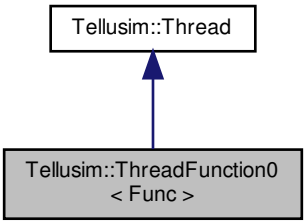
5.265 Tellusim::ThreadFunction0< Func > Class Template Reference

`#include <core/TellusimThread.h>`

Inheritance diagram for Tellusim::ThreadFunction0< Func >:



Collaboration diagram for Tellusim::ThreadFunction0< Func >:



Public Member Functions

- **ThreadFunction0** (const Func &func)

Protected Member Functions

- virtual void process ()
  thread process

Protected Attributes

- Func **func**

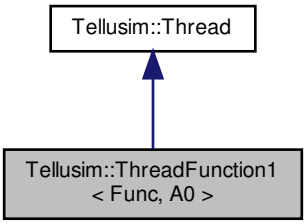
5.265.1 Detailed Description

template<class Func>
class Tellusim::ThreadFunction0< Func >

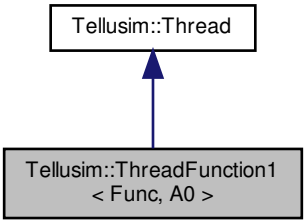
Thread function

5.266 Tellusim::ThreadFunction1< Func, A0 > Class Template Reference

Inheritance diagram for Tellusim::ThreadFunction1< Func, A0 >:



Collaboration diagram for Tellusim::ThreadFunction1< Func, A0 >:



Public Member Functions

- **ThreadFunction1** (const Func &func, A0 a0)

Protected Member Functions

- virtual void process ()

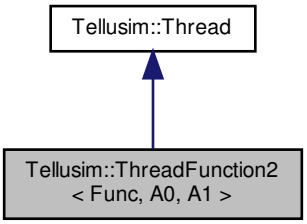
  thread process

Protected Attributes

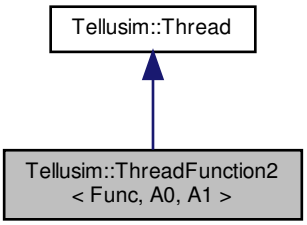
- Func **func**
- A0 **a0**

5.267 Tellusim::ThreadFunction2< Func, A0, A1 > Class Template Reference

Inheritance diagram for Tellusim::ThreadFunction2< Func, A0, A1 >:



Collaboration diagram for Tellusim::ThreadFunction2< Func, A0, A1 >:



Public Member Functions

- **ThreadFunction2** (const Func &func, A0 a0, A1 a1)

Protected Member Functions

- virtual void process ()

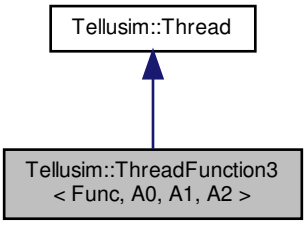
  thread process

Protected Attributes

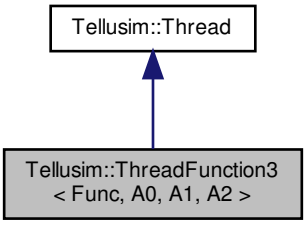
- Func **func**
- A0 **a0**
- A1 **a1**

5.268 Tellusim::ThreadFunction3< Func, A0, A1, A2 > Class Template Reference

Inheritance diagram for Tellusim::ThreadFunction3< Func, A0, A1, A2 >:



Collaboration diagram for Tellusim::ThreadFunction3< Func, A0, A1, A2 >:



Public Member Functions

- **ThreadFunction3** (const Func &func, A0 a0, A1 a1, A2 a2)

Protected Member Functions

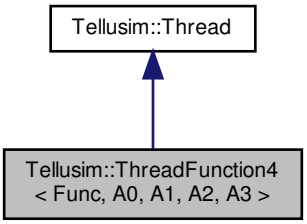
- virtual void process ()
  thread process

Protected Attributes

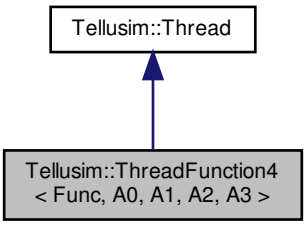
- Func **func**
- A0 **a0**
- A1 **a1**
- A2 **a2**

5.269 Tellusim::ThreadFunction4< Func, A0, A1, A2, A3 > Class Template Reference

Inheritance diagram for Tellusim::ThreadFunction4< Func, A0, A1, A2, A3 >:



Collaboration diagram for Tellusim::ThreadFunction4< Func, A0, A1, A2, A3 >:



Public Member Functions

- **ThreadFunction4** (const Func &func, A0 a0, A1 a1, A2 a2, A3 a3)

Protected Member Functions

- virtual void process ()

  thread process

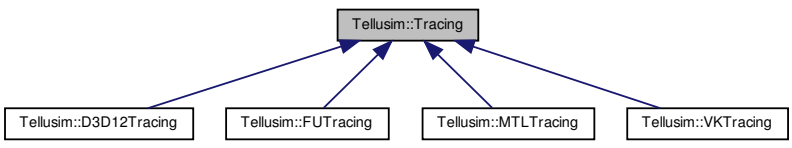
Protected Attributes

- Func **func**
- A0 **a0**
- A1 **a1**
- A2 **a2**
- A3 **a3**

5.270 Tellusim::Tracing Class Reference

`#include <platform/TellusimTracing.h>`

Inheritance diagram for Tellusim::Tracing:



Classes

- struct BuildIndirect
  build indirect parameters
- struct Instance
  tracing instance

Public Types

- enum Flags {
  FlagNone = 0,
  FlagInfo = (1 << 0),
  FlagUpdate = (1 << 1),
  FlagCompact = (1 << 2),
  FlagTransparent = (1 << 3),
  FlagFastBuild = (1 << 4),
  FlagFastTrace = (1 << 5),
  DefaultFlags = FlagNone,
  NumFlags = 6 }
  Tracing flags.
- enum { **InstanceSize** = 64 }
  instance size

Public Member Functions

- Platform getPlatform () const
  tracing platform
- const char * **getPlatformName** () const
- uint32_t getIndex () const
  tracing device index
- void clear ()
  clear tracing
- bool isCreated () const
  check tracing
- bool **isBuilt** () const
- void setName (const char *name)
  tracing name
- String **getName** () const
- bool create (Type type, Flags flags=DefaultFlags)
  create tracing
- Type getType () const
  tracing type
- const char * **getTypeName** () const
- bool **isInstanceType** () const
- bool **isTriangleType** () const
- bool **isBoundType** () const
- bool **isGeometryType** () const
- void setParameters (const Tracing &tracing)
  tracing parameters
- uint32_t getNumGeometries () const
  tracing geometries
- void setInstanceBuffer (uint32_t num_instances, Buffer &buffer, size_t offset=0)

instance buffer

- void **setInstanceBuffer** (Buffer &buffer, size_t offset=0)
- void **setNumInstances** (uint32_t num_instances)
- uint32_t **getNumInstances** () const
- Buffer **getInstanceBuffer** () const
- size_t **getInstanceOffset** () const
- void setIndirectBuffer (Buffer &buffer, size_t offset=0)

  indirect buffer
- Buffer **getIndirectBuffer** () const
- size_t **getIndirectOffset** () const
- uint32_t addVertexBuffer (uint32_t num_vertices, Format format, size_t stride, Buffer buffer=Buffer::null, size_t offset=0)

  vertex buffers
- void **setVertexBuffer** (uint32_t index, uint32_t num_vertices, Buffer &buffer, size_t offset=0)
- void **setVertexBuffer** (uint32_t index, Buffer &buffer, size_t offset=0)
- void **setNumVertices** (uint32_t index, uint32_t num_vertices)
- uint32_t **getNumVertices** (uint32_t index) const
- Format **getVertexFormat** (uint32_t index) const
- uint32_t **getVertexStride** (uint32_t index) const
- Buffer **getVertexBuffer** (uint32_t index) const
- size_t **getVertexOffset** (uint32_t index) const
- uint32_t addIndexBuffer (uint32_t num_indices, Format format, Buffer buffer=Buffer::null, size_t offset=0)

  index buffers
- void **setIndexBuffer** (uint32_t index, uint32_t num_indices, Buffer &buffer, size_t offset=0)
- void **setIndexBuffer** (uint32_t index, Buffer &buffer, size_t offset=0)
- void **setNumIndices** (uint32_t index, uint32_t num_indices)
- uint32_t **getNumIndices** (uint32_t index) const
- Format **getIndexFormat** (uint32_t index) const
- Buffer **getIndexBuffer** (uint32_t index) const
- size_t **getIndexOffset** (uint32_t index) const
- uint32_t addBoundBuffer (uint32_t num_bounds, size_t stride, Buffer buffer=Buffer::null, size_t offset=0)

  bound buffers
- void **setBoundBuffer** (uint32_t index, uint32_t num_bounds, Buffer &buffer, size_t offset=0)
- void **setBoundBuffer** (uint32_t index, Buffer &buffer, size_t offset=0)
- void **setNumBounds** (uint32_t index, uint32_t num_bounds)
- uint32_t **getNumBounds** (uint32_t index) const
- uint32_t **getBoundStride** (uint32_t index) const
- Buffer **getBoundBuffer** (uint32_t index) const
- size_t **getBoundOffset** (uint32_t index) const
- String getDescription () const

  tracing description
- uint64_t getTracingAddress () const

  tracing address
- size_t getBuildSize () const

  scratch buffer size
- size_t **getUpdateSize** () const
- size_t getMemory () const

  memory usage

Static Public Member Functions

- static const char * **getTypeName** (Type type)

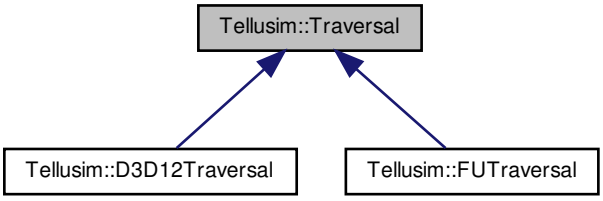
5.270.1 Detailed Description

The Tracing class provides an abstraction for managing acceleration structures used in ray-tracing tasks. It allows the creation, management, and manipulation of different types of acceleration structures, such as instances, triangles, and bounds. It provides methods to set up and manage vertex buffers, index buffers, and instance buffers, enabling the setup of geometries and the specification of tracing parameters.

5.271 Tellusim::Traversal Class Reference

`#include <platform/TellusimTraversal.h>`

Inheritance diagram for Tellusim::Traversal:



Public Member Functions

- Platform getPlatform () const
  traversal platform
- const char * **getPlatformName** () const
- uint32_t getIndex () const
  traversal device index
- void clear ()
  clear traversal
- bool isCreated () const
  check traversal
- void setName (const char *name)
  traversal name
- String **getName** () const
- bool create ()
  create traversal
- void setParameters (const Traversal &traversal)
  traversal parameters
- bool **saveState** (Stream &stream) const
- void addShader (Shader &shader, bool owner=false)
  shader pointers
- Shader **getRayGenShader** () const
- bool loadShader (Shader::Type type, const char *name, const char *format,...) 1(4
  load shaders

- bool bool **loadShaderGLSL** (Shader::Type type, const char *name, const char *format,...) 1(4
- bool bool bool **loadShader** (Shader::Type type, const char *name, const String &macros=String::null, const char **includes=nullptr, uint32_t size=0)
- bool **loadShaderGLSL** (Shader::Type type, const char *name, const String &macros=String::null, const char **includes=nullptr, uint32_t size=0)
- bool **loadShaderSPIRV** (Shader::Type type, const char *name)
- bool createShader (Shader::Type type, const char *src, const char *format,...) 1(4

  create shaders
- bool bool **createShaderGLSL** (Shader::Type type, const char *src, const char *format,...) 1(4
- bool bool bool **createShader** (Shader::Type type, const char *src, const String &macros=String::null, const char **includes=nullptr, uint32_t size=0)
- bool **createShaderGLSL** (Shader::Type type, const char *src, const String &macros=String::null, const char **includes=nullptr, uint32_t size=0)
- bool **createShaderSPIRV** (Shader::Type type, const Array< uint32_t > &data)
- uint32_t addSampler (Shader::Mask mask)

  sampler parameters
- uint32_t **getNumSamplers** () const
- Traversal & **setSamplerOffset** (uint32_t offset)
- uint32_t **getSamplerOffset** () const
- Traversal & **setSamplerMask** (uint32_t index, Shader::Mask mask)
- Shader::Mask **getSamplerMask** (uint32_t index) const
- Traversal & **setSamplerMasks** (uint32_t index, uint32_t num, Shader::Mask mask, bool array=false)
- Shader::Mask **getSamplerMasks** (uint32_t index, uint32_t num) const
- Traversal & **setSamplerArray** (uint32_t index, uint32_t num, bool array)
- uint32_t **getSamplerArray** (uint32_t index) const
- uint32_t addTexture (Shader::Mask mask)

  texture parameters
- uint32_t **getNumTextures** () const
- Traversal & **setTextureOffset** (uint32_t offset)
- uint32_t **getTextureOffset** () const
- Traversal & **setTextureMask** (uint32_t index, Shader::Mask mask)
- Shader::Mask **getTextureMask** (uint32_t index) const
- Traversal & **setTextureMasks** (uint32_t index, uint32_t num, Shader::Mask mask, bool array=false)
- Shader::Mask **getTextureMasks** (uint32_t index, uint32_t num) const
- Traversal & **setTextureArray** (uint32_t index, uint32_t num, bool array)
- uint32_t **getTextureArray** (uint32_t index) const
- uint32_t addSurface (Shader::Mask mask)

  surface parameters
- uint32_t **getNumSurfaces** () const
- Traversal & **setSurfaceOffset** (uint32_t offset)
- uint32_t **getSurfaceOffset** () const
- Traversal & **setSurfaceMask** (uint32_t index, Shader::Mask mask)
- Shader::Mask **getSurfaceMask** (uint32_t index) const
- Traversal & **setSurfaceMasks** (uint32_t index, uint32_t num, Shader::Mask mask, bool array=false)
- Shader::Mask **getSurfaceMasks** (uint32_t index, uint32_t num) const
- Traversal & **setSurfaceArray** (uint32_t index, uint32_t num, bool array)
- uint32_t **getSurfaceArray** (uint32_t index) const
- uint32_t addUniform (Shader::Mask mask, BindFlags flags=BindFlagNone)

  uniform parameters
- uint32_t **getNumUniforms** () const
- Traversal & **setUniformOffset** (uint32_t offset)
- uint32_t **getUniformOffset** () const
- Traversal & **setUniformMask** (uint32_t index, Shader::Mask mask, BindFlags flags=BindFlagNone)
- Shader::Mask **getUniformMask** (uint32_t index) const

- Traversal & **setUniformMasks** (uint32_t index, uint32_t num, Shader::Mask mask, BindFlags flags=BindFlagNone)
- Shader::Mask **getUniformMasks** (uint32_t index, uint32_t num) const
- Traversal & **setUniformFlags** (uint32_t index, BindFlags flags)
- BindFlags **getUniformFlags** (uint32_t index) const
- uint32_t addStorage (Shader::Mask mask, BindFlags flags=BindFlagNone)

  storage parameters
- uint32_t **getNumStorages** () const
- Traversal & **setStorageOffset** (uint32_t offset)
- uint32_t **getStorageOffset** () const
- Traversal & **setStorageMask** (uint32_t index, Shader::Mask mask, BindFlags flags=BindFlagNone)
- Shader::Mask **getStorageMask** (uint32_t index) const
- Traversal & **setStorageMasks** (uint32_t index, uint32_t num, Shader::Mask mask, BindFlags flags=BindFlagNone)
- Shader::Mask **getStorageMasks** (uint32_t index, uint32_t num) const
- Traversal & **setStorageFlags** (uint32_t index, BindFlags flags)
- BindFlags **getStorageFlags** (uint32_t index) const
- uint32_t addTracing (Shader::Mask mask)

  tracing parameters
- uint32_t **getNumTracings** () const
- Traversal & **setTracingOffset** (uint32_t offset)
- uint32_t **getTracingOffset** () const
- Traversal & **setTracingMask** (uint32_t index, Shader::Mask mask)
- Shader::Mask **getTracingMask** (uint32_t index) const
- Traversal & **setTracingMasks** (uint32_t index, uint32_t num, Shader::Mask mask)
- Shader::Mask **getTracingMasks** (uint32_t index, uint32_t num) const
- uint32_t addTexel (Shader::Mask mask)

  texel parameters
- uint32_t **getNumTexels** () const
- Traversal & **setTexelOffset** (uint32_t offset)
- uint32_t **getTexelOffset** () const
- Traversal & **setTexelMask** (uint32_t index, Shader::Mask mask)
- Shader::Mask **getTexelMask** (uint32_t index) const
- Traversal & **setTexelMasks** (uint32_t index, uint32_t num, Shader::Mask mask)
- Shader::Mask **getTexelMasks** (uint32_t index, uint32_t num) const
- uint32_t addTable (TableType type, uint32_t size, Shader::Mask mask, BindFlags flags=BindFlagNone)

  table parameters
- uint32_t **getNumTables** () const
- Traversal & **setTableOffset** (uint32_t offset)
- uint32_t **getTableOffset** () const
- Traversal & **setTableType** (uint32_t index, TableType type, uint32_t size, Shader::Mask mask, BindFlags flags=BindFlagNone)
- TableType **getTableType** (uint32_t index) const
- uint32_t **getTableSize** (uint32_t index) const
- Traversal & **setTableMask** (uint32_t index, Shader::Mask mask, BindFlags flags=BindFlagNone)
- Shader::Mask **getTableMask** (uint32_t index) const
- Traversal & **setTableFlags** (uint32_t index, BindFlags flags)
- BindFlags **getTableFlags** (uint32_t index) const
- void setRecursionDepth (uint32_t depth)

  recursion depth
- uint32_t **getRecursionDepth** () const

5.271.1 Detailed Description

The Traversal class manages the configuration of a ray-tracing pipeline, providing control over shaders, resource bindings, and pipeline states. It allows users to configure and retrieve various ray-tracing pipeline parameters, including samplers, textures, and buffer bindings. The class supports shader creation, loading, and compilation in multiple formats, such as native, GLSL, and SPIRV, offering fine-grained control over ray-tracing pipeline states.

5.272 Tellusim::uint16x8_t Struct Reference

`#include <math/TellusimSimd.h>`

Public Types

- enum { **Size** = 8 }

Public Member Functions

- uint16x8_t (const uint32x8_t &v)
- **uint16x8_t** (const uint16_t *v)
- **uint16x8_t** (const uint32x4_t &v0, const uint32x4_t &v1)
- **uint16x8_t** (uint16_t v)
- **uint16x8_t** (uint16_t x0, uint16_t y0, uint16_t z0, uint16_t w0, uint16_t x1, uint16_t y1, uint16_t z1, uint16_t w1)
- int16x8_t asi16x8 () const

  cast vector data
- int32x4_t **asi32x4** () const
- uint32x4_t asu32x4 () const

  cast vector data
- float16x8_t **asf16x8** () const
- float32x4_t **asf32x4** () const
- void set (const uint16x8_t &v)

  update vector data
- void **set** (uint16_t X0, uint16_t Y0, uint16_t Z0, uint16_t W0, uint16_t X1, uint16_t Y1, uint16_t Z1, uint16_t W1)
- void **set** (const uint16_t *1 v)
- void **get** (uint16_t *1 v) const
- template<uint32_t Index>
  void **set** (uint16_t V)
- template<uint32_t Index>
  uint16_t **get** () const
- uint16x8_t & operator*= (uint16_t v)

  vector to scalar operators
- uint16x8_t & **operator+=** (uint16_t v)
- uint16x8_t & **operator-=** (uint16_t v)
- uint16x8_t & **operator &=** (uint16_t v)
- uint16x8_t & **operator|=** (uint16_t v)
- uint16x8_t & **operator^=** (uint16_t v)
- uint16x8_t & **operator<<=** (uint16_t v)
- uint16x8_t & **operator>>=** (uint16_t v)
- uint16x8_t & operator*= (const uint16x8_t &v)

  vector to vector operators

- uint16x8_t & **operator+=** (const uint16x8_t &v)
- uint16x8_t & **operator-=** (const uint16x8_t &v)
- uint16x8_t & **operator &=** (const uint16x8_t &v)
- uint16x8_t & **operator|=** (const uint16x8_t &v)
- uint16x8_t & **operator^=** (const uint16x8_t &v)
- uint16x8_t xyzw10 () const

  swizzle vector
- uint16x8_t **zwxy01** () const
- uint16x8_t **yxwz01** () const
- uint32x4_t xyzw0 () const

  swizzle vector
- uint32x4_t **xyzw1** () const
- uint16_t sum () const

  sum vector components

Public Attributes

-
```
union {
  struct {
    uint16_t x0
    uint16_t y0
    uint16_t z0
    uint16_t w0
    uint16_t x1
    uint16_t y1
    uint16_t z1
    uint16_t w1
  }
  uint16_t v [Size]
};
```

5.272.1 Detailed Description

Vector of eight uint16_t components

5.272.2 Constructor & Destructor Documentation

5.272.2.1 uint16x8_t()

```
Tellusim::uint16x8_t::uint16x8_t (
            const uint32x8_t & v )  [explicit]
```

Vector of eight uint16_t components

5.273 Tellusim::uint32x4_t Struct Reference

```
#include <math/TellusimSimd.h>
```

Public Types

- enum { **Size** = 4 }

Public Member Functions

- uint32x4_t (const int32x4_t &v)
- **uint32x4_t** (const float32x4_t &v)
- **uint32x4_t** (const float64x4_t &v)
- **uint32x4_t** (const uint32_t *v)
- **uint32x4_t** (const uint32_t *v, uint32_t w)
- **uint32x4_t** (uint32_t v)
- **uint32x4_t** (uint32_t x, uint32_t y, uint32_t z, uint32_t w=0)
- int16x8_t asi16x8 () const

  cast vector data
- int32x4_t **asi32x4** () const
- uint16x8_t **asu16x8** () const
- float16x8_t asf16x8 () const

  cast vector data
- float32x4_t **asf32x4** () const
- void set (const uint32x4_t &v)

  update vector data
- void **set** (uint32_t X, uint32_t Y, uint32_t Z, uint32_t W)
- void **set** (const uint32_t *1 v, uint32_t W)
- void **set** (const uint32_t *1 v)
- void **get** (uint32_t *1 v) const
- template<uint32_t Index>
  void **set** (uint32_t V)
- template<uint32_t Index>
  uint32_t **get** () const
- template<uint32_t Index>
  uint32x4_t **get4** () const
- uint32x4_t & operator*= (uint32_t v)

  vector to scalar operators
- uint32x4_t & **operator+=** (uint32_t v)
- uint32x4_t & **operator-=** (uint32_t v)
- uint32x4_t & **operator &=** (uint32_t v)
- uint32x4_t & **operator|=** (uint32_t v)
- uint32x4_t & **operator^=** (uint32_t v)
- uint32x4_t & **operator<<=** (uint32_t v)
- uint32x4_t & **operator>>=** (uint32_t v)
- uint32x4_t & operator*= (const uint32x4_t &v)

  vector to vector operators
- uint32x4_t & **operator+=** (const uint32x4_t &v)
- uint32x4_t & **operator-=** (const uint32x4_t &v)
- uint32x4_t & **operator &=** (const uint32x4_t &v)
- uint32x4_t & **operator|=** (const uint32x4_t &v)
- uint32x4_t & **operator^=** (const uint32x4_t &v)
- uint32x4_t zwxy () const

  swizzle vector
- uint32x4_t **yxwz** () const
- uint32_t sum () const

  sum vector components

Public Attributes

- union {
  struct {
  uint32_t **x**
  uint32_t **y**
  uint32_t **z**
  uint32_t **w**
  }
  uint32_t **v** [Size]
  };

5.273.1 Detailed Description

Vector of four uint32_t components

5.273.2 Constructor & Destructor Documentation

5.273.2.1 uint32x4_t()

```
Tellusim::uint32x4_t::uint32x4_t (
            const int32x4_t & v )  [explicit]
```

Vector of four uint32_t components

5.274 Tellusim::uint32x8_t Struct Reference

```
#include <math/TellusimSimd.h>
```

Public Types

- enum { **Size** = 8 }

Public Member Functions

- uint32x8_t (const int32x8_t &v)
- **uint32x8_t** (const float32x8_t &v)
- **uint32x8_t** (const float64x8_t &v)
- **uint32x8_t** (const uint32_t *v)
- **uint32x8_t** (uint32_t v)
- **uint32x8_t** (const uint16x8_t &v)
- **uint32x8_t** (const uint32x4_t &v0, const uint32x4_t &v1)
- **uint32x8_t** (uint32_t x0, uint32_t y0, uint32_t z0, uint32_t w0, uint32_t x1, uint32_t y1, uint32_t z1, uint32_t w1)
- int32x8_t asi32x8 () const

  cast vector data
- float32x8_t **asf32x8** () const
- void set (const uint32x8_t &v)

  update vector data
- void **set** (uint32_t X0, uint32_t Y0, uint32_t Z0, uint32_t W0, uint32_t X1, uint32_t Y1, uint32_t Z1, uint32_t W1)
- void **set** (const uint32_t *1 v)
- void **get** (uint32_t *1 v) const
- template<uint32_t Index>
  void **set** (uint32_t V)
- template<uint32_t Index>
  uint32_t **get** () const
- template<uint32_t Index>
  uint32x8_t **get8** () const
- uint32x8_t & operator*= (uint32_t v)

  vector to scalar operators
- uint32x8_t & **operator+=** (uint32_t v)
- uint32x8_t & **operator-=** (uint32_t v)
- uint32x8_t & **operator &=** (uint32_t v)
- uint32x8_t & **operator**|**=** (uint32_t v)
- uint32x8_t & **operator**^**=** (uint32_t v)
- uint32x8_t & **operator**<<**=** (uint32_t v)
- uint32x8_t & **operator**>>**=** (uint32_t v)
- uint32x8_t & operator*= (const uint32x8_t &v)

  vector to vector operators
- uint32x8_t & **operator+=** (const uint32x8_t &v)
- uint32x8_t & **operator-=** (const uint32x8_t &v)
- uint32x8_t & **operator &=** (const uint32x8_t &v)
- uint32x8_t & **operator**|**=** (const uint32x8_t &v)
- uint32x8_t & **operator**^**=** (const uint32x8_t &v)
- uint32x8_t xyzw10 () const

  swizzle vector
- uint32x8_t **zwxy01** () const
- uint32x8_t **yxwz01** () const
- uint32x4_t **xyzw0** () const
- uint32x4_t **xyzw1** () const
- uint32_t sum () const

  sum vector components

Public Attributes

- union {
  struct {
  uint32_t **x0**
  uint32_t **y0**
  uint32_t **z0**
  uint32_t **w0**
  uint32_t **x1**
  uint32_t **y1**
  uint32_t **z1**
  uint32_t **w1**
  }
  uint32_t **v** [Size]
  };

5.274.1 Detailed Description

Vector of eight uint32_t components

5.274.2 Constructor & Destructor Documentation

5.274.2.1 uint32x8_t()

```
Tellusim::uint32x8_t::uint32x8_t (
            const int32x8_t & v )  [explicit]
```

Vector of eight uint32_t components

5.275 Tellusim::UnrefType< RefType > Struct Template Reference

```
#include <TellusimBase.h>
```

5.275.1 Detailed Description

template<class RefType>
struct Tellusim::UnrefType< RefType >

Move semantic

5.276 Tellusim::UnrefType< const RefType & > Struct Template Reference

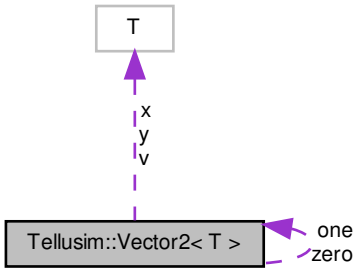
5.277 Tellusim::UnrefType< RefType & > Struct Template Reference

5.278 Tellusim::UnrefType< RefType && > Struct Template Reference

5.279 Tellusim::Vector2< T > Struct Template Reference

`#include <math/TellusimVector.h>`

Collaboration diagram for Tellusim::Vector2< T >:



Public Types

- enum { **Size** = 2 }

Public Member Functions

- **Vector2** (const Vector2 &v)
- **Vector2** (const Type &x, const Type &y)
- template<class CType >
  Vector2 (const Vector2< CType > &v)
- template<class CType >
  Vector2 (const Vector3< CType > &v)
- template<class CType >
  Vector2 (const Vector4< CType > &v)
- **Vector2** (const Type *1 v)
- **Vector2** (const Type &v)
- void set (const Type &v)

  update vector data
- void **set** (const Type &X, const Type &Y)
- void **set** (const Vector3< Type > &v)
- void **set** (const Vector4< Type > &v)
- void **set** (const Type *1 v)
- void **get** (Type *1 v) const

- Vector2 & operator*= (const Type &v)

  vector to scalar operators
- Vector2 & **operator/=** (const Type &v)
- Vector2 & **operator%=** (const Type &v)
- Vector2 & **operator+=** (const Type &v)
- Vector2 & **operator-=** (const Type &v)
- Vector2 & **operator &=** (const Type &v)
- Vector2 & **operator|=** (const Type &v)
- Vector2 & **operator^=** (const Type &v)
- Vector2 & **operator<<=** (const Type &v)
- Vector2 & **operator>>=** (const Type &v)
- Vector2 & operator*= (const Vector2 &v)

  vector to vector operators
- Vector2 & **operator/=** (const Vector2 &v)
- Vector2 & **operator%=** (const Vector2 &v)
- Vector2 & **operator+=** (const Vector2 &v)
- Vector2 & **operator-=** (const Vector2 &v)
- Vector2 & **operator &=** (const Vector2 &v)
- Vector2 & **operator|=** (const Vector2 &v)
- Vector2 & **operator^=** (const Vector2 &v)
- const Type & operator[ ] (uint32_t index) const

  vector data
- Type & **operator[ ]** (uint32_t index)
- Type cartesian () const

  homogeneous transform

Public Attributes

-
```
union {
  struct {
    Type x
    Type y
  }
  Type v [Size]
};
```

Static Public Attributes

- static const Vector2 zero

  default vectors
- static const Vector2 **one**

5.279.1 Detailed Description

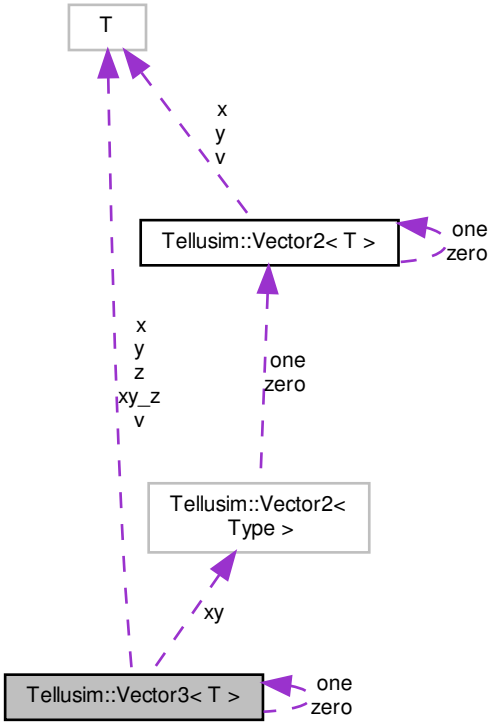
template<class T>
struct Tellusim::Vector2< T >

Vector2 class

5.280 Tellusim::Vector3< T > Struct Template Reference

`#include <math/TellusimVector.h>`

Collaboration diagram for Tellusim::Vector3< T >:



Public Types

- enum { **Size** = 3 }

Public Member Functions

- **Vector3** (const Vector3 &v)
- **Vector3** (const Type &x, const Type &y, const Type &z)
- template<class CType >
  Vector3 (const Vector2< CType > &v, const CType &z)
- template<class CType >
  Vector3 (const Vector3< CType > &v)
- template<class CType >
  Vector3 (const Vector4< CType > &v)
- **Vector3** (const Type *1 v)
- **Vector3** (const Type &v)
- void set (const Type &v)

update vector data

- void **set** (const Type &X, const Type &Y, const Type &Z)
- void **set** (const Vector2< Type > &v, const Type &Z)
- void **set** (const Vector4< Type > &v)
- void **set** (const Type *1 v)
- void **get** (Type *1 v) const
- Vector3 & operator*= (const Type &v)

vector to scalar operators

- Vector3 & **operator/=** (const Type &v)
- Vector3 & **operator%=** (const Type &v)
- Vector3 & **operator+=** (const Type &v)
- Vector3 & **operator-=** (const Type &v)
- Vector3 & **operator &=** (const Type &v)
- Vector3 & **operator|=** (const Type &v)
- Vector3 & **operator^=** (const Type &v)
- Vector3 & **operator<<=** (const Type &v)
- Vector3 & **operator>>=** (const Type &v)
- Vector3 & operator*= (const Vector3 &v)

vector to vector operators

- Vector3 & **operator/=** (const Vector3 &v)
- Vector3 & **operator%=** (const Vector3 &v)
- Vector3 & **operator+=** (const Vector3 &v)
- Vector3 & **operator-=** (const Vector3 &v)
- Vector3 & **operator &=** (const Vector3 &v)
- Vector3 & **operator|=** (const Vector3 &v)
- Vector3 & **operator^=** (const Vector3 &v)
- const Type & operator[ ] (uint32_t index) const

vector data

- Type & **operator[ ]** (uint32_t index)
- Vector2< Type > cartesian () const

homogeneous transform

Public Attributes

-
```
union {
  struct {
    Type x
    Type y
    Type z
  }
  struct {
    Vector2< Type > xy
    Type xy_z
  }
  Type v [Size]
};
```

Static Public Attributes

- static const Vector3 zero

default vectors

- static const Vector3 **one**

5.280.1 Detailed Description

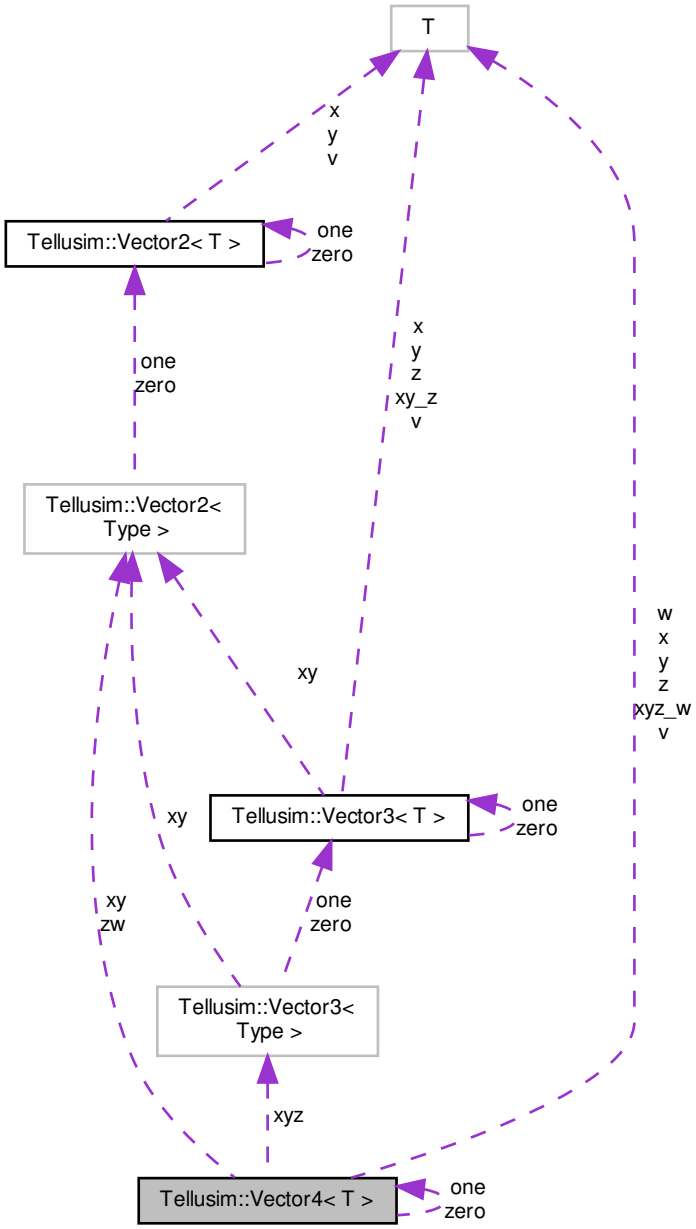
template<class T>
struct Tellusim::Vector3< T >

Vector3 class

5.281 Tellusim::Vector4< T > Struct Template Reference

```
#include <math/TellusimVector.h>
```

Collaboration diagram for Tellusim::Vector4< T >:



Public Types

- enum { **Size** = 4 }

Public Member Functions

- **Vector4** (const Vector4 &v)

- **Vector4** (const Type &x, const Type &y, const Type &z, const Type &w)
- template<class CType >
  Vector4 (const Vector2< CType > &v0, const Vector2< CType > &v1)
- template<class CType >
  Vector4 (const Vector2< CType > &v, const CType &z, const CType &w)
- template<class CType >
  Vector4 (const Vector3< CType > &v, const CType &w)
- template<class CType >
  Vector4 (const Vector4< CType > &v)
- **Vector4** (const Type *1 v)
- **Vector4** (const Type &v)
- void set (const Type &v)

  update vector data
- void **set** (const Type &X, const Type &Y, const Type &Z, const Type &W)
- void **set** (const Vector2< Type > &v, const Type &Z, const Type &W)
- void **set** (const Vector3< Type > &v, const Type &W)
- void **set** (const Type *1 v)
- void **get** (Type *1 v) const
- Vector4 & operator*= (const Type &v)

  vector to scalar operators
- Vector4 & **operator/=** (const Type &v)
- Vector4 & **operator%=** (const Type &v)
- Vector4 & **operator+=** (const Type &v)
- Vector4 & **operator-=** (const Type &v)
- Vector4 & **operator &=** (const Type &v)
- Vector4 & **operator|=** (const Type &v)
- Vector4 & **operator^=** (const Type &v)
- Vector4 & **operator<<=** (const Type &v)
- Vector4 & **operator>>=** (const Type &v)
- Vector4 & operator*= (const Vector4 &v)

  vector to vector operators
- Vector4 & **operator/=** (const Vector4 &v)
- Vector4 & **operator%=** (const Vector4 &v)
- Vector4 & **operator+=** (const Vector4 &v)
- Vector4 & **operator-=** (const Vector4 &v)
- Vector4 & **operator &=** (const Vector4 &v)
- Vector4 & **operator|=** (const Vector4 &v)
- Vector4 & **operator^=** (const Vector4 &v)
- const Type & operator[ ] (uint32_t index) const

  vector data
- Type & **operator[ ]** (uint32_t index)
- Vector3< Type > cartesian () const

  homogeneous transform

Public Attributes

-

```
union {
  struct {
    Type x
    Type y
    Type z
    Type w
  }
  struct {
    Vector2< Type > xy
    Vector2< Type > zw
  }
  struct {
    Vector3< Type > xyz
    Type xyz_w
  }
  Type v [Size]
};
```

Static Public Attributes

- static const Vector4 zero
  default vectors
- static const Vector4 **one**

5.281.1 Detailed Description

template<class T>
struct Tellusim::Vector4< T >

Vector4 class

5.282 Tellusim::VectorN< Type, N > Struct Template Reference

`#include <math/TellusimNumerical.h>`

Public Member Functions

- **VectorN** (const VectorN &vector)
- **VectorN** (uint32_t size)
- **VectorN** (const Type &value, uint32_t size=N)
- **VectorN** (const Type *vector, uint32_t size=N)
- **VectorN** (const InitializerList< Type > &list)
- template<class CType >
  VectorN (const VectorN< CType, N > &vector)
- void set (const Type &value, uint32_t size=N)
  update vector data
- void **set** (const Type *1 vector, uint32_t size=N)
- void **set** (const VectorN &vector)
- void **set** (const InitializerList< Type > &list)

- void **get** (Type *1 vector, uint32_t size=N)
- VectorN & operator*= (const Type &value)

  vector to scalar operators
- VectorN & **operator/=** (const Type &value)
- VectorN & **operator+=** (const Type &value)
- VectorN & **operator-=** (const Type &value)
- VectorN & operator= (const VectorN &vector)

  vector to vector operators
- VectorN & **operator*=** (const VectorN &vector)
- VectorN & **operator/=** (const VectorN &vector)
- VectorN & **operator+=** (const VectorN &vector)
- VectorN & **operator-=** (const VectorN &vector)
- const Type & operator[ ] (uint32_t index) const

  vector data
- Type & **operator[ ]** (uint32_t index)

Public Attributes

- Type **v** [N]
- uint32_t **Size** = N

5.282.1 Detailed Description

template<class Type, uint32_t N>
struct Tellusim::VectorN< Type, N >

VectorN class

5.283 Tellusim::Viewport Struct Reference

`#include <TellusimTypes.h>`

Public Member Functions

- **Viewport** (float32_t width, float32_t height)
- **Viewport** (float32_t x, float32_t y, float32_t width, float32_t height)
- **Viewport** (float32_t x, float32_t y, float32_t width, float32_t height, float32_t znear, float32_t zfar)
- float32_t **getLeft** () const
- float32_t **getBottom** () const
- float32_t **getRight** () const
- float32_t **getTop** () const

Public Attributes

- float32_t **x** = 0.0f
- float32_t **y** = 0.0f
- float32_t **width** = 1.0f
- float32_t **height** = 1.0f
- float32_t **znear** = 0.0f
- float32_t **zfar** = 1.0f

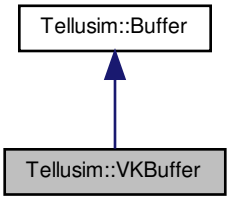
5.283.1 Detailed Description

The Viewport struct represents a 2D rectangular area in a 3D space used for rendering. It defines the position and size of the viewport as well as the near and far clipping planes. The struct contains four main properties: x, y, width, and height, which describe the position and dimensions of the viewport. It also includes znear and zfar properties, which define the near and far planes for depth clipping.

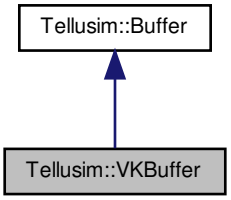
5.284 Tellusim::VKBuffer Class Reference

`#include <platform/TellusimBuffer.h>`

Inheritance diagram for Tellusim::VKBuffer:



Collaboration diagram for Tellusim::VKBuffer:



Public Member Functions

- bool create (Flags flags, size_t size, VkBuffer buffer, uint32_t access)
  create external buffer
- VkBuffer **getVKBuffer** () const
- VkBufferView **getBufferView** () const
- uint64_t **getBufferAddress** () const
- void **setBufferAccess** (uint32_t access)
- uint32_t **getBufferAccess** () const
- void * **getSharedPtr** () const
- void * **getInteropHandle** () const

Additional Inherited Members

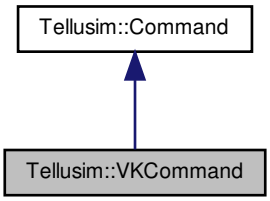
5.284.1 Detailed Description

The VKBuffer class is a Vulkan-specific implementation of the Buffer class, providing access to Vulkan buffer resources and views. It allows for the creation of external buffers, specifying Vulkan buffer handles and access modes, and includes methods for managing buffer access, retrieving the buffer address, and obtaining shared pointers for interoperability. The class also inherits the create method from the Buffer class, facilitating the initialization of buffers in Vulkan-based applications.

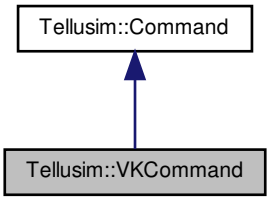
5.285 Tellusim::VKCommand Class Reference

```
#include <platform/TellusimCommand.h>
```

Inheritance diagram for Tellusim::VKCommand:



Collaboration diagram for Tellusim::VKCommand:



Public Member Functions

- VkCommandBuffer getVKCommand () const

  command context
- VkDescriptorSet getSamplerDescriptor () const

command descriptors

- VkDescriptorSet **getImageDescriptor** () const
- VkDescriptorSet **getBufferDescriptor** () const
- VkDescriptorSet **getTracingDescriptor** () const
- VkDescriptorSet **getTexelDescriptor** () const
- void update ()

  update resources

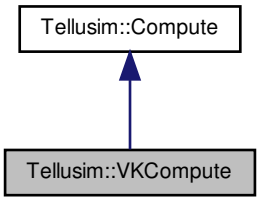
5.285.1 Detailed Description

The VKCommand class is a Vulkan-specific implementation of the Command class, providing access to the command buffer.

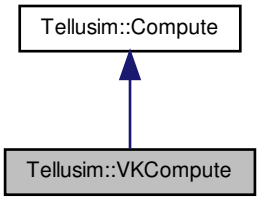
5.286 Tellusim::VKCompute Class Reference

`#include <platform/TellusimCompute.h>`

Inheritance diagram for Tellusim::VKCompute:



Collaboration diagram for Tellusim::VKCompute:



Public Member Functions

- VkCommandBuffer getCommand () const
  command context
- VkDescriptorSet getSamplerDescriptor () const
  compute descriptors
- VkDescriptorSet **getImageDescriptor** () const
- VkDescriptorSet **getBufferDescriptor** () const
- VkDescriptorSet **getTracingDescriptor** () const
- VkDescriptorSet **getTexelDescriptor** () const
- void update ()
  update resources

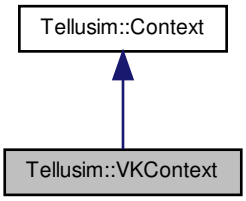
5.286.1 Detailed Description

The VKCompute class is a Vulkan-specific implementation of the Compute class, providing access to the command buffer.

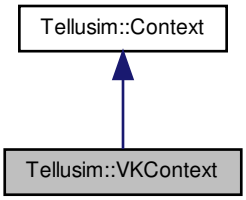
5.287 Tellusim::VKContext Class Reference

```
#include <platform/TellusimContext.h>
```

Inheritance diagram for Tellusim::VKContext:



Collaboration diagram for Tellusim::VKContext:



Public Member Functions

- bool create (VkInstance instance, PFN_vkGetInstanceProcAddr func, VkPhysicalDevice adapter, VkDevice device, uint32_t family, uint32_t index)

  create context
- VkInstance getInstance () const

  current device
- VkPhysicalDevice **getAdapter** () const
- VkDevice **getDevice** () const
- VkQueue getQueue () const

  current context
- VkCommandBuffer **getCommand** () const
- uint32_t **getFamily** () const
- uint32_t getNumQueues ()

  device queues
- uint32_t **getQueueFlags** (uint32_t index)
- uint32_t **getQueueFamily** (uint32_t index)

Static Public Member Functions

- static void addContextExtension (const char *name)

  additional extensions
- static void **addAdapterExtension** (const char *name)
- static void **addAdapterFeatures** (void *features)
- static PFN_vkGetInstanceProcAddr getInstanceProcAddress ()

  get proc address functions
- static PFN_vkGetDeviceProcAddr **getDeviceProcAddress** ()
- static void * getProcAddress (const char *name)

  Vulkan functions.
- static bool error (uint32_t result)

  check Vulkan errors

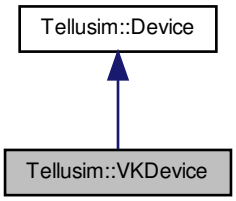
5.287.1 Detailed Description

The VKContext class is a Vulkan-specific implementation of the Context class. It initializes the rendering context using externally provided Vulkan instance, device, and queue parameters. The class provides access to the Vulkan instance, physical device, logical device, command queue, and command buffer.

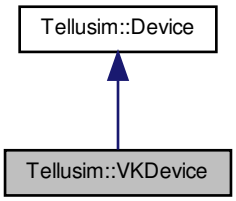
5.288 Tellusim::VKDevice Class Reference

```
#include <platform/TellusimDevice.h>
```

Inheritance diagram for Tellusim::VKDevice:



Collaboration diagram for Tellusim::VKDevice:



Public Member Functions

- **VKDevice** (Context &context)
- **VKDevice** (Surface &surface)
- **VKDevice** (Window &window)
- void setBufferAccess (Buffer &buffer, uint32_t access)

  buffer access
- void setTextureLayout (Texture &texture, uint32_t layout)

  texture layout
- bool waitVKFence (void ∗fence, uint64_t timeout, bool reset) const

  fence synchronization
- bool **signalVKFence** (void ∗fence) const
- void waitSemaphore (void ∗semaphore, uint32_t mask) const

  semaphore synchronization
- void **signalSemaphore** (void ∗semaphore) const
- bool hasMemoryType (uint32_t flags) const

  memory types
- uint32_t **getMemoryIndex** (uint32_t types, uint32_t flags) const
- VkInstance getInstance () const

  command context

- VkPhysicalDevice **getAdapter** () const
- VkDevice **getVKDevice** () const
- VkQueue **getQueue** () const
- VkCommandBuffer **getCommand** () const
- uint32_t **getFamily** () const

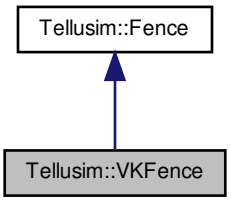
5.288.1 Detailed Description

The VKDevice class extends the Device class to provide Vulkan-specific functionality for managing a rendering device. It offers methods for buffer access control, texture layout management, and synchronization through fences and semaphores. The class supports querying memory types and obtaining the appropriate memory index for specific flags, allowing for fine-tuned memory management in Vulkan applications.

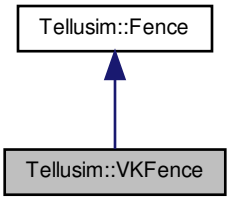
5.289 Tellusim::VKFence Class Reference

```
#include <platform/TellusimFence.h>
```

Inheritance diagram for Tellusim::VKFence:



Collaboration diagram for Tellusim::VKFence:



Public Member Functions

- VkFence **getVKFence** () const
- VkSemaphore **getSemaphore** () const
- void * **getSharedHandle** () const

Additional Inherited Members

5.289.1 Detailed Description

The VKFence class is a Vulkan-specific implementation of the Fence class, providing access to Vulkan fence and semaphore objects for synchronization in Vulkan applications. It allows interaction with Vulkan native synchronization mechanisms, enabling the management of fence states and ensuring proper synchronization between operations. The class provides access to the VkFence and VkSemaphore objects, which are essential for signaling and waiting on events in Vulkan pipelines.

5.290 Tellusim::VKTracing::VKInstance Struct Reference

tracing instance

```
#include <platform/TellusimTracing.h>
```

Public Attributes

- float32_t **transform** [12]
- uint32_t **data**: 24
- uint32_t **mask**: 8
- uint32_t **offset**: 24
- uint32_t **flags**: 8
- uint64_t **address**

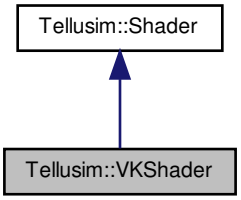
5.290.1 Detailed Description

tracing instance

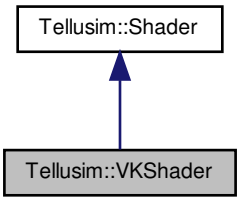
5.291 Tellusim::VKShader Class Reference

```
#include <platform/TellusimShader.h>
```

Inheritance diagram for Tellusim::VKShader:



Collaboration diagram for Tellusim::VKShader:



Public Member Functions

- VkShaderModule **getModule** () const

Additional Inherited Members

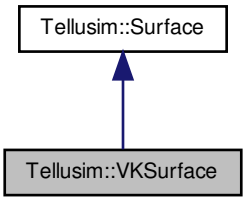
5.291.1 Detailed Description

The VKShader class extends the Shader class to specialize in managing shaders for Vulkan. It provides a method to retrieve the Vulkan shader module, enabling integration with Vulkan.

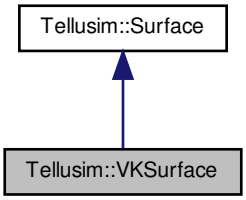
5.292 Tellusim::VKSurface Class Reference

`#include <platform/TellusimSurface.h>`

Inheritance diagram for Tellusim::VKSurface:



Collaboration diagram for Tellusim::VKSurface:



Public Member Functions

- **VKSurface** (VKContext &context)
- VkInstance getInstance () const

  current device
- VkPhysicalDevice **getAdapter** () const
- VkDevice **getDevice** () const
- VkQueue **getQueue** () const
- VkCommandBuffer **getCommand** () const
- uint32_t **getFamily** () const
- void setColorImage (VkImage image)

  image handles
- void **setDepthImage** (VkImage image)
- VkImage **getColorImage** () const
- VkImage **getDepthImage** () const
- void setColorImageView (VkImageView image_view)

  image view handles

- void **setDepthImageView** (VkImageView image_view)
- VkImageView **getColorImageView** () const
- VkImageView **getDepthImageView** () const
- void setRenderPass (VkRenderPass render_pass)

  framebuffer handle
- void **setFramebuffer** (VkFramebuffer framebuffer)
- VkRenderPass **getRenderPass** () const
- VkFramebuffer **getFramebuffer** () const
- uint32_t getColorPixelFormat () const

  surface formats
- uint32_t **getDepthPixelFormat** () const

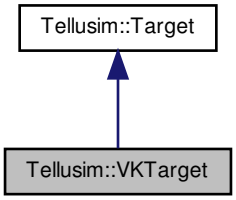
5.292.1 Detailed Description

The VKSurface class extends the Surface class to provide Vulkan-specific functionality for managing a rendering surface. It includes methods for interacting with Vulkan instances, physical devices, logical devices, command buffers, and queues, enabling rendering operations in the context of Vulkan. The class supports managing images, image views, render passes, and framebuffers, which are essential for rendering operations.

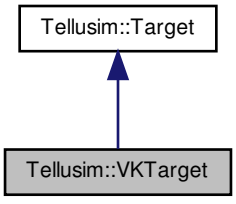
5.293 Tellusim::VKTarget Class Reference

```
#include <platform/TellusimTarget.h>
```

Inheritance diagram for Tellusim::VKTarget:



Collaboration diagram for Tellusim::VKTarget:



Public Member Functions

- VkRenderPass **getRenderPass** () const
- VkFramebuffer **getFramebuffer** () const

Additional Inherited Members

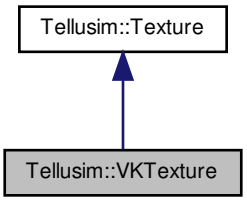
5.293.1 Detailed Description

The VKTarget class is a Vulkan-specific implementation of the Target class, offering functionality for managing Vulkan render passes and framebuffers. It provides methods to retrieve the associated Vulkan render pass and framebuffer objects, which are essential for the rendering process in Vulkan.

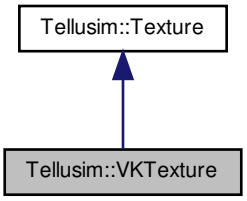
5.294 Tellusim::VKTexture Class Reference

```
#include <platform/TellusimTexture.h>
```

Inheritance diagram for Tellusim::VKTexture:



Collaboration diagram for Tellusim::VKTexture:



Public Member Functions

- bool create (Type type, uint32_t format, VkImage texture, uint32_t layout, Flags flags=DefaultFlags, Format texture_format=FormatUnknown)

  create external texture
- uint32_t **getPixelFormat** () const
- VkImage **getVKTexture** () const
- VkImageView **getTextureView** () const
- void **setTextureLayout** (uint32_t layout)
- uint32_t **getTextureLayout** () const
- void **getTextureRange** (void *range, const Slice &slice) const
- void * **getSharedPtr** () const
- void * **getInteropHandle** () const

Additional Inherited Members

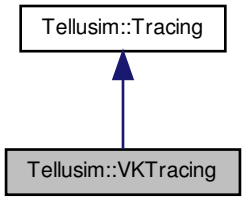
5.294.1 Detailed Description

The VKTexture class is a Vulkan-specific implementation of the Texture class, providing access to Vulkan texture resources and views. It enables the creation of external textures, specifying Vulkan image handles, formats, and layouts, while supporting various texture formats and flags. This class allows for managing texture layouts, retrieving texture views, and interacting with Vulkan-specific texture ranges, along with shared pointer functionality for interoperability. The class also inherits the create method from the Texture class, facilitating the initialization of textures in Vulkan-based applications.VKTexture

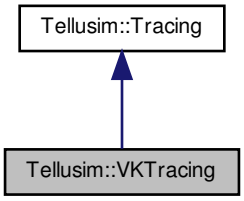
5.295 Tellusim::VKTracing Class Reference

```
#include <platform/TellusimTracing.h>
```

Inheritance diagram for Tellusim::VKTracing:



Collaboration diagram for Tellusim::VKTracing:



Classes

- struct VKInstance
  tracing instance

Public Member Functions

- void * **getBuildGeometryInfo** () const
- void * **getBuildSizeInfo** () const
- Buffer **getTracingBuffer** () const
- VkAccelerationStructureKHR **getAccelerationStructure** () const

Additional Inherited Members

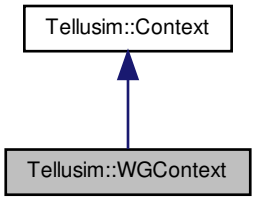
5.295.1 Detailed Description

The VKTracing class is a Vulkan-specific implementation of the Tracing class. It provides methods and structures for managing ray-tracing acceleration structures within the Vulkan API.

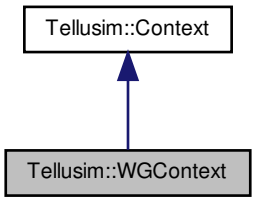
5.296 Tellusim::WGContext Class Reference

`#include <platform/TellusimContext.h>`

Inheritance diagram for Tellusim::WGContext:



Collaboration diagram for Tellusim::WGContext:



Public Member Functions

- bool create (WGPUInstance instance, WGPUAdapter adapter, WGPUDevice device)

  create context
- WGPUInstance getInstance () const

  current context
- WGPUAdapter **getAdapter** () const
- WGPUDevice **getDevice** () const

Static Public Member Functions

- static bool open (WGPUInstance instance, WGPUAdapter adapter, WGPUDevice device)

  open context

5.296.1 Detailed Description

The WGContext class is a WebGPU-specific implementation of the Context class. It facilitates the creation of a rendering context using an externally provided WebGPU instance, adapter, and device. The class provides access to the underlying WebGPU instance, adapter, and device.

5.297 Tellusim::Window Class Reference

```
#include <platform/TellusimWindow.h>
```

Public Types

- enum Flags {
  FlagNone = 0,
  FlagTitle = (1 << 0),
  FlagClose = (1 << 1),
  FlagFrame = (1 << 2),
  FlagResize = (1 << 3),
  FlagMinimize = (1 << 4),
  FlagMaximize = (1 << 5),
  FlagTransient = (1 << 6),
  FlagFullscreen = (1 << 7),
  FlagTransparent = (1 << 8),
  FlagFileDropped = (1 << 9),
  FlagMultisample2 = (1 << 10),
  FlagMultisample4 = (1 << 11),
  FlagMultisample8 = (1 << 12),
  FlagVerticalSync = (1 << 13),
  FlagRefreshSync = (1 << 14),
  FlagColorRGBAu8ns = (1 << 15),
  FlagColorRGBu10Au2n = (1 << 16),
  FlagColorRGBAf16 = (1 << 17),
  FlagMultisample = (FlagMultisample2 | FlagMultisample4 | FlagMultisample8),
  DefaultFlags = (FlagTitle | FlagClose | FlagResize | FlagMinimize | FlagMaximize),
  NumFlags = 18 }

  Window flags.
- enum Cursor {
  CursorArrow = 0,
  CursorInvalid,
  CursorLeft,
  CursorRight,
  CursorBottom,
  CursorTop,
  CursorWidth,
  CursorHeight,
  CursorMajor,
  CursorMinor,
  CursorAll,
  NumCursors }

  Mouse cursors.
- enum Button {
  ButtonNone = 0,
  ButtonLeft = (1 << 0),
  ButtonLeft2 = (1 << 1),
  ButtonRight = (1 << 2),
  ButtonRight2 = (1 << 3),
  ButtonMiddle = (1 << 4),
  ButtonMiddle2 = (1 << 5),
  ButtonAux = (1 << 6),
  ButtonAux2 = (1 << 7),
  NumButtons = 8 }

  Mouse buttons.
- enum Axis {
  AxisX = 0,
  AxisY,
  AxisZ,
  AxisW,
  NumAxes }

Mouse axes.

- enum { **NumTouches** = 16 }

  Screen touches.

- enum Key {
  KeyNone = 128,
  KeyEsc,
  KeyTab,
  KeyBackspace,
  KeyDelete,
  KeyInsert,
  KeyReturn,
  KeyPause,
  KeyPrior,
  KeyNext,
  KeyEnd,
  KeyHome,
  KeyUp,
  KeyDown,
  KeyLeft,
  KeyRight,
  KeyNum,
  KeyCaps,
  KeyScroll,
  KeyShift,
  KeyCtrl,
  KeyAlt,
  KeyWin,
  KeyCmd,
  KeyMenu,
  KeyF1,
  KeyF2,
  KeyF3,
  KeyF4,
  KeyF5,
  KeyF6,
  KeyF7,
  KeyF8,
  KeyF9,
  KeyF10,
  KeyF11,
  KeyF12,
  NumKeys,
  KeyOption = KeyCtrl }

  Keyboard keys.

- using MousePressedCallback = Function< void(Button button)>

  mouse pressed callback

- using MouseReleasedCallback = Function< void(Button button)>

  mouse released callback

- using MouseChangedCallback = Function< void(int32_t x, int32_t y)>

  mouse changed callback

- using MouseRotatedCallback = Function< void(Axis axis, float32_t delta)>

  mouse rotated callback

- using TouchChangedCallback = Function< void()>

  touch changed callback

- using KeyboardPressedCallback = Function< void(uint32_t key, uint32_t code)>

  keyboard pressed callback

- using KeyboardReleasedCallback = Function< void(uint32_t key)>
  keyboard released callback
- using SizeChangedCallback = Function< void(uint32_t width, uint32_t height)>
  size changed callback
- using FocusChangedCallback = Function< void(bool changed)>
  focus changed callback
- using CloseClickedCallback = Function< void()>
  close clicked callback
- using PauseChangedCallback = Function< void(bool paused)>
  pause changed callback
- using FileDroppedCallback = Function< void(const char *name, uint32_t remain)>
  file dropped callback
- using UpdateCallback = Function< void()>
  update callback
- using PresentCallback = Function< void()>
  present callback
- using MainLoopCallback = Function< bool()>
  window main loop callback

Public Member Functions

- Window ()
  window constructor
- **Window** (Platform platform, uint32_t index=Maxu32)
- **Window** (Surface &surface)
- Platform getPlatform () const
  window platform
- const char * **getPlatformName** () const
- uint32_t getIndex () const
  window device index
- void setSurface (Surface &surface)
  window surface
- Surface **getSurface** () const
- void * getHandle () const
  window handle
- virtual bool isCreated () const
  check window
- virtual bool create (const char *title, Flags flags=DefaultFlags)
  create window
- virtual bool **create** (const String &title, Flags flags=DefaultFlags)
- virtual bool **create** (Flags flags=DefaultFlags)
- virtual void **release** ()
- bool clear (const Color &color)
  clear window
- bool grab (Image &image) const
  grab window
- virtual bool render ()
  render window
- virtual bool **present** ()
- virtual bool **finish** ()
- Format getColorFormat () const

window format

- Format **getDepthFormat** () const
- uint32_t **getMultisample** () const
- bool **hasMultisample** () const
- virtual void setFlags (Flags flags)

  window flags
- Flags **getFlags** () const
- bool **hasFlag** (Flags flags) const
- bool **hasFlags** (Flags flags) const
- void setRefreshRate (uint32_t rate)

  window refresh rate
- uint32_t **getRefreshRate** () const
- virtual bool setHidden (bool hidden)

  hide window
- bool **isHidden** () const
- virtual bool setFocused (bool focused)

  focus window
- bool **isFocused** () const
- virtual bool setMinimized (bool minimized)

  minimize window
- bool **isMinimized** () const
- virtual bool setFullscreen (bool fullscreen)

  fullscreen window
- bool **isFullscreen** () const
- bool isOccluded () const

  occluded window
- virtual bool setTitle (const char *title)

  window title
- virtual bool **setTitle** (const String &title)
- String **getTitle** () const
- virtual bool setIcon (const Image &image)

  window icon image
- Image **getIcon** () const
- virtual bool setGeometry (int32_t x, int32_t y, uint32_t width, uint32_t height, bool force=false)

  window geometry
- virtual bool **setPosition** (int32_t x, int32_t y, bool force=false)
- int32_t **getPositionX** (bool title=false) const
- int32_t **getPositionY** (bool title=false) const
- virtual bool **setSize** (uint32_t width, uint32_t height, bool force=false)
- uint32_t **getWidth** () const
- uint32_t **getHeight** () const
- float32_t **getScale** () const
- uint32_t **getDpiX** () const
- uint32_t **getDpiY** () const
- virtual bool setMouse (int32_t x, int32_t y, bool force=false)

  mouse position
- int32_t **getMouseX** () const
- int32_t **getMouseY** () const
- bool **setMouseDelta** (int32_t dx, int32_t dy)
- int32_t **getMouseDX** () const
- int32_t **getMouseDY** () const
- virtual bool setMouseHidden (bool hidden, bool force=false)

  mouse hidden flag

- bool **isMouseHidden** () const
- virtual bool setMouseClipped (bool clipped, bool force=false)

  mouse clipped flag
- bool **isMouseClipped** () const
- bool **isMouseInside** () const
- virtual bool setMouseCursor (Cursor cursor, bool force=false)

  mouse cursor
- Cursor **getMouseCursor** () const
- bool setMouseButtons (Button buttons)

  mouse buttons
- Button **getMouseButtons** () const
- bool **setMouseButton** (Button button, bool value)
- bool **getMouseButton** (Button button, bool clear=false) const
- void **releaseMouseButtons** (Button buttons)
- Button **clearMouseButtons** ()
- bool setMouseAxis (Axis axis, float32_t value)

  mouse axes
- float32_t **getMouseAxis** (Axis axis) const
- float32_t **clearMouseAxis** (Axis axis)
- void **setMousePressedCallback** (const MousePressedCallback &func)
- MousePressedCallback **getMousePressedCallback** () const
- void **setMouseReleasedCallback** (const MouseReleasedCallback &func)
- MouseReleasedCallback **getMouseReleasedCallback** () const
- void **setMouseChangedCallback** (const MouseChangedCallback &func)
- MouseChangedCallback **getMouseChangedCallback** () const
- void **setMouseRotatedCallback** (const MouseRotatedCallback &func)
- MouseRotatedCallback **getMouseRotatedCallback** () const
- uint32_t getNumTouches () const

  touches
- uint32_t **addTouch** (int32_t x, int32_t y)
- int32_t **getTouchX** (uint32_t touch) const
- int32_t **getTouchY** (uint32_t touch) const
- uint32_t **findTouch** (int32_t x, int32_t y) const
- void **clearTouches** ()
- void **setTouchChangedCallback** (const TouchChangedCallback &func)
- TouchChangedCallback **getTouchChangedCallback** () const
- void setKeyboardKey (uint32_t key, bool value)

  keyboard keys
- bool **getKeyboardKey** (uint32_t key, bool clear=false) const
- void **setKeyboardPressedCallback** (const KeyboardPressedCallback &func)
- KeyboardPressedCallback **getKeyboardPressedCallback** () const
- void **setKeyboardReleasedCallback** (const KeyboardReleasedCallback &func)
- KeyboardReleasedCallback **getKeyboardReleasedCallback** () const
- void **setSizeChangedCallback** (const SizeChangedCallback &func)
- SizeChangedCallback **getSizeChangedCallback** () const
- void **setFocusChangedCallback** (const FocusChangedCallback &func)
- FocusChangedCallback **getFocusChangedCallback** () const
- void **setCloseClickedCallback** (const CloseClickedCallback &func)
- CloseClickedCallback **getCloseClickedCallback** () const
- void **setPauseChangedCallback** (const PauseChangedCallback &func)
- PauseChangedCallback **getPauseChangedCallback** () const
- void **setFileDroppedCallback** (const FileDroppedCallback &func)
- FileDroppedCallback **getFileDroppedCallback** () const
- void **setUpdateCallback** (const UpdateCallback &func)

- UpdateCallback **getUpdateCallback** () const
- void **setPresentCallback** (const PresentCallback &func)
- PresentCallback **getPresentCallback** () const
- MainLoopCallback **getMainLoopCallback** () const
- virtual bool **run** (const MainLoopCallback &func)
- virtual bool **isRunning** () const
- virtual void **stop** ()
- bool setCopyText (const char *text)

  window copy/paste buffer
- bool **setCopyText** (const String &text)
- String **getPasteText** () const

Static Public Member Functions

- static uint32_t getNumWindows ()

  all windows
- static Window **getWindow** (uint32_t index)
- static void update (bool wait=false)

  update windows

5.297.1 Detailed Description

The Window class provides a cross-platform abstraction for creating, managing, and interacting with application windows. It allows the user to configure various window properties, such as title, size, visibility, and multisample support. The class includes methods for handling mouse input, keyboard events, touch interactions, and other common window features like fullscreen and transparency. This class provides essential functionality for building graphical user interfaces and handling real-time user input in applications across different platforms.

5.298 Tellusim::Xml Class Reference

`#include <format/TellusimXml.h>`

Public Member Functions

- **Xml** (const char *name, const char *attributes=nullptr)
- **Xml** (const String &name, const char *attributes=nullptr)
- **Xml** (Xml *parent, const char *name, const char *attributes=nullptr)
- **Xml** (Xml *parent, const String &name, const char *attributes=nullptr)
- void clear ()

  clear xml
- bool create (const char *str, size_t size=0, bool owner=false)

  create xml
- bool **create** (const String &str, size_t size=0, bool owner=false)
- bool load (const char *name)

  load xml
- bool **load** (const String &name)
- bool **load** (Stream &stream)
- bool save (const char *name, bool compact=false) const

  save xml

- bool **save** (const String &name, bool compact=false) const
- bool **save** (Stream &stream, bool compact=false) const
- const Xml getRoot () const

  xml root
- Xml **getRoot** ()
- uint32_t setParent (Xml &parent, bool check=true)

  xml parent
- const Xml **getParent** () const
- Xml **getParent** ()
- Xml addChild (const char *name, bool check=true)

  xml children
- uint32_t **addChild** (Xml &child, bool check=true)
- bool **removeChild** (Xml &child)
- void **releaseChildren** ()
- uint32_t **findChild** (const char *name) const
- bool **isChild** (const char *name) const
- const Xml **getChild** (const char *name) const
- Xml **getChild** (const char *name)
- uint32_t **getNumChildren** () const
- const Array< Xml > **getChildren** () const
- Array< Xml > **getChildren** ()
- const Xml **getChild** (uint32_t index) const
- Xml **getChild** (uint32_t index)
- String getPathName () const

  xml path name
- void setName (const char *name)

  xml name
- void **setName** (const String &name)
- String **getName** () const
- void setData (bool value)
- void **setData** (const char *value)
- void **setData** (const String &value)
- void **setData** (int32_t value, uint32_t radix=10)
- void **setData** (uint32_t value, uint32_t radix=10)
- void **setData** (uint64_t value, uint32_t radix=10)
- void **setData** (float32_t value, uint32_t digits=6, bool compact=true, bool exponent=true)
- void **setData** (float64_t value, uint32_t digits=12, bool compact=true, bool exponent=true)
- template<class Type >
  Xml **setData** (const char *name, Type value)
- String **getData** () const
- bool **getDataBool** () const
- int32_t **getDatai32** (uint32_t radix=10) const
- uint32_t **getDatau32** (uint32_t radix=10) const
- uint64_t **getDatau64** (uint32_t radix=10) const
- float32_t **getDataf32** () const
- float64_t **getDataf64** () const
- String **getData** (const char *name, const String &value=String::null) const
- bool **getData** (const char *name, bool value) const
- int32_t **getData** (const char *name, int32_t value, uint32_t radix=10) const
- uint32_t **getData** (const char *name, uint32_t value, uint32_t radix=10) const
- uint64_t **getData** (const char *name, uint64_t value, uint32_t radix=10) const
- float32_t **getData** (const char *name, float32_t value) const
- float64_t **getData** (const char *name, float64_t value) const
- void setData (const char **values, uint32_t size, uint32_t wrap=Maxu32)

- void **setData** (const String *values, uint32_t size, uint32_t wrap=Maxu32)
- void **setData** (const int32_t *values, uint32_t size, uint32_t radix=10, uint32_t wrap=Maxu32)
- void **setData** (const uint32_t *values, uint32_t size, uint32_t radix=10, uint32_t wrap=Maxu32)
- void **setData** (const float32_t *values, uint32_t size, uint32_t digits=6, bool compact=true, bool exponent=true, uint32_t wrap=Maxu32)
- void **setData** (const float64_t *values, uint32_t size, uint32_t digits=12, bool compact=true, bool exponent=true, uint32_t wrap=Maxu32)
- template<class Type >
  Xml **setData** (const char *name, Type *values, uint32_t size)
- template<class Type >
  void **setData** (const Array< Type > &values)
- template<class Type >
  void **setData** (const char *name, const Array< Type > &values)
- uint32_t **getData** (String *values, uint32_t size) const
- uint32_t **getData** (int32_t *values, uint32_t size, uint32_t radix=10) const
- uint32_t **getData** (uint32_t *values, uint32_t size, uint32_t radix=10) const
- uint32_t **getData** (float32_t *values, uint32_t size) const
- uint32_t **getData** (float64_t *values, uint32_t size) const
- template<class Type >
  uint32_t **getData** (const char *name, Type *values, uint32_t size) const
- template<class Type >
  uint32_t **getData** (Array< Type > &values) const
- template<class Type >
  uint32_t **getData** (const char *name, Array< Type > &values) const
- uint32_t addAttribute (const char *name)

  xml attributes
- bool **removeAttribute** (const char *name)
- uint32_t **findAttribute** (const char *name) const
- bool **isAttribute** (const char *name) const
- void **removeAttributes** ()
- uint32_t **getNumAttributes** () const
- String **getAttributeName** (uint32_t index) const
- bool **setAttributes** (const char *str)
- void **setAttribute** (uint32_t index, bool value)
- void **setAttribute** (uint32_t index, const char *value)
- void **setAttribute** (uint32_t index, const String &value)
- void **setAttribute** (uint32_t index, int32_t value, uint32_t radix=10)
- void **setAttribute** (uint32_t index, uint32_t value, uint32_t radix=10)
- void **setAttribute** (uint32_t index, uint64_t value, uint32_t radix=10)
- void **setAttribute** (uint32_t index, float32_t value, uint32_t digits=6, bool compact=true, bool exponent=true)
- void **setAttribute** (uint32_t index, float64_t value, uint32_t digits=12, bool compact=true, bool exponent=true)
- template<class Type >
  uint32_t **setAttribute** (const char *name, Type value)
- String **getAttribute** (uint32_t index) const
- int32_t **getAttributei32** (uint32_t index, uint32_t radix=10) const
- uint32_t **getAttributeu32** (uint32_t index, uint32_t radix=10) const
- uint64_t **getAttributeu64** (uint32_t index, uint32_t radix=10) const
- float32_t **getAttributef32** (uint32_t index) const
- float64_t **getAttributef64** (uint32_t index) const
- String **getAttribute** (const char *name, const String &value=String::null) const
- bool **getAttribute** (const char *name, bool value) const
- int32_t **getAttribute** (const char *name, int32_t value, uint32_t radix=10) const
- uint32_t **getAttribute** (const char *name, uint32_t value, uint32_t radix=10) const
- uint64_t **getAttribute** (const char *name, uint64_t value, uint32_t radix=10) const
- float32_t **getAttribute** (const char *name, float32_t value) const

- float64_t **getAttribute** (const char *name, float64_t value) const
- void setAttribute (uint32_t index, const char **values, uint32_t size, const char *delimiter=nullptr)

  xml array attributes
- void **setAttribute** (uint32_t index, const String *values, uint32_t size, const char *delimiter=nullptr)
- void **setAttribute** (uint32_t index, const int32_t *values, uint32_t size, uint32_t radix=10)
- void **setAttribute** (uint32_t index, const uint32_t *values, uint32_t size, uint32_t radix=10)
- void **setAttribute** (uint32_t index, const float32_t *values, uint32_t size, uint32_t digits=6, bool compact=true, bool exponent=true)
- void **setAttribute** (uint32_t index, const float64_t *values, uint32_t size, uint32_t digits=12, bool compact=true, bool exponent=true)
- template<class Type >
  uint32_t **setAttribute** (const char *name, Type *values, uint32_t size)
- template<class Type >
  void **setAttribute** (uint32_t index, const Array< Type > &values)
- template<class Type >
  void **setAttribute** (const char *name, const Array< Type > &values)
- uint32_t **getAttribute** (uint32_t index, String *values, uint32_t size, const char *delimiter=nullptr) const
- uint32_t **getAttribute** (uint32_t index, int32_t *values, uint32_t size, uint32_t radix=10) const
- uint32_t **getAttribute** (uint32_t index, uint32_t *values, uint32_t size, uint32_t radix=10) const
- uint32_t **getAttribute** (uint32_t index, float32_t *values, uint32_t size) const
- uint32_t **getAttribute** (uint32_t index, float64_t *values, uint32_t size) const
- template<class Type >
  uint32_t **getAttribute** (const char *name, Type *values, uint32_t size) const
- template<class Type >
  uint32_t **getAttribute** (uint32_t index, Array< Type > &values) const
- template<class Type >
  uint32_t **getAttribute** (const char *name, Array< Type > &values) const

5.298.1 Detailed Description

The Xml class is a comprehensive utility for managing Xml data in a flexible and efficient manner. It provides an object-oriented interface for working with Xml documents and supports a wide range of features for creating, manipulating, and querying Xml data.

5.298.2 Member Function Documentation

5.298.2.1 setData() [1/2]

```
void Tellusim::Xml::setData (
            bool value )
```

xml data

Parameters

radix	The decimal number radix (use 16 for hexadecimal numbers).
digits	The number of digits in the floating-point representation.
compact	Remove redundant zeros at the end of the number.
exponent	Use exponent representation.

5.298.2.2 setData() [2/2]

```
void Tellusim::Xml::setData (
             const char ** values,
             uint32_t size,
             uint32_t wrap = Maxu32 )
```

xml array data

Parameters

wrap	The data string wrap parameter to avoid extra-long lines.
radix	The decimal number radix (use 16 for hexadecimal numbers).
digits	The number of digits in the floating-point representation.
compact	Remove redundant zeros at the end of the number.
exponent	Use exponent representation.

Index