ParCompMark Reference Manual

v0.2

IT^2 ParCompMark Dev. Team

2006

Contents

Chapter 1

ParCompMark(p. 5) API Reference

This is the complete API reference for **ParCompMark**(p. 5); contained within are the specifications for each class and the methods on those classes which you can refer to when writing code which uses **ParCompMark**(p. 5).

Chapter 2

ParCompMark Class Index

2.1 ParCompMark Class List

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

Chapter 3

ParCompMark Namespace Documentation

3.1 ParCompMark Namespace Reference

3.1.1 Detailed Description

This namespace contains the classes of project **ParCompMark**(p. 5). The source files starts with `PCM` prefix. There is a unit test for this project called **ParCompMarkTest**(p. 8).

Classes

- class **Application**
- class **Buffer**
- class **Cluster**
- class **Container**
- class **Context**
- class **DummyLock**
- class **DynLoad**
- class **Exception**
- class **GLXGLContext**
- class **GLXRenderWindow**
- class **HandleClient**
- class **Host**
- class **HostInfo**
- class **Lock**
- class **Logger**
- class **Mutex**
- class **Name**
- class **Network**
- class **Node**
- class **OldContainer**
- class **OpenGLRenderingEngine**
- class **OutputNode**
- class **Pointer**

- class **Process**
- class **Singleton**
- class **SqVM**
- class **Thread**
- class **Timer**
- class **XDisplay**

Functions

- void **squirrelClassBindings** ()

3.1.2 Typedef Documentation

3.1.2.1 typedef double Real

Unsigned floating type.

3.1.2.2 typedef __s16 s16

Signed 16-bit type.

3.1.2.3 typedef __s32 s32

Signed 32-bit type.

3.1.2.4 typedef __s64 s64

Signed 64-bit type.

3.1.2.5 typedef __s8 s8

Signed 8-bit type.

3.1.2.6 typedef __u16 u16

Unsigned 16-bit type.

3.1.2.7 typedef __u32 u32

Unsigned 32-bit type.

3.1.2.8 typedef __u64 u64

Unsigned 64-bit type.

3.1.2.9 typedef __u8 u8

Unsigned 8-bit type.

3.1.3 Function Documentation

3.1.3.1 void ParCompMark::squirrelClassBindings ()

Call static squirrelGlue methods of the binded classes.

3.2 ParCompMarkTest Namespace Reference

3.2.1 Detailed Description

This namespace contains the classes of project **ParCompMarkTest**(p. 8). The source files starts with `Test` prefix. This is a unit test project for project **ParCompMark**(p. 5).

Chapter 4

ParCompMark Class Documentation

4.1 Application Class Reference

Inherits **Singleton**< **Application** >.

4.1.1 Detailed Description

This singleton class handles the application initializing, command line parsing, starting tasks etc.

Getters & setters

- const bool & **getInitialized** () const
- **OutputNode::Pointer** & **getOutputDocument** ()
- static const std::string & **getUsageString** ()
- static const bool & **getCommanderMode** ()
- static const bool & **getGUIMode** ()
- static const bool & **getManualClusterDescription** ()
- static const std::string & **getClusterDescription** ()
- static const bool & **getLowLevelMode** ()
- static const bool & **getInteractiveParameters** ()
- static const std::string & **getParameters** ()
- static const std::string & **getInput** ()
- static const std::string & **getOutput** ()

Methods

- virtual void **setupHandlers** () const
- virtual void **initalize** ()
- virtual void **finalize** ()
- virtual bool **startOperation** ()
- virtual void **NetworkTest** ()
- virtual void **writeOutput** ()
- virtual bool **commanderOperation** ()
- virtual bool **soldierOperation** ()

Public Member Functions

Constructors & destructor

- **Application** ()
- virtual ~**Application** ()

Static Public Member Functions

Class methods

- static void **parseCommandLine** (const **u32** &argc, const char **&argv)
- static void **showHelp** (const std::string &strarg)
- static void **showVersion** (const std::string &strarg)
- static void **setCommanderOn** (const std::string &strarg)
- static void **setGUIOn** (const std::string &strarg)
- static void **setLowLevelOn** (const std::string &strarg)
- static void **setCluster** (const std::string &strarg)
- static void **setParameters** (const std::string &strarg)
- static void **setInput** (const std::string &strarg)
- static void **setOutput** (const std::string &strarg)
- static void **terminateHandler** ()
- static void **unexpectedHandler** ()
- static void **segfaultHandler** (int value)
- static void **interruptHandler** (int value)

Protected Attributes

Variables

- **Logger** * **mLogger**
- bool **mInitialized**
- **OutputNode::Pointer mOutputDocument**

Static Protected Attributes

Class constants

- static const **Application::CommandLineOption mCommandLineOptions** [ ]
- static const **u32 mCommandLineOptionCount**
- static const std::string **mUsageString**

Class variables

- static bool **mCommanderMode**
- static bool **mGUIMode**
- static bool **mManualClusterDescription**
- static std::string **mClusterDescription**
- static bool **mLowLevelMode**
- static bool **mInteractiveParameters**
- static std::string **mParameters**
- static std::string **mInput**
- static std::string **mOutput**

Classes

- struct **CommandLineOption**

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Application ()

Default constructor.

4.1.2.2 virtual ~Application () [virtual]

The destructor. This class has virtual destructor.

4.1.3 Member Function Documentation

4.1.3.1 virtual bool commanderOperation () [protected, virtual]

Operation of a commander mode application.

Returns:
Return true on error.

4.1.3.2 virtual void finalize () [virtual]

Finalize the PCM application.

4.1.3.3 const std::string & getClusterDescription () [inline, static]

Getter of mClusterDescription. Returns value of mClusterDescription.

Returns:
The value of mClusterDescription

4.1.3.4 const bool & getCommanderMode () [inline, static]

Getter of mCommanderMode. Returns value of mCommanderMode.

Returns:
The value of mCommanderMode

4.1.3.5 const bool & getGUIMode () [inline, static]

Getter of mGUIMode. Returns value of mGUIMode.

Returns:
The value of mGUIMode

4.1.3.6 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:
The value of mInitialized

4.1.3.7 const std::string & getInput () [inline, static]

Getter of mInput. Returns value of mInput.

Returns:
The value of mInput

4.1.3.8 const bool & getInteractiveParameters () [inline, static]

Getter of mInteractiveParameters. Returns value of mInteractiveParameters.

Returns:
The value of mInteractiveParameters

4.1.3.9 const bool & getLowLevelMode () [inline, static]

Getter of mLowLevelMode. Returns value of mLowLevelMode.

Returns:
The value of mLowLevelMode

4.1.3.10 const bool & getManualClusterDescription () [inline, static]

Getter of mManualClusterDescription. Returns value of mManualClusterDescription.

Returns:
The value of mManualClusterDescription

4.1.3.11 const std::string & getOutput () [inline, static]

Getter of mOutput. Returns value of mOutput.

Returns:
The value of mOutput

4.1.3.12 OutputNode::Pointer & getOutputDocument () [inline]

Getter of mOutputDocument. Returns value of mOutputDocument.

Returns:
The value of mOutputDocument

4.1.3.13 const std::string & getParameters () [inline, static]

Getter of mParameters. Returns value of mParameters.

Returns:
The value of mParameters

4.1.3.14 const std::string & getUsageString () [inline, static]

Getter of mUsageString. Returns value of mUsageString.

Returns:
The value of mUsageString

4.1.3.15 virtual void initalize () [virtual]

Initialize the PCM application.

4.1.3.16 static void interruptHandler (int *value*) [static]

Interrupt signal handler.

Parameters:
← ***value*** Signal parameter.

4.1.3.17 virtual void NetworkTest () [virtual]

Network(p. 85) testing method.

4.1.3.18 static void parseCommandLine (const u32 & *argc*, const char **& *argv*) [static]

Parse ANSI C command line.

Parameters:
← ***argc*** Number of command line arguments.

← ***argv*** Array of command line arguments.

4.1.3.19 static void segfaultHandler (int *value*) [static]

Segmentation fault handler.

Parameters:
← ***value*** Signal parameter.

4.1.3.20 static void setCluster (const std::string & *strarg*) [static]

Set cluster description file.

Parameters:
← ***strarg*** String argument.

4.1.3.21 static void setCommanderOn (const std::string & *strarg*) [static]

Set commander mode.

Parameters:
← ***strarg*** String argument (dummy).

4.1.3.22 static void setGUIOn (const std::string & *strarg*) [static]

Set GUI mode.

Parameters:
← ***strarg*** String argument (dummy).

4.1.3.23 static void setInput (const std::string & *strarg*) [static]

Set input file.

Parameters:
← ***strarg*** String argument.

4.1.3.24 static void setLowLevelOn (const std::string & *strarg*) [static]

Set low-level scripting mode.

Parameters:
← ***strarg*** String argument (dummy).

4.1.3.25 static void setOutput (const std::string & *strarg*) [static]

Set output file.

Parameters:
← ***strarg*** String argument.

4.1.3.26 static void setParameters (const std::string & *strarg*) [static]

Set parameters description file.

Parameters:
← ***strarg*** String argument.

4.1.3.27 virtual void setupHandlers () const [virtual]

Setup special event handlers.

4.1.3.28 static void showHelp (const std::string & *strarg*) [static]

Write help to std out.

Parameters:
← ***strarg*** String argument (dummy).

4.1.3.29 static void showVersion (const std::string & *strarg*) [static]

Write version to std out.

Parameters:
← ***strarg*** String argument (dummy).

4.1.3.30 virtual bool soldierOperation () [protected, virtual]

Operation of a soldier mode (not commander mode) application.

Returns:
Return true on error.

4.1.3.31 virtual bool startOperation () [virtual]

The application starts its operation. The operation depends on the commander mode flag.

Returns:
Return true on error.

4.1.3.32 static void terminateHandler () [static]

Abnormal termination handler.

4.1.3.33 static void unexpectedHandler () [static]

Unexpected exception handler.

4.1.3.34 virtual void writeOutput () [virtual]

Write collected output.

4.1.4 Member Data Documentation

4.1.4.1 std::string mClusterDescription [static, protected]

Cluster(p. 25) description file name.

Remarks:
This is own attribute of this class.

4.1.4.2 bool mCommanderMode [static, protected]

Indicates commander mode (default false).

Remarks:
This is own attribute of this class.

4.1.4.3 const u32 mCommandLineOptionCount [static, protected]

Number of command line options.

Remarks:
This is own attribute of this class.

4.1.4.4 const Application::CommandLineOption mCommandLineOptions[ ] [static, protected]

Command line options for **ParCompMark**(p. 5).

Remarks:
This is own attribute of this class.

4.1.4.5 bool mGUIMode [static, protected]

Indicates GUI mode (default false).

Remarks:
This is own attribute of this class.

4.1.4.6 bool mInitialized [protected]

The application is initialized.

Remarks:
This is own attribute of this class.

4.1.4.7 std::string mInput [static, protected]

Input script file name.

Remarks:
This is own attribute of this class.

4.1.4.8 bool mInteractiveParameters [static, protected]

Indicates interactive parameter settings (default true).

Remarks:
This is own attribute of this class.

4.1.4.9 Logger* mLogger [protected]

Logger(p. 76) object.

Remarks:
This attribute references an attribute.

4.1.4.10 bool mLowLevelMode [static, protected]

Indicates low-level scripting mode (default false).

Remarks:
This is own attribute of this class.

4.1.4.11 bool mManualClusterDescription [static, protected]

Indicates manual cluster description (default false).

Remarks:
This is own attribute of this class.

4.1.4.12 std::string mOutput [static, protected]

Output file name.

Remarks:
This is own attribute of this class.

4.1.4.13 OutputNode::Pointer mOutputDocument [protected]

Root of the output document.

Remarks:
This is own attribute of this class.

4.1.4.14 std::string mParameters [static, protected]

Parameters description file name.

Remarks:
This is own attribute of this class.

4.1.4.15 const std::string mUsageString [static, protected]

Application(p. 9) usage string.

Remarks:
This is own attribute of this class.

4.2 Application::CommandLineOption Struct Reference

4.2.1 Detailed Description

Structure describing a command line option for command line parsing.

Public Attributes

- **s8 shortName**
- std::string **longName**
- std::string **description**
- bool **hasArgument**
- void(∗ **handler**)(const std::string &)

4.2.2 Member Data Documentation

4.2.2.1 std::string description

Description

4.2.2.2 void(∗ handler)(const std::string &)

Argument handler method

4.2.2.3 bool hasArgument

The option has an argument

4.2.2.4 std::string longName

Long name

4.2.2.5 s8 shortName

Short name

4.3 Buffer Class Reference

4.3.1 Detailed Description

Memory buffer class.

Methods

- virtual void **init** (const int &left, const int &top, const int &width, const int &height, const int &depthFormat)
- virtual void **freeBuffers** ()

Public Types

- typedef **Pointer**< **Buffer**, **Mutex** > **Pointer**

Public Member Functions

Constructors & destructor

- **Buffer** ()
- virtual ~**Buffer** ()

Getters & setters

- const PCuint & **getLeft** () const
- const PCuint & **getTop** () const
- const PCuint & **getWidth** () const
- const PCuint & **getHeight** () const
- const bool & **getOwnPointers** () const
- PCuint * **getColour** ()
- void **setColour** (PCuint *colour)
- void * **getDepth** ()
- void **setDepth** (void *depth)
- const PCint & **getDepthFormat** () const
- const PCint & **getOutputRowPixel** () const

Protected Attributes

Variables

- PCuint **mLeft**
- PCuint **mTop**
- PCuint **mWidth**
- PCuint **mHeight**
- bool **mOwnPointers**
- PCuint * **mColour**
- void * **mDepth**
- PCint **mDepthFormat**
- PCint **mOutputRowPixel**

4.3.2 Member Typedef Documentation

4.3.2.1 typedef Pointer< Buffer, Mutex > Pointer

Type for pointer on this class.

4.3.3 Constructor & Destructor Documentation

4.3.3.1 Buffer ()

Default constructor.

4.3.3.2 ~Buffer () [virtual]

The destructor. This class has virtual destructor.

4.3.4 Member Function Documentation

4.3.4.1 void freeBuffers () [protected, virtual]

Free mColour and mDepth buffers.

4.3.4.2 PCuint * getColour () [inline]

Getter of mColour. Returns value of mColour.

Returns:
The value of mColour

4.3.4.3 void * getDepth () [inline]

Getter of mDepth. Returns value of mDepth.

Returns:
The value of mDepth

4.3.4.4 const PCint & getDepthFormat () const [inline]

Getter of mDepthFormat. Returns value of mDepthFormat.

Returns:
The value of mDepthFormat

4.3.4.5 const PCuint & getHeight () const [inline]

Getter of mHeight. Returns value of mHeight.

Returns:
The value of mHeight

4.3.4.6 const PCuint & getLeft () const [inline]

Getter of mLeft. Returns value of mLeft.

Returns:
The value of mLeft

4.3.4.7 const PCint & getOutputRowPixel () const [inline]

Getter of mOutputRowPixel. Returns value of mOutputRowPixel.

Returns:
The value of mOutputRowPixel

4.3.4.8 const bool & getOwnPointers () const [inline]

Getter of mOwnPointers. Returns value of mOwnPointers.

Returns:
The value of mOwnPointers

4.3.4.9 const PCuint & getTop () const [inline]

Getter of mTop. Returns value of mTop.

Returns:
The value of mTop

4.3.4.10 const PCuint & getWidth () const [inline]

Getter of mWidth. Returns value of mWidth.

Returns:
The value of mWidth

4.3.4.11 void init (const int & *left*, const int & *top*, const int & *width*, const int & *height*, const int & *depthFormat*) [virtual]

Init the buffer.

Parameters:
- ← ***left***
- ← ***top***
- ← ***width***
- ← ***height***
- ← ***depthFormat***

4.3.4.12 void setColour (PCuint * *colour*) [inline]

Setter of mColour. Sets value of mColour.

Parameters:
- ← ***colour*** The value of mColour

4.3.4.13 void setDepth (void * *depth*) [inline]

Setter of mDepth. Sets value of mDepth.

Parameters:
- ← ***depth*** The value of mDepth

4.3.5 Member Data Documentation

4.3.5.1 PCuint* mColour [protected]

Colour information buffer.

Remarks:
This is own attribute of this class.

4.3.5.2 void* mDepth [protected]

Depth information buffer.

Remarks:
This is own attribute of this class.

4.3.5.3 PCint mDepthFormat [protected]

Depth format for deleting.

Remarks:
This is own attribute of this class.

4.3.5.4 PCuint mHeight [protected]

Remarks:
This is own attribute of this class.

4.3.5.5 PCuint mLeft [protected]

Remarks:
This is own attribute of this class.

4.3.5.6 PCint mOutputRowPixel [protected]

If the output is not the whole frame, this is the remainde pixels in a row.

Remarks:
This is own attribute of this class.

4.3.5.7 bool mOwnPointers [protected]

The **Buffer**(p. 20) is responsible for deallocation of mColour and mDepth.

Remarks:
This is own attribute of this class.

4.3.5.8 PCuint mTop [protected]

Remarks:
This is own attribute of this class.

4.3.5.9 PCuint mWidth [protected]

Remarks:
This is own attribute of this class.

4.4 Cluster Class Reference

Inherits **Singleton**< **Cluster** >.

4.4.1 Detailed Description

Class contain hosts. Description of the physical grid.

Public Member Functions

Constructors & destructor

- **Cluster** ()
- virtual ~**Cluster** ()

Getters & setters

- **OldContainer::Pointer** & **getHosts** ()

Protected Attributes

Variables

- **OldContainer::Pointer mHosts**

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Cluster ()

Default constructor.

4.4.2.2 ~Cluster () [virtual]

The destructor. This class has virtual destructor.

4.4.3 Member Function Documentation

4.4.3.1 OldContainer::Pointer & getHosts () [inline]

Getter of mHosts. Returns value of mHosts.

Returns:
The value of mHosts

4.4.4 Member Data Documentation

4.4.4.1 OldContainer::Pointer mHosts [protected]

Host(p. 69) map.

Remarks:
This is own attribute of this class.

4.5 Container Class Template Reference

4.5.1 Detailed Description

template<class ElementType, class LockType> class ParCompMark::Container< ElementType, LockType >

String addressed map of typed smart pointers.

Public Types

- typedef **Pointer**< **Container**< ElementType, LockType >, LockType > **Pointer**
- typedef ElementType::Pointer **ElementPointer**
- typedef std::map< std::string, **ElementPointer** > **ElementsMap**
- typedef ElementsMap::iterator **Iterator**

Public Member Functions

Constructors & destructor

- **Container** ()
- virtual ~**Container** ()

Methods

- virtual void **add** (std::string name, **ElementPointer** element)
- virtual **ElementPointer get** (const std::string &name)
- virtual bool **has** (const std::string &name)
- virtual void **remove** (const std::string &name)
- virtual **u32 getSize** ()
- virtual std::string * **getList** ()
- virtual bool **isEmpty** ()

Protected Attributes

Variables

- **ElementsMap mElements**

4.5.2 Member Typedef Documentation

4.5.2.1 typedef ElementType::Pointer ElementPointer

Type definition for pointer on elements.

4.5.2.2 typedef std::map< std::string, ElementPointer > ElementsMap

Type definition for map of elements.

4.5.2.3 typedef ElementsMap::iterator Iterator

Type definition for iterator on map of elements.

4.5.2.4 typedef Pointer< Container < ElementType, LockType >, LockType > Pointer

Type for pointer on this class.

4.5.3 Constructor & Destructor Documentation

4.5.3.1 Container () [inline]

Default constructor.

4.5.3.2 ~Container () [inline, virtual]

The destructor. This class has virtual destructor.

4.5.4 Member Function Documentation

4.5.4.1 void add (std::string *name*, ElementPointer *element*) [virtual]

Add an element.

Parameters:

← ***name*** **Name**(p. 83) of the element

← ***element*** Element

4.5.4.2 ElementType::Pointer get (const std::string & *name*) [virtual]

Get an element by name.

Parameters:

← ***name*** **Name**(p. 83) of the element

Returns:

Pointer(p. 107) to the element

4.5.4.3 std::string * getList () [virtual]

Return name list of elements.

Returns:

Name(p. 83) list.

4.5.4.4 u32 getSize () [virtual]

Return the number of elements.

Returns:
Pointer(p. 107) to the element

4.5.4.5 bool has (const std::string & *name*) [virtual]

Search for an element by name.

Parameters:
← ***name*** **Name**(p. 83) of the element

Returns:
The container has the element.

4.5.4.6 bool isEmpty () [virtual]

Return true if the container is empty.

Returns:
True if the container is empty

4.5.4.7 void remove (const std::string & *name*) [virtual]

Remove an element by name.

Parameters:
← ***name*** **Name**(p. 83) of the element

4.5.5 Member Data Documentation

4.5.5.1 ElementsMap mElements [protected]

Map of elements.

Remarks:
This is own attribute of this class.

4.6 Context Class Reference

4.6.1 Detailed Description

Class containing PC context information.

Public Types

- typedef **Pointer**< **Context**, **DummyLock** > **Pointer**
- enum **ContextType** { **LOCAL**, **GLOBAL** }

Public Member Functions

Constructors & destructor

- **Context** (**Process** *parent)
- virtual ~**Context** ()

Getters & setters

- const bool & **getUseGL** () const
- void **setUseGL** (const bool &usegl)
- const **Context::ContextType** & **getContextType** () const
- void **setContextType** (const **Context::ContextType** &contexttype)
- const PCint & **getFrameWidth** () const
- void **setFrameWidth** (const PCint &framewidth)
- const PCint & **getFrameHeight** () const
- void **setFrameHeight** (const PCint &frameheight)
- const PCint & **getColourFormat** () const
- void **setColourFormat** (const PCint &colourformat)
- const PCint & **getDepthFormat** () const
- void **setDepthFormat** (const PCint &depthformat)
- const PCint & **getPixelFormat** () const
- const PCint & **getCompositeType** () const
- void **setCompositeType** (const PCint &compositetype)
- const PCint & **getCompressionHint** () const
- void **setCompressionHint** (const PCint &compressionhint)
- const PCint & **getRetainBuffers** () const
- void **setRetainBuffers** (const PCint &retainbuffers)
- const PCint & **getOutputDepth** () const
- void **setOutputDepth** (const PCint &outputdepth)
- char ** **getNodes** () const
- const PCint & **getNodeNumber** () const
- const int & **getNodeIndex** () const
- void **setNodeIndex** (const int &nodeindex)
- const PCint & **getHostIndex** () const
- const PCint & **getNetworkID** () const
- void **setNetworkID** (const PCint &networkid)
- const PCcontext & **getContext** () const
- **Process** * **getParent** ()

Methods

- virtual void **setNodes** (const std::string &nodes)
- virtual void **init** ()
- virtual void **finalize** ()

Protected Attributes

Variables

- bool **mUseGL**
- **ContextType mContextType**
- PCint **mFrameWidth**
- PCint **mFrameHeight**
- PCint **mColourFormat**
- PCint **mDepthFormat**
- PCint **mPixelFormat**
- PCint **mCompositeType**
- PCint **mCompressionHint**
- PCint **mRetainBuffers**
- PCint **mOutputDepth**
- char ∗∗ **mNodes**
- PCint **mNodeNumber**
- int **mNodeIndex**
- PCint **mHostIndex**
- PCint **mNetworkID**
- PCcontext **mContext**
- **Process** ∗ **mParent**

4.6.2 Member Typedef Documentation

4.6.2.1 typedef Pointer< Context, DummyLock > Pointer

Type for pointer on this class.

4.6.3 Member Enumeration Documentation

4.6.3.1 enum ContextType

The control type of PC context (local, global).

Enumerator:

LOCAL Local context.

GLOBAL Global context.

4.6.4 Constructor & Destructor Documentation

4.6.4.1 Context (Process ∗ *parent*)

Create **Context**(p. 30). Normally **Process**(p. 115) calls this constructor.

Parameters:

← ***parent*** Parent host

4.6.4.2 ∼Context () [virtual]

The destructor. This class has virtual destructor.

4.6.5 Member Function Documentation

4.6.5.1 void finalize () [virtual]

Finalize the PC context.

4.6.5.2 const PCint & getColourFormat () const [inline]

Getter of mColourFormat. Returns value of mColourFormat.

Returns:
The value of mColourFormat

4.6.5.3 const PCint & getCompositeType () const [inline]

Getter of mCompositeType. Returns value of mCompositeType.

Returns:
The value of mCompositeType

4.6.5.4 const PCint & getCompressionHint () const [inline]

Getter of mCompressionHint. Returns value of mCompressionHint.

Returns:
The value of mCompressionHint

4.6.5.5 const PCcontext & getContext () const [inline]

Getter of mContext. Returns value of mContext.

Returns:
The value of mContext

4.6.5.6 const Context::ContextType & getContextType () const [inline]

Getter of mContextType. Returns value of mContextType.

Returns:
The value of mContextType

4.6.5.7 const PCint & getDepthFormat () const [inline]

Getter of mDepthFormat. Returns value of mDepthFormat.

Returns:
The value of mDepthFormat

4.6.5.8 const PCint & getFrameHeight () const [inline]

Getter of mFrameHeight. Returns value of mFrameHeight.

Returns:
The value of mFrameHeight

4.6.5.9 const PCint & getFrameWidth () const [inline]

Getter of mFrameWidth. Returns value of mFrameWidth.

Returns:
The value of mFrameWidth

4.6.5.10 const PCint & getHostIndex () const [inline]

Getter of mHostIndex. Returns value of mHostIndex.

Returns:
The value of mHostIndex

4.6.5.11 const PCint & getNetworkID () const [inline]

Getter of mNetworkID. Returns value of mNetworkID.

Returns:
The value of mNetworkID

4.6.5.12 const int & getNodeIndex () const [inline]

Getter of mNodeIndex. Returns value of mNodeIndex.

Returns:
The value of mNodeIndex

4.6.5.13 const PCint & getNodeNumber () const [inline]

Getter of mNodeNumber. Returns value of mNodeNumber.

Returns:
The value of mNodeNumber

4.6.5.14 char ** getNodes () const [inline]

Getter of mNodes. Returns value of mNodes.

Returns:
The value of mNodes

4.6.5.15 const PCint & getOutputDepth () const [inline]

Getter of mOutputDepth. Returns value of mOutputDepth.

Returns:
The value of mOutputDepth

4.6.5.16 Process * getParent () [inline]

Getter of mParent. Returns value of mParent.

Returns:
The value of mParent

4.6.5.17 const PCint & getPixelFormat () const [inline]

Getter of mPixelFormat. Returns value of mPixelFormat.

Returns:
The value of mPixelFormat

4.6.5.18 const PCint & getRetainBuffers () const [inline]

Getter of mRetainBuffers. Returns value of mRetainBuffers.

Returns:
The value of mRetainBuffers

4.6.5.19 const bool & getUseGL () const [inline]

Getter of mUseGL. Returns value of mUseGL.

Returns:
The value of mUseGL

4.6.5.20 void init () [virtual]

Init the PC context.

4.6.5.21 void setColourFormat (const PCint & *colourformat*) [inline]

Setter of mColourFormat. Sets value of mColourFormat.

Parameters:
← ***colourformat*** The value of mColourFormat

4.6.5.22 void setCompositeType (const PCint & *compositetype*) [inline]

Setter of mCompositeType. Sets value of mCompositeType.

Parameters:
← ***compositetype*** The value of mCompositeType

4.6.5.23 void setCompressionHint (const PCint & *compressionhint*) [inline]

Setter of mCompressionHint. Sets value of mCompressionHint.

Parameters:
← ***compressionhint*** The value of mCompressionHint

4.6.5.24 void setContextType (const Context::ContextType & *contexttype*) [inline]

Setter of mContextType. Sets value of mContextType.

Parameters:
← ***contexttype*** The value of mContextType

4.6.5.25 void setDepthFormat (const PCint & *depthformat*) [inline]

Setter of mDepthFormat. Sets value of mDepthFormat.

Parameters:
← ***depthformat*** The value of mDepthFormat

4.6.5.26 void setFrameHeight (const PCint & *frameheight*) [inline]

Setter of mFrameHeight. Sets value of mFrameHeight.

Parameters:
← ***frameheight*** The value of mFrameHeight

4.6.5.27 void setFrameWidth (const PCint & *framewidth*) [inline]

Setter of mFrameWidth. Sets value of mFrameWidth.

Parameters:
← ***framewidth*** The value of mFrameWidth

4.6.5.28 void setNetworkID (const PCint & *networkid*) [inline]

Setter of mNetworkID. Sets value of mNetworkID.

Parameters:
← ***networkid*** The value of mNetworkID

4.6.5.29 void setNodeIndex (const int & *nodeindex*) [inline]

Setter of mNodeIndex. Sets value of mNodeIndex.

Parameters:
← ***nodeindex*** The value of mNodeIndex

4.6.5.30 void setNodes (const std::string & *nodes*) [virtual]

Create nodes char struct from string.

Parameters:
← ***nodes*** Nodes description.

4.6.5.31 void setOutputDepth (const PCint & *outputdepth*) [inline]

Setter of mOutputDepth. Sets value of mOutputDepth.

Parameters:
← ***outputdepth*** The value of mOutputDepth

4.6.5.32 void setRetainBuffers (const PCint & *retainbuffers*) [inline]

Setter of mRetainBuffers. Sets value of mRetainBuffers.

Parameters:
← ***retainbuffers*** The value of mRetainBuffers

4.6.5.33 void setUseGL (const bool & *usegl*) [inline]

Setter of mUseGL. Sets value of mUseGL.

Parameters:
← ***usegl*** The value of mUseGL

4.6.6 Member Data Documentation

4.6.6.1 PCint mColourFormat [protected]

Colour format of rendered frame by the context.

Remarks:
This is own attribute of this class.

4.6.6.2 PCint mCompositeType [protected]

Type of composition.

Remarks:
This is own attribute of this class.

4.6.6.3 PCint mCompressionHint [protected]

Compression hint.

Remarks:
This is own attribute of this class.

4.6.6.4 PCcontext mContext [protected]

The PC context.

Remarks:
This is own attribute of this class.

4.6.6.5 ContextType mContextType [protected]

Type of the context.

Remarks:
This is own attribute of this class.

4.6.6.6 PCint mDepthFormat [protected]

Depth format of rendered frame by the context.

Remarks:
This is own attribute of this class.

4.6.6.7 PCint mFrameHeight [protected]

Height of rendered frame by the context.

Remarks:
This is own attribute of this class.

4.6.6.8 PCint mFrameWidth [protected]

Width of rendered frame by the context.

Remarks:
This is own attribute of this class.

4.6.6.9 PCint mHostIndex [protected]

The own host index.

Remarks:
This is own attribute of this class.

4.6.6.10 PCint mNetworkID [protected]

The network ID.

Remarks:
This is own attribute of this class.

4.6.6.11 int mNodeIndex [protected]

Our place in the mNodes list.

Remarks:
This is own attribute of this class.

4.6.6.12 PCint mNodeNumber [protected]

Our place in the mNodes list.

Remarks:
This is own attribute of this class.

4.6.6.13 char** mNodes [protected]

Nodes in the context.

Remarks:
This is own attribute of this class.

4.6.6.14 PCint mOutputDepth [protected]

Needs depth information of the composition.

Remarks:
This is own attribute of this class.

4.6.6.15 Process* mParent [protected]

Parent **Process**(p. 115) of this **Context**(p. 30).

Remarks:
This attribute references an attribute.

4.6.6.16 PCint mPixelFormat [protected]

The pixel format. Or link between depth and colour format.

Remarks:
This is own attribute of this class.

4.6.6.17 PCint mRetainBuffers [protected]

Needs retain buffers.

Remarks:
This is own attribute of this class.

4.6.6.18 bool mUseGL [protected]

Use graphics card video memory for composition (OpenGL).

Remarks:
This is own attribute of this class.

4.7 DummyLock Class Reference

Inherits **Lock**.

4.7.1 Detailed Description

Dummy lock implementing the **Lock**(p. 74) interface. Does not do anything. The **DummyLock**(p. 40) can always be locked.

Public Member Functions

Methods

- virtual void **lock** ()
- virtual bool **trylock** ()
- virtual void **unlock** ()

4.7.2 Member Function Documentation

4.7.2.1 void lock () [inline, virtual]

Lock(p. 74) the lock. Does not do anything.

Implements **Lock** (p. 74).

4.7.2.2 bool trylock () [inline, virtual]

Try locking the lock. Does not do anything. Always return true.

Returns:
True if the locking was successful. Always true.

Implements **Lock** (p. 75).

4.7.2.3 void unlock () [inline, virtual]

Unlock the lock. Does not do anything.

Implements **Lock** (p. 75).

4.8 DynLoad Class Reference

4.8.1 Detailed Description

Dynamic load library.

Methods

- virtual void * **getFunc** (const std::string &funcName) const
- virtual void **load** ()
- virtual void **unload** ()

Public Member Functions

Constructors & destructor

- **DynLoad** (const std::string &libName)
- virtual ~**DynLoad** ()

Getters & setters

- void * **getHandle** () const
- const std::string & **getLibraryName** () const

Protected Attributes

Variables

- void * **mHandle**
- std::string **mLibraryName**

4.8.2 Constructor & Destructor Documentation

4.8.2.1 DynLoad (const std::string & *libName*)

Create a Dynamic load class, and load the libName named library.

Parameters:
← ***libName*** **Name**(p. 83) of the loaded library

4.8.2.2 ~DynLoad () [virtual]

The destructor. This class has virtual destructor.

4.8.3 Member Function Documentation

4.8.3.1 void * getFunc (const std::string & *funcName*) const [virtual]

Get a function pointer from dynamic library by name(symbol).

Parameters:
← ***funcName*** **Name**(p. 83) of function

Returns:
Pointer(p. 107) to the function

4.8.3.2 void * getHandle () const [inline]

Getter of mHandle. Returns value of mHandle.

Returns:
The value of mHandle

4.8.3.3 const std::string & getLibraryName () const [inline]

Getter of mLibraryName. Returns value of mLibraryName.

Returns:
The value of mLibraryName

4.8.3.4 void load () [protected, virtual]

Load a library. Called by constructor.

4.8.3.5 void unload () [protected, virtual]

Load a library. Called by constructor.

4.8.4 Member Data Documentation

4.8.4.1 void* mHandle [protected]

The dynamic library handler.

Remarks:
This is own attribute of this class.

4.8.4.2 std::string mLibraryName [protected]

The dynamic library name.

Remarks:
This is own attribute of this class.

4.9 Exception Class Reference

4.9.1 Detailed Description

Provides information about an internal error.

Remarks:
An application using PCM that the exceptions are caught, so all PCM functions should occur within a try{} catch(PCM::Exception& e) {} block.

Getters & setters

- const **Exception::ExceptionType** & **getType** () const
- const std::string & **getDescription** () const
- const std::string & **getFileName** () const
- const std::string & **getFunctionName** () const
- const **u32** & **getLineNumber** () const
- static **Exception** ∗ **getLastException** ()

Public Types

- enum **ExceptionType** {

  INTERNAL_ERROR, **NULL_POINTER_ERROR**, **INVALID_NAME_ERROR**, **INVALID_-ENUM_ERROR**,

  INVALID_VALUE_ERROR, **INVALID_OBJECT_ERROR**, **INVALID_CLASS_ERROR**, **INVALID_OPERATION_ERROR**,

  OPERATION_NOT_SUPPORTED_ERROR, **OUT_OF_MEMORY_ERROR**, **FILE_IO_-ERROR**, **FILE_FORMAT_ERROR**,

  USER_BREAK_ERROR, **MMAP_ERROR**, **IOCTL_ERROR**, **INVALID_DEVICE_ERROR**,

  SCRIPT_ERROR }

Public Member Functions

Constructors & destructor

- **Exception** (const **ExceptionType** &type=INTERNAL_ERROR, const std::string &description="unknown", const std::string &fileName="unknown", const std::string &functionName="unknown", const **u32** &lineNumber=0)

Static Public Member Functions

Class methods

- static std::string **translateType** (const **Exception::ExceptionType** &type)

Protected Attributes

Variables

- **ExceptionType mType**
- std::string **mDescription**
- std::string **mFileName**
- std::string **mFunctionName**
- **u32 mLineNumber**

Static Protected Attributes

Class variables

- static **Exception** ∗ **mLastException** = 0

4.9.2 Member Enumeration Documentation

4.9.2.1 enum ExceptionType

Definitions of error codes.

Enumerator:

INTERNAL_ERROR Unknown internal error (mostly occured by another library).

NULL_POINTER_ERROR Nullpointer error.

INVALID_NAME_ERROR Invalid name error.

INVALID_ENUM_ERROR Invalid enumerated value error.

INVALID_VALUE_ERROR Invalid value error.

INVALID_OBJECT_ERROR Invalid object error.

INVALID_CLASS_ERROR Invalid class error (not proper derived class).

INVALID_OPERATION_ERROR Invalid operation error.

OPERATION_NOT_SUPPORTED_ERROR Operation is not supported on this platform.

OUT_OF_MEMORY_ERROR Out of memory error.

FILE_IO_ERROR File I/O error.

FILE_FORMAT_ERROR Invalid file format error.

USER_BREAK_ERROR The user stopped the application.

MMAP_ERROR Memory mapping error.

IOCTL_ERROR I/O control error.

INVALID_DEVICE_ERROR Not a valid device.

SCRIPT_ERROR Error in a script file.

4.9.3 Constructor & Destructor Documentation

4.9.3.1 Exception (const ExceptionType & *type* = `INTERNAL_ERROR`, const std::string & *description* = `"unknown"`, const std::string & *fileName* = `"unknown"`, const std::string & *functionName* = `"unknown"`, const u32 & *lineNumber* = `0`) `[inline]`

Default constructor.

Parameters:

← ***type*** Type of exception.

← ***description*** Textual description of the exception.

← ***fileName*** **Name**(p. 83) of the file where the exception was thrown.

← ***functionName*** **Name**(p. 83) of the function where the exception was thrown.

← ***lineNumber*** Number of the line where the exception was thrown.

4.9.4 Member Function Documentation

4.9.4.1 const std::string & getDescription () const [inline]

Getter of mDescription. Returns value of mDescription.

Returns:
The value of mDescription

4.9.4.2 const std::string & getFileName () const [inline]

Getter of mFileName. Returns value of mFileName.

Returns:
The value of mFileName

4.9.4.3 const std::string & getFunctionName () const [inline]

Getter of mFunctionName. Returns value of mFunctionName.

Returns:
The value of mFunctionName

4.9.4.4 Exception * getLastException () [inline, static]

Getter of mLastException. Returns value of mLastException.

Returns:
The value of mLastException

4.9.4.5 const u32 & getLineNumber () const [inline]

Getter of mLineNumber. Returns value of mLineNumber.

Returns:
The value of mLineNumber

4.9.4.6 const Exception::ExceptionType & getType () const [inline]

Getter of mType. Returns value of mType.

Returns:
The value of mType

4.9.4.7 std::string translateType (const Exception::ExceptionType & *type*) [static]

Translate type to human readable format.

Parameters:
← ***type*** Enum exception value.

Returns:

4.9.5 Member Data Documentation

4.9.5.1 std::string mDescription [protected]

Textual description of the exception.

Remarks:
This is own attribute of this class.

4.9.5.2 std::string mFileName [protected]

Name(p. 83) of the file where the exception was thrown.

Remarks:
This is own attribute of this class.

4.9.5.3 std::string mFunctionName [protected]

Name(p. 83) of the function where the exception was thrown.

Remarks:
This is own attribute of this class.

4.9.5.4 Exception * mLastException = 0 [static, protected]

Pointer(p. 107) to the last raised exception.

Remarks:
This attribute references an attribute.

4.9.5.5 u32 mLineNumber [protected]

Number of the line where the exception was thrown.

Remarks:
This is own attribute of this class.

4.9.5.6 ExceptionType mType [protected]

Type of the exception.

Remarks:
This is own attribute of this class.

4.10 GLXGLContext Class Reference

4.10.1 Detailed Description

Class that encapsulates a GLX context. Original source can be found in Ogre3D sources (`http://ogre3d.org`).

Public Types

- typedef **Pointer**< **GLXGLContext**, **Mutex** > **Pointer**

Public Member Functions

Constructors & destructor

- **GLXGLContext** (**XDisplay::Pointer** &display, **GLXRenderWindow** *glxWindow,::XVisualInfo *visualInfo)
- virtual ~**GLXGLContext** ()

Getters & setters

- const bool & **getInitialized** () const
- const **XDisplay::Pointer** & **getDisplay** () const
- **GLXRenderWindow** * **getGLXWindow** () const
- ::XVisualInfo * **getVisualInfo** () const
- const ::GLXContext & **getGLXContext** () const

Methods

- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual void **setCurrent** ()

Protected Attributes

Variables

- bool **mInitialized**
- **XDisplay::Pointer mDisplay**
- **GLXRenderWindow** * **mGLXWindow**
- ::XVisualInfo * **mVisualInfo**
- ::GLXContext **mGLXContext**

4.10.2 Member Typedef Documentation

4.10.2.1 typedef Pointer< GLXGLContext, Mutex > Pointer

Type for pointer on this class.

4.10.3 Constructor & Destructor Documentation

4.10.3.1 GLXGLContext (XDisplay::Pointer & *display*, GLXRenderWindow * *glxWindow*, ::XVisualInfo * *visualInfo*)

Create GLX context.

Parameters:

→ ***display*** X display

← ***glxWindow*** Corresponding GLX rendering window.

← ***visualInfo*** Visualinfo for context creation

4.10.3.2 ~GLXGLContext () [virtual]

The destructor. This class has virtual destructor.

4.10.4 Member Function Documentation

4.10.4.1 void finalize () [virtual]

Finalize the GL context.

4.10.4.2 const XDisplay::Pointer & getDisplay () const [inline]

Getter of mDisplay. Returns value of mDisplay.

Returns:

The value of mDisplay

4.10.4.3 const ::GLXContext & getGLXContext () const [inline]

Getter of mGLXContext. Returns value of mGLXContext.

Returns:

The value of mGLXContext

4.10.4.4 GLXRenderWindow * getGLXWindow () const [inline]

Getter of mGLXWindow. Returns value of mGLXWindow.

Returns:

The value of mGLXWindow

4.10.4.5 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:

The value of mInitialized

4.10.4.6 inline::XVisualInfo * getVisualInfo () const

Getter of mVisualInfo. Returns value of mVisualInfo.

Returns:
The value of mVisualInfo

4.10.4.7 void initialize () [virtual]

Initialize the GL context.

4.10.4.8 void setCurrent () [virtual]

Enable the context. All subsequent rendering commands will go here.

4.10.5 Member Data Documentation

4.10.5.1 XDisplay::Pointer mDisplay [protected]

Corresponding X Display.

Remarks:
This is own attribute of this class.

4.10.5.2 ::GLXContext mGLXContext [protected]

Wrapped GLX context.

Remarks:
This is own attribute of this class.

4.10.5.3 GLXRenderWindow* mGLXWindow [protected]

Corresponding GLX rendering window.

Remarks:
This attribute references an attribute.

4.10.5.4 bool mInitialized [protected]

The context is initialized.

Remarks:
This is own attribute of this class.

4.10.5.5 ::XVisualInfo* mVisualInfo [protected]

Visual info for the context.

Remarks:
This is own attribute of this class.

4.11 GLXRenderWindow Class Reference

4.11.1 Detailed Description

Manages the target rendering window in GLX environment. Original source can be found in Ogre3D sources (`http://ogre3d.org`).

Methods

- virtual void **reposition** (const **s32** &left, const **s32** &top)
- virtual void **resize** (const **u32** &width, const **u32** &height)
- virtual void **startFrame** ()
- virtual void **finishFrame** ()
- virtual void **setCurrent** ()
- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual void **createWindow** ()
- virtual void **destroyWindow** ()
- virtual void **resetStatistics** ()
- virtual void **updateStatistics** ()
- virtual void **_reposition** ()
- virtual void **_resize** ()
- virtual void **_setCaption** ()

Public Types

- typedef **Pointer**< **GLXRenderWindow**, **Mutex** > **Pointer**

Public Member Functions

Constructors & destructor

- **GLXRenderWindow** (**XDisplay::Pointer** &display, **Process** *process, const std::string caption="PCM Framework", const bool &fullScreen=true, const **u32** &colourDepth=0, const **u32** &width=**GLXRenderWindow::MAXIMALSIZE**, const **u32** &height=**GLXRenderWindow::MAXIMALSIZE**, const **s32** &left=**GLXRenderWindow::CENTERED**, const **s32** &top=**GLXRenderWindow::CENTERED**, const **u32** &fsaaSamples=0)
- virtual ~**GLXRenderWindow** ()

Getters & setters

- **XDisplay::Pointer** & **getDisplay** ()
- **GLXGLContext::Pointer** & **getGLXGLContext** ()
- **Process** * **getProcess** () const
- const ::Window & **getWindow** () const
- const bool & **getInitialized** () const
- const bool & **getVisible** () const
- void **setVisible** (const bool &visible)
- const bool & **getFullScreen** () const
- void **setFullScreen** (const bool &fullscreen)
- const std::string & **getCaption** () const

- void **setCaption** (const std::string &caption)
- const **s32** & **getLeft** () const
- void **setLeft** (const **s32** &left)
- const **s32** & **getTop** () const
- void **setTop** (const **s32** &top)
- const **u32** & **getWidth** () const
- void **setWidth** (const **u32** &width)
- const **u32** & **getHeight** () const
- void **setHeight** (const **u32** &height)
- const **u32** & **getColourDepth** () const
- void **setColourDepth** (const **u32** &colourdepth)
- const **u32** & **getFSAASamples** () const
- void **setFSAASamples** (const **u32** &fsaasamples)
- const **u32** & **getFrameNumber** () const
- const **GLXRenderWindow::WindowStatistics** & **getWindowStatistics** () const

Static Public Attributes

Class constants

- static const **s32 CENTERED** = -1
- static const **u32 MAXIMALSIZE** = 0
- static const **Real UNDEFINEDSTATISTICS** = -1.0
- static const **s32 UNDEFINEDXRRCONFIGURATION** = -1

Protected Attributes

Variables

- **XDisplay::Pointer mDisplay**
- **GLXGLContext::Pointer mGLXGLContext**
- **Process** ∗ **mProcess**
- ::Window **mWindow**
- bool **mInitialized**
- bool **mVisible**
- bool **mFullScreen**
- std::string **mCaption**
- **s32 mLeft**
- **s32 mTop**
- **u32 mWidth**
- **u32 mHeight**
- **u32 mColourDepth**
- **u32 mFSAASamples**
- **s32 mOriginalXRRConfiguation**
- ::Atom **mAtomDeleteWindow**
- **u32 mFrameNumber**
- **WindowStatistics mWindowStatistics**
- **Real mFrameBeginTime**
- **Real mSumTriangleCount**
- **Real mSumFPS**

Classes

- struct **WindowStatistics**

4.11.2 Member Typedef Documentation

4.11.2.1 typedef Pointer< GLXRenderWindow, Mutex > Pointer

Type for pointer on this class.

4.11.3 Constructor & Destructor Documentation

4.11.3.1 GLXRenderWindow (XDisplay::Pointer & *display*, Process * *process*, const std::string *caption* = "PCM Framework", const bool & *fullScreen* = true, const u32 & *colourDepth* = 0, const u32 & *width* = GLXRenderWindow::MAXIMALSIZE, const u32 & *height* = GLXRenderWindow::MAXIMALSIZE, const s32 & *left* = GLXRenderWindow::CENTERED, const s32 & *top* = GLXRenderWindow::CENTERED, const u32 & *fsaaSamples* = 0)

Create GLX render window. Normally called by XDisplay::createRenderWindow.

Parameters:

→ ***display*** X display

← ***process*** Corresponding process.

← ***caption*** Window caption

← ***fullScreen*** The window appears in full screen mode

← ***colourDepth*** Colour depth of the window

← ***width*** Horizontal size

← ***height*** Vertical size

← ***left*** Horizontal position

← ***top*** Vertical position

← ***fsaaSamples*** Number of fullscreen antialasing samples (Do not use FSAA samples other than 0 now with nVidia cards!)

4.11.3.2 ~GLXRenderWindow () [virtual]

The destructor. This class has virtual destructor.

4.11.4 Member Function Documentation

4.11.4.1 void _reposition () [inline, protected, virtual]

Efficiently set window position (for internal use).

4.11.4.2 void _resize () [inline, protected, virtual]

Efficiently set window sizes (for internal use).

4.11.4.3 void _setCaption () [protected, virtual]

Efficiently set window caption (for internal use).

4.11.4.4 void createWindow () [protected, virtual]

Create GLX Window. Protected method for internal use.

4.11.4.5 void destroyWindow () [protected, virtual]

Destroy GLX Window. Protected method for internal use.

4.11.4.6 void finalize () [virtual]

Finalize the GLX RenderWindow. Protected method for internal use.

4.11.4.7 void finishFrame () [inline, virtual]

Finish current frame.

4.11.4.8 const std::string & getCaption () const [inline]

Getter of mCaption. Returns value of mCaption.

Returns:
The value of mCaption

4.11.4.9 const u32 & getColourDepth () const [inline]

Getter of mColourDepth. Returns value of mColourDepth.

Returns:
The value of mColourDepth

4.11.4.10 XDisplay::Pointer & getDisplay () [inline]

Getter of mDisplay. Returns value of mDisplay.

Returns:
The value of mDisplay

4.11.4.11 const u32 & getFrameNumber () const [inline]

Getter of mFrameNumber. Returns value of mFrameNumber.

Returns:
The value of mFrameNumber

4.11.4.12 const u32 & getFSAASamples () const [inline]

Getter of mFSAASamples. Returns value of mFSAASamples.

Returns:
The value of mFSAASamples

4.11.4.13 const bool & getFullScreen () const [inline]

Getter of mFullScreen. Returns value of mFullScreen.

Returns:
The value of mFullScreen

4.11.4.14 GLXGLContext::Pointer & getGLXGLContext () [inline]

Getter of mGLXGLContext. Returns value of mGLXGLContext.

Returns:
The value of mGLXGLContext

4.11.4.15 const u32 & getHeight () const [inline]

Getter of mHeight. Returns value of mHeight.

Returns:
The value of mHeight

4.11.4.16 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:
The value of mInitialized

4.11.4.17 const s32 & getLeft () const [inline]

Getter of mLeft. Returns value of mLeft.

Returns:
The value of mLeft

4.11.4.18 Process * getProcess () const [inline]

Getter of mProcess. Returns value of mProcess.

Returns:
The value of mProcess

4.11.4.19 const s32 & getTop () const [inline]

Getter of mTop. Returns value of mTop.

Returns:
The value of mTop

4.11.4.20 const bool & getVisible () const [inline]

Getter of mVisible. Returns value of mVisible.

Returns:
The value of mVisible

4.11.4.21 const u32 & getWidth () const [inline]

Getter of mWidth. Returns value of mWidth.

Returns:
The value of mWidth

4.11.4.22 const ::Window & getWindow () const [inline]

Getter of mWindow. Returns value of mWindow.

Returns:
The value of mWindow

4.11.4.23 const GLXRenderWindow::WindowStatistics & getWindowStatistics () const [inline]

Getter of mWindowStatistics. Returns value of mWindowStatistics.

Returns:
The value of mWindowStatistics

4.11.4.24 void initialize () [virtual]

Initialize the GLX RenderWindow. Protected method for internal use.

4.11.4.25 void reposition (const s32 & *left*, const s32 & *top*) [inline, virtual]

Set new window position.

Parameters:
← ***left*** Horizontal position
← ***top*** Vertical position

4.11.4.26 void resetStatistics () [protected, virtual]

Reset statistics. Protected method for internal use.

4.11.4.27 void resize (const u32 & *width*, const u32 & *height*) [inline, virtual]

Set new window sizes.

Parameters:
← ***width*** Horizontal size
← ***height*** Vertical size

4.11.4.28 void setCaption (const std::string & *caption*) [inline]

Setter of mCaption. Sets value of mCaption.

Parameters:
← ***caption*** The value of mCaption

4.11.4.29 void setColourDepth (const u32 & *colourdepth*) [inline]

Setter of mColourDepth. Sets value of mColourDepth.

Parameters:
← ***colourdepth*** The value of mColourDepth

4.11.4.30 void setCurrent () [virtual]

Enable the context of the window. All subsequent rendering commands will go on this window. The sideeffect of calling this method is reseting the window statistics.

4.11.4.31 void setFSAASamples (const u32 & *fsaasamples*) [inline]

Setter of mFSAASamples. Sets value of mFSAASamples.

Parameters:
← ***fsaasamples*** The value of mFSAASamples

4.11.4.32 void setFullScreen (const bool & *fullscreen*) [inline]

Setter of mFullScreen. Sets value of mFullScreen.

Parameters:
← ***fullscreen*** The value of mFullScreen

4.11.4.33 void setHeight (const u32 & *height*) [inline]

Setter of mHeight. Sets value of mHeight.

Parameters:
← ***height*** The value of mHeight

4.11.4.34 void setLeft (const s32 & *left*) [inline]

Setter of mLeft. Sets value of mLeft.

Parameters:
← ***left*** The value of mLeft

4.11.4.35 void setTop (const s32 & *top*) [inline]

Setter of mTop. Sets value of mTop.

Parameters:
← ***top*** The value of mTop

4.11.4.36 void setVisible (const bool & *visible*) [inline]

Setter of mVisible. Sets value of mVisible.

Parameters:
← ***visible*** The value of mVisible

4.11.4.37 void setWidth (const u32 & *width*) [inline]

Setter of mWidth. Sets value of mWidth.

Parameters:
← ***width*** The value of mWidth

4.11.4.38 void startFrame () [inline, virtual]

Start a frame.

4.11.4.39 void updateStatistics () [protected, virtual]

Update statistics of the window. Protected method for internal use.

4.11.5 Member Data Documentation

4.11.5.1 const s32 CENTERED = -1 [static]

Constant for centered position.

Remarks:
This is own attribute of this class.

4.11.5.2 ::Atom mAtomDeleteWindow [protected]

Atom to recognize window close events.

Remarks:
This is own attribute of this class.

4.11.5.3 const u32 MAXIMALSIZE = 0 [static]

Constant for maximal size.

Remarks:
This is own attribute of this class.

4.11.5.4 std::string mCaption [protected]

Window caption.

Remarks:
This is own attribute of this class.

4.11.5.5 u32 mColourDepth [protected]

Colour depth of the window.

Remarks:
This is own attribute of this class.

4.11.5.6 XDisplay::Pointer mDisplay [protected]

Corresponding X Display.

Remarks:
This is own attribute of this class.

4.11.5.7 Real mFrameBeginTime [protected]

Time of stating a new frame.

Remarks:
This is own attribute of this class.

4.11.5.8 u32 mFrameNumber [protected]

Number of current frame.

Remarks:
This is own attribute of this class.

4.11.5.9 u32 mFSAASamples [protected]

Number of fullscreen antialasing samples.

Remarks:
This is own attribute of this class.

4.11.5.10 bool mFullScreen [protected]

The window appears in full screen mode.

Remarks:
This is own attribute of this class.

4.11.5.11 GLXGLContext::Pointer mGLXGLContext [protected]

GLX context of the render window.

Remarks:
This is own attribute of this class.

4.11.5.12 u32 mHeight [protected]

Vertical size of the window.

Remarks:
This is own attribute of this class.

4.11.5.13 bool mInitialized [protected]

The window is initialized.

Remarks:
This is own attribute of this class.

4.11.5.14 s32 mLeft [protected]

Horizontal position of the window.

Remarks:
This is own attribute of this class.

4.11.5.15 s32 mOriginalXRRConfiguation [protected]

For storing original XRR configuration mode.

Remarks:
This is own attribute of this class.

4.11.5.16 Process* mProcess [protected]

Corresponding process.

Remarks:
This attribute references an attribute.

4.11.5.17 Real mSumFPS [protected]

Summarized frame rates for all frames for calculating avgFPS statistics.

Remarks:
This is own attribute of this class.

4.11.5.18 Real mSumTriangleCount [protected]

Summarized triangle count for all frames for calculating avgTriangleCount statistics.

Remarks:
This is own attribute of this class.

4.11.5.19 s32 mTop [protected]

Vertical position of the window.

Remarks:
This is own attribute of this class.

4.11.5.20 bool mVisible [protected]

The window is visible.

Remarks:
This is own attribute of this class.

4.11.5.21 u32 mWidth [protected]

Horizontal size of the window.

Remarks:
This is own attribute of this class.

4.11.5.22 ::Window mWindow [protected]

Wrapped GLX Window.

Remarks:
This is own attribute of this class.

4.11.5.23 WindowStatistics mWindowStatistics [protected]

Current window statistics.

Remarks:
This is own attribute of this class.

4.11.5.24 const Real UNDEFINEDSTATISTICS = -1.0 [static]

Undefined statistics value.

Remarks:
This is own attribute of this class.

4.11.5.25 const s32 UNDEFINEDXRRCONFIGURATION = -1 [static]

Constant for undefined XRR configuration.

Remarks:
This is own attribute of this class.

4.12 GLXRenderWindow::WindowStatistics Struct Reference

4.12.1 Detailed Description

Struct for window statistics (FPS values, frame times).

Public Attributes

- **Real lastFPS**
- **Real avgFPS**
- **Real bestFPS**
- **Real worstFPS**
- **Real lastFrameTime**
- **Real bestFrameTime**
- **Real worstFrameTime**
- **Real lastTriangleCount**
- **Real avgTriangleCount**
- **Real minTriangleCount**
- **Real maxTriangleCount**

4.12.2 Member Data Documentation

4.12.2.1 Real avgFPS

Average frame rate

4.12.2.2 Real avgTriangleCount

Average triangle count

4.12.2.3 Real bestFPS

Frame rate for the frame with the best frame time

4.12.2.4 Real bestFrameTime

Best frame time

4.12.2.5 Real lastFPS

Frame rate for the last frame

4.12.2.6 Real lastFrameTime

Last frame time

4.12.2.7 Real lastTriangleCount

Triangle count for the last frame

4.12.2.8 Real maxTriangleCount

Maximal triangle count

4.12.2.9 Real minTriangleCount

Minimal triangle count

4.12.2.10 Real worstFPS

Frame rate for the frame with the worst frame time

4.12.2.11 Real worstFrameTime

Worst frame time

4.13 HandleClient Class Reference

Inherits **Thread**.

4.13.1 Detailed Description

Class for handle client messages.

Methods

- virtual void **sendMessage** (const std::string &mType, const std::string &message)
- virtual std::string **createInitString** ()
- virtual void **task** ()

Public Types

- typedef **Pointer**< **HandleClient**, **DummyLock** > **Pointer**

Public Member Functions

Constructors & destructor

- **HandleClient** (const **Network** *net)
- virtual ~**HandleClient** ()

Getters & setters

- const int & **getSocket** () const
- void **setSocket** (const int &socket)
- **Network** * **getParentNetwork** () const
- void **setParentNetwork** (**Network** *parentnetwork)

Protected Attributes

Variables

- int **mSocket**
- **Network** * **mParentNetwork**

4.13.2 Member Typedef Documentation

4.13.2.1 typedef Pointer< HandleClient, DummyLock > Pointer

Type for pointer or this class.

4.13.3 Constructor & Destructor Documentation

4.13.3.1 HandleClient (const Network * *net*)

Create **HandleClient**(p. 66) class.

Parameters:
← ***net*** Parent network.

4.13.3.2 ~HandleClient () [virtual]

The destructor. This class has virtual destructor.

4.13.4 Member Function Documentation

4.13.4.1 std::string createInitString () [virtual]

Send a TCP/IP message.

Returns:
The initial string.

4.13.4.2 Network * getParentNetwork () const [inline]

Getter of mParentNetwork. Returns value of mParentNetwork.

Returns:
The value of mParentNetwork

4.13.4.3 const int & getSocket () const [inline]

Getter of mSocket. Returns value of mSocket.

Returns:
The value of mSocket

4.13.4.4 void sendMessage (const std::string & *mType*, const std::string & *message*) [virtual]

Send a TCP/IP message.

Parameters:
← ***mType*** Type of the message.

← ***message*** The message.

4.13.4.5 void setParentNetwork (Network * *parentnetwork*) [inline]

Setter of mParentNetwork. Sets value of mParentNetwork.

Parameters:
← ***parentnetwork*** The value of mParentNetwork

4.13.4.6 void setSocket (const int & *socket*) [inline]

Setter of mSocket. Sets value of mSocket.

Parameters:
← ***socket*** The value of mSocket

4.13.4.7 void task () [protected, virtual]

Handle a client.

Reimplemented from **Thread** (p. 139).

4.13.5 Member Data Documentation

4.13.5.1 Network* mParentNetwork [protected]

Parent network.

Remarks:
This is own attribute of this class.

4.13.5.2 int mSocket [protected]

The communication socket.

Remarks:
This is own attribute of this class.

4.14 Host Class Reference

Inherits **HostInfo**, and **Singleton**< **Host** >.

4.14.1 Detailed Description

Class for describe a hosts.

Public Member Functions

Constructors & destructor

- **Host** (const std::string &name)
- virtual ~**Host** ()

Getters & setters

- **Container**< **Node**, **Mutex** >::**Pointer** & **getNodes** ()
- const bool & **getInitialized** () const
- const **Real** & **getTimeCorrection** () const
- **OutputNode::Pointer** & **getOutputDocument** ()

Methods

- virtual **XDisplay::Pointer openXDisplay** (const std::string &displayName="")
- virtual void **init** (const std::string &conf)
- virtual void **stop** ()
- virtual void **finalize** ()
- virtual void **collectData** ()

Protected Attributes

Variables

- **Container**< **XDisplay**, **Mutex** >::**Pointer mXDisplays**
- **Container**< **Node**, **Mutex** >::**Pointer mNodes**
- bool **mInitialized**
- **Real mTimeCorrection**
- **OutputNode::Pointer mOutputDocument**

4.14.2 Constructor & Destructor Documentation

4.14.2.1 Host (const std::string & *name*)

Creates a host with a specified name.

Parameters:

← ***name*** **Name**(p. 83) of the host.

4.14.2.2 ~Host () [virtual]

The destructor. This class has virtual destructor.

4.14.3 Member Function Documentation

4.14.3.1 void collectData () [virtual]

Collect data from nodes.

4.14.3.2 void finalize () [virtual]

Finalize host.

4.14.3.3 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:
The value of mInitialized

4.14.3.4 Container< Node, Mutex >::Pointer & getNodes () [inline]

Getter of mNodes. Returns value of mNodes.

Returns:
The value of mNodes

4.14.3.5 OutputNode::Pointer & getOutputDocument () [inline]

Getter of mOutputDocument. Returns value of mOutputDocument.

Returns:
The value of mOutputDocument

4.14.3.6 const Real & getTimeCorrection () const [inline]

Getter of mTimeCorrection. Returns value of mTimeCorrection.

Returns:
The value of mTimeCorrection

4.14.3.7 void init (const std::string & *conf*) [virtual]

Create and call init functions of the nodes on the hosts.

Parameters:
← ***conf*** Config string.

4.14.3.8 XDisplay::Pointer openXDisplay (const std::string & *displayName* = "") [virtual]

Open an X display. An X display can be opened several times, and does not have to be closed. The host will close it when it is destructed. When no parameter is set, the default display will be opened.

Parameters:
← ***displayName*** X display name

Returns:
Opened X Display

4.14.3.9 void stop () [virtual]

Stop the nodes.

4.14.4 Member Data Documentation

4.14.4.1 bool mInitialized [protected]

The host is initialized.

Remarks:
This is own attribute of this class.

4.14.4.2 Container< Node, Mutex >::Pointer mNodes [protected]

Nodes on this host.

Remarks:
This is own attribute of this class.

4.14.4.3 OutputNode::Pointer mOutputDocument [protected]

Root of the output document on this host.

Remarks:
This is own attribute of this class.

4.14.4.4 Real mTimeCorrection [protected]

Time correction for this host according to the host of the commander mode application.

Remarks:
This is own attribute of this class.

4.14.4.5 Container< XDisplay, Mutex >::Pointer mXDisplays [protected]

X Displays on this host.

Remarks:
This is own attribute of this class.

4.15 HostInfo Class Reference

Inherits **Name**.

Inherited by **Host**.

4.15.1 Detailed Description

Class for contain host information.

Public Member Functions

Constructors & destructor

- **HostInfo** (const std::string &name)
- virtual ~**HostInfo** ()

4.15.2 Constructor & Destructor Documentation

4.15.2.1 HostInfo (const std::string & *name*)

Creates a host with a specified name.

Parameters:
← ***name*** **Name**(p. 83) of the host.

4.15.2.2 ~HostInfo () `[virtual]`

The destructor. This class has virtual destructor.

4.16 Lock Class Reference

Inherited by **DummyLock**, and **Mutex**.

4.16.1 Detailed Description

Lock(p. 74) interface. Provides lock, trylock and unlock methods.

Public Member Functions

Constructors & destructor

- **Lock** ()

Getters & setters

- const bool & **getLocked** () const

Methods

- virtual void **lock** ()=0
- virtual bool **trylock** ()=0
- virtual void **unlock** ()=0

Protected Attributes

Variables

- bool **mLocked**

4.16.2 Constructor & Destructor Documentation

4.16.2.1 Lock () [inline]

Default constructor.

4.16.3 Member Function Documentation

4.16.3.1 const bool & getLocked () const [inline]

Getter of mLocked. Returns value of mLocked.

Returns:
The value of mLocked

4.16.3.2 virtual void lock () [pure virtual]

Lock(p. 74) the lock.

Implemented in **DummyLock** (p. 40), and **Mutex** (p. 81).

4.16.3.3 virtual bool trylock () [pure virtual]

Try locking the lock.

Returns:
True if the locking was successful.

Implemented in **DummyLock** (p. 40), and **Mutex** (p. 82).

4.16.3.4 virtual void unlock () [pure virtual]

Unlock the lock.

Implemented in **DummyLock** (p. 40), and **Mutex** (p. 82).

4.16.4 Member Data Documentation

4.16.4.1 bool mLocked [protected]

Indicates that the lock is locked.

Remarks:
This is own attribute of this class.

4.17 Logger Class Reference

Inherits **Singleton**< **Logger** >.

4.17.1 Detailed Description

Class for very simple logging mechanism.

Public Types

- enum **LogLevel** {
  FATAL, **ERROR**, **WARNING**, **NOTICE**,
  DEBUG }

Public Member Functions

Constructors & destructor

- **Logger** (const std::string &logFileName="pcm.log")
- virtual ~**Logger** ()

Getters & setters

- const **u8** & **getLogMode** () const
- const **Logger::LogLevel** & **getConsoleLogLevel** () const
- void **setConsoleLogLevel** (const **Logger::LogLevel** &consoleloglevel)
- const **Logger::LogLevel** & **getFileLogLevel** () const
- void **setFileLogLevel** (const **Logger::LogLevel** &fileloglevel)
- const bool & **getInitialized** () const
- const std::string & **getLogFileName** () const
- void **setLogFileName** (const std::string &logfilename)

Methods

- virtual void **init** ()
- virtual void **log** (const **LogLevel** &loglevel, const std::string &message)
- virtual void **logMultiLine** (const **LogLevel** &loglevel, const std::string &message)
- virtual void **log** (const **Exception** &exception)

Static Public Member Functions

Class methods

- static std::string **translateLogLevel** (const **LogLevel** &loglevel)

Static Public Attributes

Class constants

- static const **u8 LOGTOCONSOLE** = 1
- static const **u8 LOGTOFILE** = 2

Protected Attributes

Variables

- **u8 mLogMode**
- **LogLevel mConsoleLogLevel**
- **LogLevel mFileLogLevel**
- **Pointer**< FILE, **Mutex** > **mFp**
- bool **mInitialized**
- std::string **mLogFileName**

4.17.2 Member Enumeration Documentation

4.17.2.1 enum LogLevel

Definitions logging levels.

Enumerator:

FATAL Fatal error.

ERROR Error.

WARNING Warning.

NOTICE Notice.

DEBUG Debug.

4.17.3 Constructor & Destructor Documentation

4.17.3.1 Logger (const std::string & *logFileName* = "pcm.log")

Create logger

Parameters:

← ***logFileName*** **Name**(p. 83) of the log file.

4.17.3.2 ~Logger () [virtual]

The destructor. This class has virtual destructor.

4.17.4 Member Function Documentation

4.17.4.1 const Logger::LogLevel & getConsoleLogLevel () const [inline]

Getter of mConsoleLogLevel. Returns value of mConsoleLogLevel.

Returns:

The value of mConsoleLogLevel

4.17.4.2 const Logger::LogLevel & getFileLogLevel () const [inline]

Getter of mFileLogLevel. Returns value of mFileLogLevel.

Returns:
The value of mFileLogLevel

4.17.4.3 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:
The value of mInitialized

4.17.4.4 const std::string & getLogFileName () const [inline]

Getter of mLogFileName. Returns value of mLogFileName.

Returns:
The value of mLogFileName

4.17.4.5 const u8 & getLogMode () const [inline]

Getter of mLogMode. Returns value of mLogMode.

Returns:
The value of mLogMode

4.17.4.6 void init () [virtual]

Initializes the **Logger**(p. 76).

4.17.4.7 void log (const Exception & *exception*) [inline, virtual]

Logs an exception.

Parameters:
← ***exception*** **Exception**(p. 43).

4.17.4.8 void log (const LogLevel & *loglevel*, const std::string & *message*) [inline, virtual]

Logs a message. (Does not care about multiline messages, because of performance).

Parameters:
← ***loglevel*** Message level.
← ***message*** Message string.

4.17.4.9 void logMultiLine (const LogLevel & *loglevel*, const std::string & *message*) [virtual]

Logs a message. Handles multiline messages too.

Parameters:
← ***loglevel*** Message level.
← ***message*** Message string.

4.17.4.10 void setConsoleLogLevel (const Logger::LogLevel & *consoleloglevel*) [inline]

Setter of mConsoleLogLevel. Sets value of mConsoleLogLevel.

Parameters:
← ***consoleloglevel*** The value of mConsoleLogLevel

4.17.4.11 void setFileLogLevel (const Logger::LogLevel & *fileloglevel*) [inline]

Setter of mFileLogLevel. Sets value of mFileLogLevel.

Parameters:
← ***fileloglevel*** The value of mFileLogLevel

4.17.4.12 void setLogFileName (const std::string & *logfilename*) [inline]

Setter of mLogFileName. Sets value of mLogFileName.

Parameters:
← ***logfilename*** The value of mLogFileName

4.17.4.13 std::string translateLogLevel (const LogLevel & *loglevel*) [inline, static]

Translate log level to human readable.

Parameters:
← ***loglevel*** Logging level.

Returns:

4.17.5 Member Data Documentation

4.17.5.1 const u8 LOGTOCONSOLE = 1 [static]

Indicates that the logger logs to console.

Remarks:
This is own attribute of this class.

4.17.5.2 const u8 LOGTOFILE = 2 [static]

Indicates that the logger logs to a textfile.

Remarks:
This is own attribute of this class.

4.17.5.3 LogLevel mConsoleLogLevel [protected]

Logging level for console (from FATAL to NOTICE).

Remarks:
This is own attribute of this class.

4.17.5.4 LogLevel mFileLogLevel [protected]

Logging level for file (from FATAL to NOTICE).

Remarks:
This is own attribute of this class.

4.17.5.5 Pointer< FILE, Mutex > mFp [protected]

File pointer.

Remarks:
This is own attribute of this class.

4.17.5.6 bool mInitialized [protected]

Indicates that the logger is initialized.

Remarks:
This is own attribute of this class.

4.17.5.7 std::string mLogFileName [protected]

Name(p. 83) of the log file.

Remarks:
This is own attribute of this class.

4.17.5.8 u8 mLogMode [protected]

Logging mode (console and/or textfile).

Remarks:
This is own attribute of this class.

4.18 Mutex Class Reference

Inherits **Lock**.

4.18.1 Detailed Description

Mutual exception lock.

Public Member Functions

Constructors & destructor

- **Mutex** ()
- virtual ~**Mutex** ()

Methods

- virtual void **lock** ()
- virtual bool **trylock** ()
- virtual void **unlock** ()

Protected Attributes

Variables

- pthread_mutex_t **mMutex**

4.18.2 Constructor & Destructor Documentation

4.18.2.1 Mutex () [inline]

Default constructor.

4.18.2.2 ~Mutex () [inline, virtual]

The destructor. This class has virtual destructor.

4.18.3 Member Function Documentation

4.18.3.1 void lock () [inline, virtual]

Lock(p. 74) the mutex.

Implements **Lock** (p. 74).

4.18.3.2 bool trylock () [inline, virtual]

Try locking the mutex.

Returns:
True if the locking was successful.

Implements **Lock** (p. 75).

4.18.3.3 void unlock () [inline, virtual]

Unlock the mutex.

Implements **Lock** (p. 75).

4.18.4 Member Data Documentation

4.18.4.1 pthread_mutex_t mMutex [protected]

Guard mutex.

Remarks:
This is own attribute of this class.

4.19 Name Class Reference

Inherited by **HostInfo**, **Node**, **OutputNode**, **Process**, and **SqVM**.

4.19.1 Detailed Description

This is a dummy class for inheritance. Contain grid data.

Public Member Functions

Constructors & destructor

- **Name** ()
- **Name** (const std::string &name)
- virtual ~**Name** ()

Getters & setters

- const std::string & **getName** () const
- void **setName** (const std::string &name)

Protected Attributes

Variables

- std::string **mName**

4.19.2 Constructor & Destructor Documentation

4.19.2.1 Name ()

Default constructor.

4.19.2.2 Name (const std::string & *name*)

Default constructor.

Parameters:
← ***name*** **Name**(p. 83) of **Name**(p. 83):)

4.19.2.3 ~Name () [virtual]

The destructor. This class has virtual destructor.

4.19.3 Member Function Documentation

4.19.3.1 const std::string & getName () const [inline]

Getter of mName. Returns value of mName.

Returns:
The value of mName

4.19.3.2 void setName (const std::string & *name*) [inline]

Setter of mName. Sets value of mName.

Parameters:
← ***name*** The value of mName

4.19.4 Member Data Documentation

4.19.4.1 std::string mName [protected]

Name(p. 83).

Remarks:
This is own attribute of this class.

4.20 Network Class Reference

Inherits **Thread**.

4.20.1 Detailed Description

Class for network routines.

Getters & setters

- const bool & **getCommanderMode** () const
- void **setCommanderMode** (const bool &commandermode)
- const int & **getServerSocket** () const
- const int & **getClientSocket** () const
- const int & **getBroadcastSendSocket** () const
- const int & **getBroadcastRecieveSocket** () const
- const struct sockaddr_in & **getBroadcastAddress** () const
- const int & **getNodeNumber** () const
- void **setNodeNumber** (const int &nodenumber)
- const std::string & **getMasterNodeIP** () const
- void **setMasterNodeIP** (const std::string &masternodeip)
- const std::string & **getOwnIP** () const
- static const unsigned short & **getCommunicationPort** ()
- static const unsigned short & **getBroadcastPort** ()

Methods

- virtual void **initServer** ()
- virtual void **initClient** ()
- virtual void **closeClient** ()
- virtual void **initBroadcastSend** ()
- virtual void **initBroadcastRecieve** ()
- virtual void **sendMessage** (const std::string &type, const std::string &message)
- virtual void **sendBroadcastMessage** (const std::string &type, const std::string &message)
- virtual void **getIP** ()
- virtual void **recieveBroadcastMessage** ()
- virtual void **recieveMessage** ()
- virtual void **acceptClientConnection** ()
- virtual void **initNetwork** ()
- virtual void **buildCluster** ()
- virtual void **task** ()

Public Types

- typedef **Pointer**< **Network**, **DummyLock** > **Pointer**
- enum **MessageType** {
  INIT, **LOGIN**, **TIME**, **RESULT**,
  INITOK }

Public Member Functions

Constructors & destructor

- **Network** (const std::string &name)
- virtual ~**Network** ()

Static Public Member Functions

Class methods

- static **Network::MessageType str2enum** (const std::string &strType)
- static std::string **enum2str** (const **Network::MessageType** &enumType)

Protected Attributes

Variables

- bool **mCommanderMode**
- int **mServerSocket**
- int **mClientSocket**
- int **mBroadcastSendSocket**
- int **mBroadcastRecieveSocket**
- sockaddr_in **mBroadcastAddress**
- int **mNodeNumber**
- std::string **mMasterNodeIP**
- std::string **mOwnIP**

Static Protected Attributes

Class variables

- static unsigned short **mCommunicationPort** = 1234
- static unsigned short **mBroadcastPort** = 1235

4.20.2 Member Typedef Documentation

4.20.2.1 typedef Pointer< Network, DummyLock > Pointer

Type for pointer or this class.

4.20.3 Member Enumeration Documentation

4.20.3.1 enum MessageType

Type of sended message.

Enumerator:

INIT Initiate string message.

LOGIN Client must log in.

TIME Client send their local time.

RESULT Client send their local time.

INITOK Wait for init message.

4.20.4 Constructor & Destructor Documentation

4.20.4.1 Network (const std::string & *name*)

Create network class.

Parameters:
← ***name*** **Name**(p. 83) of the network.

4.20.4.2 ~Network () [virtual]

The destructor. This class has virtual destructor.

4.20.5 Member Function Documentation

4.20.5.1 void acceptClientConnection () [virtual]

Server accept the client connections.

4.20.5.2 void buildCluster () [virtual]

Build cluster information.

4.20.5.3 void closeClient () [virtual]

Init a client process.

4.20.5.4 std::string enum2str (const Network::MessageType & *enumType*) [static]

String convert to enum.

Parameters:
← ***enumType*** Type as enum.

Returns:
Type as string.

4.20.5.5 const struct sockaddr_in & getBroadcastAddress () const [inline]

Getter of mBroadcastAddress. Returns value of mBroadcastAddress.

Returns:
The value of mBroadcastAddress

4.20.5.6 const unsigned short & getBroadcastPort () [inline, static]

Getter of mBroadcastPort. Returns value of mBroadcastPort.

Returns:
The value of mBroadcastPort

4.20.5.7 const int & getBroadcastRecieveSocket () const [inline]

Getter of mBroadcastRecieveSocket. Returns value of mBroadcastRecieveSocket.

Returns:
The value of mBroadcastRecieveSocket

4.20.5.8 const int & getBroadcastSendSocket () const [inline]

Getter of mBroadcastSendSocket. Returns value of mBroadcastSendSocket.

Returns:
The value of mBroadcastSendSocket

4.20.5.9 const int & getClientSocket () const [inline]

Getter of mClientSocket. Returns value of mClientSocket.

Returns:
The value of mClientSocket

4.20.5.10 const bool & getCommanderMode () const [inline]

Getter of mCommanderMode. Returns value of mCommanderMode.

Returns:
The value of mCommanderMode

4.20.5.11 const unsigned short & getCommunicationPort () [inline, static]

Getter of mCommunicationPort. Returns value of mCommunicationPort.

Returns:
The value of mCommunicationPort

4.20.5.12 void getIP () [virtual]

Get local machine IP address-es.

4.20.5.13 const std::string & getMasterNodeIP () const [inline]

Getter of mMasterNodeIP. Returns value of mMasterNodeIP.

Returns:
The value of mMasterNodeIP

4.20.5.14 const int & getNodeNumber () const [inline]

Getter of mNodeNumber. Returns value of mNodeNumber.

Returns:
The value of mNodeNumber

4.20.5.15 const std::string & getOwnIP () const [inline]

Getter of mOwnIP. Returns value of mOwnIP.

Returns:
The value of mOwnIP

4.20.5.16 const int & getServerSocket () const [inline]

Getter of mServerSocket. Returns value of mServerSocket.

Returns:
The value of mServerSocket

4.20.5.17 void initBroadcastRecieve () [virtual]

Init a broadcast reciever process.

4.20.5.18 void initBroadcastSend () [virtual]

Init a broadcast sender process.

4.20.5.19 void initClient () [virtual]

Init a client process.

4.20.5.20 void initNetwork () [virtual]

Initial the network interface.

4.20.5.21 void initServer () [virtual]

Init a server process.

4.20.5.22 void recieveBroadcastMessage () [virtual]

Wait for a broadcast message.

4.20.5.23 void recieveMessage () [virtual]

Wait for a TCP/IP message.

4.20.5.24 void sendBroadcastMessage (const std::string & *type*, const std::string & *message*) [virtual]

Send a broadcast message.

Parameters:

← ***type*** Type of the message.

← ***message*** The message.

4.20.5.25 void sendMessage (const std::string & *type*, const std::string & *message*) [virtual]

Send message with TCP protocol.

Parameters:

← ***type*** Type of the message.

← ***message*** The message.

4.20.5.26 void setCommanderMode (const bool & *commandermode*) [inline]

Setter of mCommanderMode. Sets value of mCommanderMode.

Parameters:

← ***commandermode*** The value of mCommanderMode

4.20.5.27 void setMasterNodeIP (const std::string & *masternodeip*) [inline]

Setter of mMasterNodeIP. Sets value of mMasterNodeIP.

Parameters:

← ***masternodeip*** The value of mMasterNodeIP

4.20.5.28 void setNodeNumber (const int & *nodenumber*) [inline]

Setter of mNodeNumber. Sets value of mNodeNumber.

Parameters:
← ***nodenumber*** The value of mNodeNumber

4.20.5.29 Network::MessageType str2enum (const std::string & *strType*) [static]

String convert to enum.

Parameters:
← ***strType*** Type as string.

Returns:
Type as enum.

4.20.5.30 void task () [protected, virtual]

Thread(p. 135) for accept calling.

Reimplemented from **Thread** (p. 139).

4.20.6 Member Data Documentation

4.20.6.1 struct sockaddr_in mBroadcastAddress [protected]

The broadcast address structure for send later time.

Remarks:
This is own attribute of this class.

4.20.6.2 unsigned short mBroadcastPort = 1235 [static, protected]

The broadcast port number.

Remarks:
This is own attribute of this class.

4.20.6.3 int mBroadcastRecieveSocket [protected]

The broadcast socket number.

Remarks:
This is own attribute of this class.

4.20.6.4 int mBroadcastSendSocket [protected]

The broadcast socket number.

Remarks:
This is own attribute of this class.

4.20.6.5 int mClientSocket [protected]

The client socket number.

Remarks:
This is own attribute of this class.

4.20.6.6 bool mCommanderMode [protected]

Indicates commander mode (default false).

Remarks:
This is own attribute of this class.

4.20.6.7 unsigned short mCommunicationPort = 1234 [static, protected]

The communication port number.

Remarks:
This is own attribute of this class.

4.20.6.8 std::string mMasterNodeIP [protected]

IP address of the master node.

Remarks:
This is own attribute of this class.

4.20.6.9 int mNodeNumber [protected]

Number of the nodes in the grid.

Remarks:
This is own attribute of this class.

4.20.6.10 std::string mOwnIP [protected]

IP address of the master node.

Remarks:
This is own attribute of this class.

4.20.6.11 int mServerSocket [protected]

The server socket number.

Remarks:
This is own attribute of this class.

4.21 Node Class Reference

Inherits **Name**.

4.21.1 Detailed Description

Node(p. 94) class. Entity for composite and/or render a framelet.

Public Types

- typedef **Pointer**< **Node**, **Mutex** > **Pointer**

Public Member Functions

Constructors & destructor

- **Node** (const std::string &name, **Host** *parent)
- virtual ~**Node** ()

Getters & setters

- **Host** * **getParent** () const
- const PCstring & **getSearchPath** () const
- **Container**< **Process**, **Mutex** >::**Pointer** & **getProcesses** ()
- **OutputNode::Pointer** & **getOutputDocument** ()

Methods

- virtual void **init** (const std::string &conf)
- virtual void **stop** ()
- virtual void **collectData** ()

Protected Attributes

Variables

- **Host** * **mParent**
- PCstring **mSearchPath**
- **Container**< **Process**, **Mutex** >::**Pointer mProcesses**
- **OutputNode::Pointer mOutputDocument**

4.21.2 Member Typedef Documentation

4.21.2.1 typedef Pointer< Node, Mutex > Pointer

Type for pointer on this class.

4.21.3 Constructor & Destructor Documentation

4.21.3.1 Node (const std::string & *name*, Host * *parent*)

Create node. Normally **Host**(p. 69) calls this constructor.

Parameters:

← ***name*** **Name**(p. 83) of the node.

← ***parent*** Parent host

4.21.3.2 ~Node () [virtual]

The destructor. This class has virtual destructor.

4.21.4 Member Function Documentation

4.21.4.1 void collectData () [virtual]

Collect data from nodes.

4.21.4.2 OutputNode::Pointer & getOutputDocument () [inline]

Getter of mOutputDocument. Returns value of mOutputDocument.

Returns:

The value of mOutputDocument

4.21.4.3 Host * getParent () const [inline]

Getter of mParent. Returns value of mParent.

Returns:

The value of mParent

4.21.4.4 Container< Process, Mutex >::Pointer & getProcesses () [inline]

Getter of mProcesses. Returns value of mProcesses.

Returns:

The value of mProcesses

4.21.4.5 const PCstring & getSearchPath () const [inline]

Getter of mSearchPath. Returns value of mSearchPath.

Returns:

The value of mSearchPath

4.21.4.6 void init (const std::string & *conf*) [virtual]

Create and call init functions of the nodes on the hosts.

Parameters:
← ***conf*** Config string.

4.21.4.7 void stop () [virtual]

Stop the threads.

4.21.5 Member Data Documentation

4.21.5.1 OutputNode::Pointer mOutputDocument [protected]

Node(p. 94) output document.

Remarks:
This is own attribute of this class.

4.21.5.2 Host* mParent [protected]

Parent host of the node. (Parent reference is standard pointer to avoid circular reference)

Remarks:
This attribute references an attribute.

4.21.5.3 Container< Process, Mutex >::Pointer mProcesses [protected]

Processes on this node.

Remarks:
This is own attribute of this class.

4.21.5.4 PCstring mSearchPath [protected]

Search path of PC library.

Remarks:
This is own attribute of this class.

4.22 OldContainer Class Reference

4.22.1 Detailed Description

Class for contain something in a hashmap.

Public Types

- typedef **Pointer**< **OldContainer**, **Mutex** > **Pointer**
- typedef std::map< std::string, **Name** * > **Map**
- typedef Map::iterator **Iterator**

Public Member Functions

Constructors & destructor

- **OldContainer** ()
- virtual ~**OldContainer** ()

Getters & setters

- const int & **getElementNumber** () const

Methods

- virtual void **add** (**Name** *&element)
- virtual void **add** (std::string name)
- virtual **Name** * **get** (const std::string &name)
- virtual std::string * **getList** ()
- virtual bool **has** (const std::string &name)
- virtual void **remove** (const std::string &name)
- virtual void **remove** (**Name** *element)

Protected Attributes

Variables

- **Map mElements**
- int **mElementNumber**

4.22.2 Member Typedef Documentation

4.22.2.1 typedef Map::iterator Iterator

Type definition for iterator on map of elements.

4.22.2.2 typedef std::map< std::string, Name * > Map

Type definition for map of elements.

4.22.2.3 typedef Pointer< OldContainer, Mutex > Pointer

Type for pointer on this class.

4.22.3 Constructor & Destructor Documentation

4.22.3.1 OldContainer ()

Default constructor.

4.22.3.2 ~OldContainer () [virtual]

The destructor. This class has virtual destructor.

4.22.4 Member Function Documentation

4.22.4.1 void add (std::string *name*) [virtual]

Add an element by name.

Parameters:
← ***name*** name of the element

4.22.4.2 void add (Name *& *element*) [virtual]

Add an element by object.

Parameters:
→ ***element*** pointer to the element

4.22.4.3 Name * get (const std::string & *name*) [virtual]

Get an element by name.

Parameters:
← ***name*** name of the element

Returns:
Pointer(p. 107) to the element

4.22.4.4 const int & getElementNumber () const [inline]

Getter of mElementNumber. Returns value of mElementNumber.

Returns:
The value of mElementNumber

4.22.4.5 std::string * getList () [virtual]

Get a list.

Returns:
List from the elements.

4.22.4.6 bool has (const std::string & *name*) [virtual]

Search for an element by name. Already exist?

Parameters:
← ***name*** name of the element

Returns:
True is the **OldContainer**(p. 97) has the element.

4.22.4.7 void remove (Name * *element*) [virtual]

Remove an element by pointer to the removed element.

Parameters:
← ***element*** name of the element

4.22.4.8 void remove (const std::string & *name*) [virtual]

Remove an element by name.

Parameters:
← ***name*** name of the element

4.22.5 Member Data Documentation

4.22.5.1 int mElementNumber [protected]

Number of elements.

Remarks:
This is own attribute of this class.

4.22.5.2 Map mElements [protected]

Map of elements.

Remarks:
This is own attribute of this class.

4.23 OpenGLRenderingEngine Class Reference

4.23.1 Detailed Description

Collection of rendering methods implemented using OpenGL API.

Public Member Functions

Constructors & destructor

- **OpenGLRenderingEngine** ()
- virtual ∼**OpenGLRenderingEngine** ()

Static Public Member Functions

Scripting binding

- static void **squirrelGlue** ()

Class methods

- static void **drawTriangle** ()

4.23.2 Constructor & Destructor Documentation

4.23.2.1 OpenGLRenderingEngine ()

Default constructor.

4.23.2.2 ∼OpenGLRenderingEngine () `[virtual]`

The destructor. This class has virtual destructor.

4.23.3 Member Function Documentation

4.23.3.1 void drawTriangle () `[static]`

Draw a triangle.

4.23.3.2 static void squirrelGlue () `[inline, static]`

Static glue code method for Squirrel-C++ binding. This method should be call from each Squirrel virtual machines to generate the proper Squirrel glue code. To content of this method is fully autogenerated in the header implementation file, you should not modify it.

4.24 OutputNode Class Reference

Inherits **Name**.

4.24.1 Detailed Description

Node(p. 94) of the output document.

Getters & setters

- const **OutputNode::NodeType** & **getType** () const
- const std::string & **getText** () const
- void **setText** (const std::string &text)
- static const std::string & **getEXTNODENAME** ()

Public Types

- typedef **Pointer**< **OutputNode**, **Mutex** > **Pointer**
- typedef std::list< **OutputNode::Pointer** > **ChildNodeList**
- typedef ChildNodeList::iterator **ChildNodeListIterator**
- typedef std::map< std::string, std::string > **AttributeMap**
- typedef AttributeMap::const_iterator **AttributeMapIterator**
- enum **NodeType** {
  DEFINITION, **INFORMATION**, **REFERENCE**, **STATISTICS**,
  TEXT, **CDATA** }

Public Member Functions

Constructors & destructor

- **OutputNode** (const std::string &name, const **NodeType** &type)
- **OutputNode** (const std::string &text)
- virtual ~**OutputNode** ()

Methods

- virtual **OutputNode::Pointer createChildNode** (const std::string &name, const **NodeType** &type)
- virtual **OutputNode::Pointer createChildNode** (const std::string &text)
- virtual void **addChildNode** (**OutputNode::Pointer** &child)
- virtual bool **hasAttribute** (const std::string &attribute) const
- virtual const std::string & **getAttribute** (const std::string &attribute) const
- virtual void **setAttribute** (const std::string &attribute, const std::string &value)
- virtual std::ostringstream & **serialize2XML** (std::ostringstream &osstr)
- virtual std::string **serialize2XML** ()

Static Protected Member Functions

Class methods

- static bool **_testXMLName** (const std::string &name)
- static std::string **_convertSpecialChars** (const std::string &string)

Protected Attributes

Variables

- **NodeType mType**
- **AttributeMap mAttributes**
- **ChildNodeList mChildren**
- std::string **mText**

Static Protected Attributes

Class constants

- static const std::string **TEXTNODENAME** = "#text"

4.24.2 Member Typedef Documentation

4.24.2.1 typedef std::map< std::string, std::string > AttributeMap

Type definition for map of attributes.

4.24.2.2 typedef AttributeMap::const_iterator AttributeMapIterator

Type definition for iterator on map of attributes.

4.24.2.3 typedef std::list< OutputNode::Pointer > ChildNodeList

Type definition for list of child nodes.

4.24.2.4 typedef ChildNodeList::iterator ChildNodeListIterator

Type definition for iterator on list of child nodes.

4.24.2.5 typedef Pointer< OutputNode, Mutex > Pointer

Type for pointer on this class.

4.24.3 Member Enumeration Documentation

4.24.3.1 enum NodeType

Node(p. 94) type definitions.

Enumerator:

DEFINITION Define something (cluster, host, node, etc.), anything that can be described in the script or in the GUI.

INFORMATION Information about the hardware or software element. The user cannot redefine its value.

REFERENCE A reference, new element cannot be defined, only referencing an existing element is possible.

STATISTICS Statistics, it cannot be defined, referenced, or determined from the hardware directly. These values are measured during the benchmark process.

TEXT The node is a text field.

CDATA The node is a CDATA text field.

4.24.4 Constructor & Destructor Documentation

4.24.4.1 OutputNode (const std::string & *name*, const NodeType & *type*)

Creates output node.

Parameters:

← ***name*** **Name**(p. 83) of the node.

← ***type*** Type of the node.

4.24.4.2 OutputNode (const std::string & *text*)

Creates text field.

Parameters:

← ***text*** Text data.

4.24.4.3 ~OutputNode () `[virtual]`

The destructor. This class has virtual destructor.

4.24.5 Member Function Documentation

4.24.5.1 std::string _convertSpecialChars (const std::string & *string*) `[static, protected]`

Convert special characters to XML entites.

Parameters:

← ***string*** String to convert.

Returns:

Converted string

4.24.5.2 bool _testXMLName (const std::string & *name*) `[static, protected]`

Test name if its a correct xml name.

Parameters:

← ***name*** **Name**(p. 83) to test.

Returns:

True if the specified name is a correct XML name

4.24.5.3 void addChildNode (OutputNode::Pointer & *child*) [virtual]

Add child node.

Parameters:
→ ***child*** Child node.

4.24.5.4 OutputNode::Pointer createChildNode (const std::string & *text*) [virtual]

Create textual child node.

Parameters:
← ***text*** Text data.

Returns:
Created child node

4.24.5.5 OutputNode::Pointer createChildNode (const std::string & *name*, const NodeType & *type*) [virtual]

Create nontextual child node.

Parameters:
← ***name*** **Name**(p. 83) of the child node.

← ***type*** Type of the child node.

Returns:
Created child node

4.24.5.6 const std::string & getAttribute (const std::string & *attribute*) const [virtual]

Get an attribute by name.

Parameters:
← ***attribute*** **Name**(p. 83) of the attribute.

Returns:
Attribute value.

4.24.5.7 const std::string & getEXTNODENAME () [inline, static]

Getter of TEXTNODENAME. Returns value of TEXTNODENAME.

Returns:
The value of TEXTNODENAME

4.24.5.8 const std::string & getText () const [inline]

Getter of mText. Returns value of mText.

Returns:
The value of mText

4.24.5.9 const OutputNode::NodeType & getType () const [inline]

Getter of mType. Returns value of mType.

Returns:
The value of mType

4.24.5.10 bool hasAttribute (const std::string & *attribute*) const [virtual]

Return true if the node has attribute with the specified name.

Parameters:
← ***attribute*** **Name**(p. 83) of the attribute.

Returns:
The node has the specified attribute.

4.24.5.11 std::string serialize2XML () [virtual]

Serialize the node to XML (this is a recursive method, calls the same methods of the children too). Use the other version of this method with parameter std::ostringstream if it is possible for memory saving.

Returns:
Serialized node.

4.24.5.12 std::ostringstream & serialize2XML (std::ostringstream & *osstr*) [virtual]

Serialize the node to XML (this is a recursive method, calls the same methods of the children too).

Parameters:
→ ***osstr*** Output string stream.

Returns:
Serialized node (added to osstr).

4.24.5.13 void setAttribute (const std::string & *attribute*, const std::string & *value*) [virtual]

Set attribute, also create if does not exists, overwrite otherwise.

Parameters:
← ***attribute*** **Name**(p. 83) of the attribute.
← ***value*** Value of the attribute.

4.24.5.14 void setText (const std::string & *text*) [inline]

Setter of mText. Sets value of mText.

Parameters:
← ***text*** The value of mText

4.24.6 Member Data Documentation

4.24.6.1 AttributeMap mAttributes [protected]

Attributes of the node.

Remarks:
This is own attribute of this class.

4.24.6.2 ChildNodeList mChildren [protected]

Child nodes.

Remarks:
This is own attribute of this class.

4.24.6.3 std::string mText [protected]

Text data.

Remarks:
This is own attribute of this class.

4.24.6.4 NodeType mType [protected]

Type of the node.

Remarks:
This is own attribute of this class.

4.24.6.5 const std::string TEXTNODENAME = "#text" [static, protected]

String constant for node name for text data nodes.

Remarks:
This is own attribute of this class.

4.25 Pointer Class Template Reference

4.25.1 Detailed Description

template<typename T, class Lock> class ParCompMark::Pointer< T, Lock >

Template smart pointer class.

Remarks:

This class provides: exception safe, garbage colletion, thread safeness, and more efficiency

Methods

- virtual bool **isNull** () const
- virtual bool **isNotNull** () const
- virtual void **assignWithLock** (**Pointer**< T, **Lock** > &pointer)
- virtual void **reference** (const T *pointer)
- virtual T * **getPtr** ()
- virtual void **kill** (const bool &force=false)
- virtual void **setNull** (const bool &force=false)
- virtual void **lock** ()
- virtual bool **trylock** ()
- virtual void **unlock** ()
- virtual bool **getLocked** () const
- virtual void **_deletePointer** (const bool &force=false, const bool &keepLock=false)
- virtual void **_assignCPointer** (const T *pointer, const bool &takeOwnership=true)
- virtual void **_assignPointer** (**Pointer**< T, **Lock** > &pointer, const bool &keepLock=false)
- virtual void **_switchPointer** (**Pointer**< T, **Lock** > &pointer, const bool &keepLock=false)
- virtual bool **_equalsCPointer** (const T *pointer) const
- virtual bool **_equalsPointer** (**Pointer**< T, **Lock** > &pointer) const

Public Member Functions

Constructors & destructor

- **Pointer** ()
- **Pointer** (const T *pointer, const bool &takeOwnership=true)
- **Pointer** (**Pointer**< T, **Lock** > &pointer)
- **Pointer** (const **Pointer**< T, **Lock** > &pointer)
- virtual ~**Pointer** ()

Operators

- virtual const **Pointer**< T, **Lock** > & **operator=** (const T *pointer)
- virtual const **Pointer**< T, **Lock** > & **operator=** (**Pointer**< T, **Lock** > &pointer)
- virtual const **Pointer**< T, **Lock** > & **operator=** (const **Pointer**< T, **Lock** > &pointer)
- virtual T *& **operator** → ()
- virtual bool **operator==** (const T *pointer)
- virtual bool **operator==** (**Pointer**< T, **Lock** > &pointer)
- virtual bool **operator!=** (const T *pointer)
- virtual bool **operator!=** (**Pointer**< T, **Lock** > &pointer)

Static Public Attributes

Class constants

- static const **Pointer**< T, **Lock** > ∗ **NULLPTR**

Protected Attributes

Variables

- **Meta** ∗ **mMeta**

Classes

- struct **Meta**

4.25.2 Constructor & Destructor Documentation

4.25.2.1 Pointer () [inline]

Create a NULL pointer.

4.25.2.2 Pointer (const T ∗ *pointer*, const bool & *takeOwnership* = true) [inline, explicit]

Create smart pointer from a C style pointer.

Parameters:

← ***pointer*** C style pointer

← ***takeOwnership*** Takes ownership of the assigned object. It will be deleted when smart pointer dies.

4.25.2.3 Pointer (Pointer< T, Lock > & *pointer*) [inline]

Create smart pointer from an another smart pointer. (Copy constructor)

Parameters:

→ ***pointer*** Another smart pointer to assign.

4.25.2.4 Pointer (const Pointer< T, Lock > & *pointer*) [inline]

Create smart pointer from an another const smart pointer. (Constantant copy constructor)

Parameters:

← ***pointer*** Another smart pointer to assign.

4.25.2.5 ∼Pointer () [virtual]

The destructor. This class has virtual destructor.

4.25.3 Member Function Documentation

4.25.3.1 void _assignCPointer (const T * *pointer*, const bool & *takeOwnership* = `true`) `[inline, protected, virtual]`

Assign a C style pointer.

Parameters:

← ***pointer*** C style pointer

← ***takeOwnership*** Takes ownership of the assigned object. It will be deleted when smart pointer dies.

4.25.3.2 void _assignPointer (Pointer< T, Lock > & *pointer*, const bool & *keepLock* = `false`) `[inline, protected, virtual]`

Assign another smart pointer.

Parameters:

→ ***pointer*** Another smart pointer to assign.

← ***keepLock*** Keep the pointer locked.

4.25.3.3 void _deletePointer (const bool & *force* = `false`, const bool & *keepLock* = `false`) `[inline, protected, virtual]`

Delete the pointer. Decrease reference counter, relese lock, free memory if needed.

Parameters:

← ***force*** If false, the object will only be deallocated when one it has one reference.

← ***keepLock*** Keep the pointer locked.

4.25.3.4 bool _equalsCPointer (const T * *pointer*) const `[inline, protected, virtual]`

Test indentity.

Parameters:

← ***pointer*** C style pointer

Returns:

True if the data in parameter is identical to the smart pointer.

4.25.3.5 bool _equalsPointer (Pointer< T, Lock > & *pointer*) const `[inline, protected, virtual]`

Test indentity.

Parameters:

→ ***pointer*** Another smart pointer to assign.

Returns:

True if the data in parameter is identical to the smart pointer.

4.25.3.6 void _switchPointer (Pointer< T, Lock > & *pointer*, const bool & *keepLock* = false) [inline, protected, virtual]

Switch to another smart pointer. Helps changing reference under continous lock.

Parameters:

→ ***pointer*** Another smart pointer to assign.

← ***keepLock*** Keep the pointer locked.

4.25.3.7 void assignWithLock (Pointer< T, Lock > & *pointer*) [inline, virtual]

Assign another smart pointer with keeping it locked.

Parameters:

→ ***pointer*** Another smart pointer to assign.

4.25.3.8 bool getLocked () const [inline, virtual]

Return true, if the object is locked.

Returns:

True if the referenced object is locked.

4.25.3.9 T * getPtr () [inline, virtual]

Return a C style pointer to the referenced object.

Returns:

Pointer(p. 107) to the referenced object.

4.25.3.10 bool isNotNull () const [inline, virtual]

Return true if the pointer does reference something.

Returns:

True if the pointer does reference something.

4.25.3.11 bool isNull () const [inline, virtual]

Return true if the pointer does not reference anything.

Returns:

True if the pointer does not reference anything.

4.25.3.12 void kill (const bool & *force* = false) [inline, virtual]

Force deallocating referenced object.

Parameters:
← ***force*** If false, the object will only be deallocated when one it has one reference.

4.25.3.13 void lock () [inline, virtual]

Lock(p. 74) the referenced object.

4.25.3.14 bool operator!= (Pointer< T, Lock > & *pointer*) [inline, virtual]

Equality test operator.

Parameters:
→ ***pointer*** Another smart pointer to test.

Returns:
True if the referenced objects are not identical.

4.25.3.15 bool operator!= (const T * *pointer*) [inline, virtual]

Equality test operator on a C style pointer.

Parameters:
← ***pointer*** C style pointer

Returns:
True if the referenced object is not identical to the parameter.

4.25.3.16 T *& operator → () [inline, virtual]

Get referenced object.

Returns:
Referenced object.

4.25.3.17 const Pointer< T, Lock > & operator= (const Pointer< T, Lock > & *pointer*) [inline, virtual]

Assign another smart pointer (const version).

Parameters:
← ***pointer*** Another smart pointer to assign.

Returns:
Self reference.

4.25.3.18 const Pointer< T, Lock > & operator= (Pointer< T, Lock > & *pointer*) [inline, virtual]

Assign another smart pointer.

Parameters:
→ ***pointer*** Another smart pointer to assign.

Returns:
Self reference.

4.25.3.19 const Pointer< T, Lock > & operator= (const T * *pointer*) [inline, virtual]

Assign a C style pointer.

Parameters:
← ***pointer*** C style pointer

Returns:
Self reference.

4.25.3.20 bool operator== (Pointer< T, Lock > & *pointer*) [inline, virtual]

Equality test operator.

Parameters:
→ ***pointer*** Another smart pointer to test.

Returns:
True if the referenced objects are identical.

4.25.3.21 bool operator== (const T * *pointer*) [inline, virtual]

Equality test operator on a C style pointer.

Parameters:
← ***pointer*** C style pointer

Returns:
True if the referenced object is identical to the parameter.

4.25.3.22 void reference (const T * *pointer*) [inline, virtual]

Take reference from a C style pointer (does not delete referenced object when smart pointer dies).

Parameters:
← ***pointer*** C style pointer

4.25.3.23 void setNull (const bool & *force* = false) [inline, virtual]

Set the referenced object to null without trying to deallocate the actual object.

Remarks:

The typical usage of this method the external deallocating a pointer, like fclose.

Parameters:
← ***force*** If false, the reference will only be set to null when one it has one reference.

4.25.3.24 bool trylock () [inline, virtual]

Try locking referenced object.

Returns:
True if the locking was successful.

4.25.3.25 void unlock () [inline, virtual]

Unlock the referenced object.

4.25.4 Member Data Documentation

4.25.4.1 Meta* mMeta [protected]

Meta(p. 114) field for the referenced object.

Remarks:
This is own attribute of this class.

4.25.4.2 const Pointer< T, Lock > * NULLPTR [static]

Initial value:

```
Pointer < T, DummyLock > ()
```

Null pointer constant.

Remarks:
This is own attribute of this class.

4.26 Pointer::Meta Struct Reference

4.26.1 Detailed Description

template<typename T, class Lock> struct ParCompMark::Pointer< T, Lock >::Meta

Meta(p. 114) field type for the referenced object.

Public Attributes

- T * **ptr**
- bool **dead**
- **u32 usage**
- bool **ownMemory**
- **Lock lock**

4.26.2 Member Data Documentation

4.26.2.1 bool dead

Indicates that the object is deleted

4.26.2.2 Lock lock

Lock(p. 74) for the object

4.26.2.3 bool ownMemory

Indicates that the referenced memory should deallocated by the smart pointer

4.26.2.4 T* ptr

Pointer(p. 107) to the referenced object

4.26.2.5 u32 usage

Usage counter

4.27 Process Class Reference

Inherits **Thread**, and **Name**.

4.27.1 Detailed Description

Entity that composite or render something.

Methods

- virtual void **init** ()
- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual void **openRenderWindow** (const std::string caption="PCM Framework", const std::string &displayName="", const bool &fullScreen=true, const **u32** &colourDepth=0, const **u32** &width=**GLXRenderWindow::MAXIMALSIZE**, const **u32** &height=**GLXRenderWindow::MAXIMALSIZE**, const **s32** &left=**GLXRenderWindow::CENTERED**, const **s32** &top=**GLXRenderWindow::CENTERED**, const **u32** &fsaaSamples=0)
- virtual void **actualizeRenderWindow** ()
- virtual void **displayFrameletIcon** ()
- virtual void **initProcess** ()
- virtual void **runningProcess** ()
- virtual void **stopProcess** ()
- virtual void **task** ()

Public Types

- typedef **Pointer**< **Process**, **Mutex** > **Pointer**
- enum **ProcessType** { **COMPOSITE**, **RENDER** }

Public Member Functions

Constructors & destructor

- **Process** (const std::string &name, **Node** *parent)
- virtual ~**Process** ()

Getters & setters

- const **Process::ProcessType** & **getProcessType** () const
- **Node** * **getParent** () const
- **GLXRenderWindow::Pointer** & **getRenderWindow** ()
- const std::string & **getConfig** () const
- void **setConfig** (const std::string &config)
- **Buffer::Pointer** & **getBuffer** ()
- **Buffer::Pointer** & **getFramelet** ()
- const bool & **getInitialized** () const
- const PCid & **getFrameID** () const
- const **Real** & **getStartTime** () const
- void **setStartTime** (const **Real** &starttime)
- const GLuint & **getOutputTexture** () const

- const bool & **getOutputTextureCreated** () const
- const PCid & **getStopID** () const
- const bool & **getStop** () const
- **OutputNode::Pointer** & **getOutputDocument** ()

Protected Attributes

Variables

- **ProcessType mProcessType**
- **Node** ∗ **mParent**
- **GLXRenderWindow::Pointer mRenderWindow**
- std::string **mConfig**
- **Context::Pointer mContext**
- **SqVM::Pointer mSqVM**
- **Buffer::Pointer mBuffer**
- **Buffer::Pointer mFramelet**
- bool **mInitialized**
- PCid **mFrameID**
- **Real mStartTime**
- GLuint **mOutputTexture**
- bool **mOutputTextureCreated**
- PCid **mStopID**
- bool **mStop**
- **OutputNode::Pointer mOutputDocument**

4.27.2 Member Typedef Documentation

4.27.2.1 typedef Pointer< Process, Mutex > Pointer

Type for pointer on this class.

4.27.3 Member Enumeration Documentation

4.27.3.1 enum ProcessType

The control type of PC context (Local, Global).

Enumerator:

COMPOSITE **Process**(p. 115) composite and provides frame and requires framelets.

RENDER **Process**(p. 115) provides framelets.

4.27.4 Constructor & Destructor Documentation

4.27.4.1 Process (const std::string & *name*, Node ∗ *parent*)

Creates a process with a specified name.

Parameters:

← ***name*** **Name**(p. 83) of the host.

← ***parent*** Parent node

4.27.4.2 ~Process () [virtual]

The destructor. This class has virtual destructor.

4.27.5 Member Function Documentation

4.27.5.1 void actualizeRenderWindow () [virtual]

Actualize GLX render window on PC information (frame sizes etc).

4.27.5.2 void displayFrameletIcon () [virtual]

Display small icon on to top-left corner of the GLX window indicating the frame/framelet relationship for this process.

4.27.5.3 void finalize () [virtual]

Some finitiate step as thread.

Reimplemented from **Thread** (p. 136).

4.27.5.4 Buffer::Pointer & getBuffer () [inline]

Getter of mBuffer. Returns value of mBuffer.

Returns:
The value of mBuffer

4.27.5.5 const std::string & getConfig () const [inline]

Getter of mConfig. Returns value of mConfig.

Returns:
The value of mConfig

4.27.5.6 const PCid & getFrameID () const [inline]

Getter of mFrameID. Returns value of mFrameID.

Returns:
The value of mFrameID

4.27.5.7 Buffer::Pointer & getFramelet () [inline]

Getter of mFramelet. Returns value of mFramelet.

Returns:
The value of mFramelet

4.27.5.8 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:
The value of mInitialized

4.27.5.9 OutputNode::Pointer & getOutputDocument () [inline]

Getter of mOutputDocument. Returns value of mOutputDocument.

Returns:
The value of mOutputDocument

4.27.5.10 const GLuint & getOutputTexture () const [inline]

Getter of mOutputTexture. Returns value of mOutputTexture.

Returns:
The value of mOutputTexture

4.27.5.11 const bool & getOutputTextureCreated () const [inline]

Getter of mOutputTextureCreated. Returns value of mOutputTextureCreated.

Returns:
The value of mOutputTextureCreated

4.27.5.12 Node * getParent () const [inline]

Getter of mParent. Returns value of mParent.

Returns:
The value of mParent

4.27.5.13 const Process::ProcessType & getProcessType () const [inline]

Getter of mProcessType. Returns value of mProcessType.

Returns:
The value of mProcessType

4.27.5.14 GLXRenderWindow::Pointer & getRenderWindow () [inline]

Getter of mRenderWindow. Returns value of mRenderWindow.

Returns:
The value of mRenderWindow

4.27.5.15 const Real & getStartTime () const [inline]

Getter of mStartTime. Returns value of mStartTime.

Returns:
The value of mStartTime

4.27.5.16 const bool & getStop () const [inline]

Getter of mStop. Returns value of mStop.

Returns:
The value of mStop

4.27.5.17 const PCid & getStopID () const [inline]

Getter of mStopID. Returns value of mStopID.

Returns:
The value of mStopID

4.27.5.18 void init () [virtual]

Init the process.

4.27.5.19 void initialize () [virtual]

Some init step as thread.

Reimplemented from **Thread** (p. 138).

4.27.5.20 void initProcess () [virtual]

This is an initializing method. Called at the start of the working process. Starts a Squirrel VM and executes the initialization code.

4.27.5.21 void openRenderWindow (const std::string *caption* = "PCM Framework", const std::string & *displayName* = "", const bool & *fullScreen* = true, const u32 & *colourDepth* = 0, const u32 & *width* = GLXRenderWindow::MAXIMALSIZE, const u32 & *height* = GLXRenderWindow::MAXIMALSIZE, const s32 & *left* = GLXRenderWindow::CENTERED, const s32 & *top* = GLXRenderWindow::CENTERED, const u32 & *fsaaSamples* = 0) [virtual]

Open GLX render window for rendering.

Parameters:

← ***caption*** Window caption

← ***displayName*** X display name

← ***fullScreen*** The window appears in full screen mode

← ***colourDepth*** Colour depth of the window

← ***width*** Horizontal size

← ***height*** Vertical size

← ***left*** Horizontal position

← ***top*** Vertical position

← ***fsaaSamples*** Number of fullscreen antialasing samples (Do not use FSAA samples other than 0 now with nVidia cards!)

4.27.5.22 void runningProcess () [virtual]

This method achieves the "real functionality" of the process. Called at every frame. Starts a Squirrel VM and executes the initialization code.

4.27.5.23 void setConfig (const std::string & *config*) [inline]

Setter of mConfig. Sets value of mConfig.

Parameters:

← ***config*** The value of mConfig

4.27.5.24 void setStartTime (const Real & *starttime*) [inline]

Setter of mStartTime. Sets value of mStartTime.

Parameters:

← ***starttime*** The value of mStartTime

4.27.5.25 void stopProcess () [virtual]

Stop the process. Shyncronize compositing.

4.27.5.26 void task () [protected, virtual]

What we do during rendering/compositing.

Reimplemented from **Thread** (p. 139).

4.27.6 Member Data Documentation

4.27.6.1 Buffer::Pointer mBuffer [protected]

Memory buffer for output (if has). It is lockable.

Remarks:
This is own attribute of this class.

4.27.6.2 std::string mConfig [protected]

Config string.

Remarks:
This is own attribute of this class.

4.27.6.3 Context::Pointer mContext [protected]

Context(p. 30) description.

Remarks:
This is own attribute of this class.

4.27.6.4 PCid mFrameID [protected]

The current frame id. It sets by PC.

Remarks:
This is own attribute of this class.

4.27.6.5 Buffer::Pointer mFramelet [protected]

Memory buffers for framelets (if has).

Remarks:
This is own attribute of this class.

4.27.6.6 bool mInitialized [protected]

The process is initialized.

Remarks:
This is own attribute of this class.

4.27.6.7 OutputNode::Pointer mOutputDocument [protected]

Process(p. 115) output document.

Remarks:
This is own attribute of this class.

4.27.6.8 GLuint mOutputTexture [protected]

OpenGL texture for displaying output.

Remarks:
This is own attribute of this class.

4.27.6.9 bool mOutputTextureCreated [protected]

Indicates that the output texture is created (using glTexImage2D) and further operations can be faster (glTexSubImage2D).

Remarks:
This is own attribute of this class.

4.27.6.10 Node* mParent [protected]

Parent node of the process. (Parent reference is standard pointer to avoid circular reference)

Remarks:
This attribute references an attribute.

4.27.6.11 ProcessType mProcessType [protected]

Composite or render process.

Remarks:
This is own attribute of this class.

4.27.6.12 GLXRenderWindow::Pointer mRenderWindow [protected]

Renderwindow of this process. Each process can have zero or one renderwindow.

Remarks:
This is own attribute of this class.

4.27.6.13 SqVM::Pointer mSqVM [protected]

Squirrel virtual machine.

Remarks:
This is own attribute of this class.

4.27.6.14 Real mStartTime [protected]

Local (uncorrected) time at the start of the benchmark for this process.

Remarks:
This is own attribute of this class.

4.27.6.15 bool mStop [protected]

Indicates that process must be stopped.

Remarks:
This is own attribute of this class.

4.27.6.16 PCid mStopID [protected]

Indicates the stopping frame ID.

Remarks:
This is own attribute of this class.

4.28 Singleton Class Template Reference

4.28.1 Detailed Description

template<typename T> class ParCompMark::Singleton< T >

Template class for creating single-instance global classes.

Public Member Functions

Constructors & destructor

- **Singleton** ()
- virtual ~**Singleton** ()

Static Public Member Functions

Class methods

- static T * **getInstance** ()
- static void **createInstance** ()
- static void **destroyInstance** ()

Static Protected Attributes

Class variables

- static T * **mInstance**

4.28.2 Constructor & Destructor Documentation

4.28.2.1 Singleton ()

Default constructor.

4.28.2.2 ~Singleton () [virtual]

The destructor. This class has virtual destructor.

4.28.3 Member Function Documentation

4.28.3.1 void createInstance () [inline, static]

Create singleton instance.

4.28.3.2 void destroyInstance () [inline, static]

Destroy singleton instance.

4.28.3.3 T * getInstance () [inline, static]

Gets the instance of the singleton class. You have to construct singleton object (createInstance) before calling this method.

Returns:
Instance of the class

4.28.4 Member Data Documentation

4.28.4.1 T * mInstance [static, protected]

Class level instance.

Remarks:
This is own attribute of this class.

4.29 SqVM Class Reference

Inherits **Name**.

4.29.1 Detailed Description

Squirrel virtual machine.

Getters & setters

- const bool & **getError** () const
- const bool & **getInitialized** () const
- static const **u32** & **getStringBufferSize** ()

Methods

- virtual void **runScriptFromFile** (const std::string &filename, const std::string &mainMethod="main")
- virtual void **runScriptFromString** (const std::string &scriptString, const std::string &scriptName="", const std::string &mainMethod="main")
- virtual void **runScriptByName** (const std::string &scriptName)
- virtual void **activate** ()
- virtual void **deactivate** ()
- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual **SqVM::Script::Pointer createScript** (const std::string &scriptName="", const bool &dynamic=false, const std::string &mainMethod="main")
- virtual **SqVM::Script::Pointer findScript** (const std::string &scriptName)
- virtual **SqVM::Script::Pointer findOrAddScript** (const std::string &scriptName, const bool &dynamic=false, const std::string &mainMethod="main")
- virtual void **compileAndExecuteScript** (**SqVM::Script::Pointer** &script)

Class constants

- static const std::string **NOMAINMETHOD** = "null"
- static const **u32 mStringBufferSize** = 32768

Public Types

- typedef **Pointer**< **SqVM**, **Mutex** > **Pointer**

Public Member Functions

Constructors & destructor

- **SqVM** (const std::string &name)
- virtual ~**SqVM** ()

Protected Types

- typedef **ParCompMark::SqVM::Script Script**
- enum **ScriptState** { **UNCOMPILED**, **COMPILED**, **EXECUTED** }

Static Protected Member Functions

Class methods

- static void **printFunction** (::HSQUIRRELVM sqVM, const SQChar ∗s,...)

Protected Attributes

Variables

- **SqVM::Pointer mThis**
- SquirrelVMSys ∗ **mSquirrelVMSys**
- bool **mError**
- bool **mInitialized**

Static Protected Attributes

Class variables

- static **SqVM::Pointer mCurrentVM**
- static **Container**< **SqVM::Script**, **Mutex** >::**Pointer mScripts**
- static char **mStringBuffer** [32768] = ""

Classes

- struct **Script**

4.29.2 Member Typedef Documentation

4.29.2.1 typedef Pointer< SqVM, Mutex > Pointer

Type for pointer on this class.

4.29.2.2 typedef struct ParCompMark::SqVM::Script Script [protected]

Struct for script attributes.

4.29.3 Member Enumeration Documentation

4.29.3.1 enum ScriptState [protected]

Definitions of script states.

Enumerator:

UNCOMPILED The script is not compiled

COMPILED The script is compiled

EXECUTED The script is executed (main method called)

4.29.4 Constructor & Destructor Documentation

4.29.4.1 SqVM (const std::string & *name*)

Create Squirrel virtual machine.

Parameters:

← ***name*** **Name**(p. 83) of the virtual machine.

4.29.4.2 ~SqVM () `[virtual]`

The destructor. This class has virtual destructor.

4.29.5 Member Function Documentation

4.29.5.1 void activate () `[protected, virtual]`

Activate this VM. This virtual machine will be selected to operate.

4.29.5.2 void compileAndExecuteScript (SqVM::Script::Pointer & *script*) `[protected, virtual]`

Compile and execute the script.

Parameters:

→ ***script*** **Script**(p. 133) handle.

4.29.5.3 SqVM::Script::Pointer createScript (const std::string & *scriptName* = "", const bool & *dynamic* = `false`, const std::string & *mainMethod* = "main") `[protected, virtual]`

Create script object with the specified name and entry point. The dynamic flag is also set, and the returned script object is locked by default.

Parameters:

← ***scriptName*** **Name**(p. 83) of the script.

← ***dynamic*** Dynamic flag.

← ***mainMethod*** **Script**(p. 133) entry method name. Giving **SqVM::NOMAINMETHOD**(p. 132) as mainMethod indicates that no main method.

Returns:

Pointer(p. 107) to the created script object.

4.29.5.4 void deactivate () [protected, virtual]

Deactivate this VM. This virtual machine will go to sleep and let other VMs to be activated.

4.29.5.5 void finalize () [protected, virtual]

Finalize virtual machine.

4.29.5.6 SqVM::Script::Pointer findOrAddScript (const std::string & *scriptName*, const bool & *dynamic* = false, const std::string & *mainMethod* = "main") [protected, virtual]

If the script exists with the given name then returns, if not then adds it to the internal class level container and return it.

Parameters:

← ***scriptName*** **Name**(p. 83) of the searched script.

← ***dynamic*** Dynamic flag; if the script is have to be created this flag is set.

← ***mainMethod*** **Script**(p. 133) entry method name. Giving **SqVM::NOMAINMETHOD**(p. 132) as mainMethod indicates that no main method.

Returns:

Pointer(p. 107) of the found or the newly created script object.

4.29.5.7 SqVM::Script::Pointer findScript (const std::string & *scriptName*) [protected, virtual]

Finds a prevously added script.

Parameters:

← ***scriptName*** **Name**(p. 83) of a dynamic script name or filename.

Returns:

Pointer(p. 107) of the found script object.

4.29.5.8 const bool & getError () const [inline]

Getter of mError. Returns value of mError.

Returns:

The value of mError

4.29.5.9 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:

The value of mInitialized

4.29.5.10 const u32 & getStringBufferSize () [inline, static]

Getter of mStringBufferSize. Returns value of mStringBufferSize.

Returns:
The value of mStringBufferSize

4.29.5.11 void initialize () [protected, virtual]

Initialize virtual machine.

4.29.5.12 void printFunction (::HSQUIRRELVM *sqVM*, const SQChar * *s*, ...) [static, protected]

The print function of the virtual machine. This function is used by the built-in function 'print()' to output text.

Parameters:
← ***sqVM*** Squirrel VM
← ***s*** Format string
← **...** Additional parameters

4.29.5.13 void runScriptByName (const std::string & *scriptName*) [virtual]

Execute a previously stored script.

Parameters:
← ***scriptName*** **Name**(p. 83) of a dynamic script name or filename.

4.29.5.14 void runScriptFromFile (const std::string & *filename*, const std::string & *mainMethod* = "main") [virtual]

Execute script loaded from file. Giving **SqVM::NOMAINMETHOD**(p. 132) as mainMethod indicates that no main method.

Parameters:
← ***filename*** **Script**(p. 133) filename
← ***mainMethod*** **Script**(p. 133) entry method name. Giving **SqVM::NOMAINMETHOD**(p. 132) as mainMethod indicates that no main method.

4.29.5.15 void runScriptFromString (const std::string & *scriptString*, const std::string & *scriptName* = "", const std::string & *mainMethod* = "main") [virtual]

Execute script from the given string. If the scriptName is not empty the VM store this script into the class level store for better performance. Giving **SqVM::NOMAINMETHOD**(p. 132) as mainMethod indicates that no main method.

Parameters:

← ***scriptString*** **Script**(p. 133) string

← ***scriptName*** **Name**(p. 83) of the script. If it is not empty the VM store this script into the class level store for better performance.

← ***mainMethod*** **Script**(p. 133) entry method name. Giving **SqVM::NOMAINMETHOD**(p. 132) as mainMethod indicates that no main method.

4.29.6 Member Data Documentation

4.29.6.1 SqVM::Pointer mCurrentVM [static, protected]

Currently active Squirrel Virtual Machine. Only one VM can be active.

Remarks:

This is own attribute of this class.

4.29.6.2 bool mError [protected]

An error occured on this virtual machine.

Remarks:

This is own attribute of this class.

4.29.6.3 bool mInitialized [protected]

The virtual machine is initialized.

Remarks:

This is own attribute of this class.

4.29.6.4 Container< SqVM::Script, Mutex >::Pointer mScripts [static, protected]

Class level script container. Many VMs can use the same script objects since only one VM can be active at the same time. Dynamic script can be also stored here when they have a proper name.

Remarks:

This is own attribute of this class.

4.29.6.5 SquirrelVMSys* mSquirrelVMSys [protected]

Squirrel virtual machine.

Remarks:

This attribute references an attribute.

4.29.6.6 char mStringBuffer = "" [static, protected]

Common C style string buffer of printFunction.

Remarks:
This is own attribute of this class.

4.29.6.7 const u32 mStringBufferSize = 32768 [static, protected]

Size of mStringBuffer.

Remarks:
This is own attribute of this class.

4.29.6.8 SqVM::Pointer mThis [protected]

Smart pointer on this object.

Remarks:
This is own attribute of this class.

4.29.6.9 const std::string NOMAINMETHOD = "null" [static]

Constant for indicating that no main method are defined for a script.

Remarks:
This is own attribute of this class.

4.30 SqVM::Script Struct Reference

4.30.1 Detailed Description

Struct for script attributes.

Public Types

- typedef **ParCompMark::Pointer< Script, DummyLock > Pointer**

Public Attributes

- bool **dynamic**
- std::string **name**
- std::string **scriptString**
- SquirrelObject **scriptObject**
- bool **compiled**
- std::string **mainMethod**

4.30.2 Member Typedef Documentation

4.30.2.1 typedef ParCompMark::Pointer< Script, DummyLock > Pointer

Smart pointer on this struct

4.30.3 Member Data Documentation

4.30.3.1 bool compiled

The loaded script

4.30.3.2 bool dynamic

Indicate that the script is dynamic otherwise loaded from a file

4.30.3.3 std::string mainMethod

Name(p. 83) of the entry method

4.30.3.4 std::string name

Name(p. 83) of the script (filename or dynamic script name)

4.30.3.5 SquirrelObject scriptObject

Squirrel script object

4.30.3.6 std::string scriptString

Script(p. 133) containing the source code. This is only set at dynamic scripts

4.31 Thread Class Reference

Inherited by **HandleClient**, **Network**, and **Process**.

4.31.1 Detailed Description

This is a unique thread class.

Methods

- virtual void **initThread** (const **u32** &iterationNumber=0, const **u32** &expectedFPS=0, const bool &joinable=false, const bool &waitThread=false)
- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual void **startThread** ()
- virtual void **joinThread** ()
- virtual void **shutDownThread** ()
- virtual void **stopThread** ()
- virtual void **thread** ()
- virtual bool **iteration** ()
- virtual void **task** ()

Class methods

- static void **yield** ()
- static **Real getUSTime** ()
- static void * **entryPoint** (void *arg)

Public Member Functions

Constructors & destructor

- **Thread** (const std::string &name)
- virtual ~**Thread** ()

Getters & setters

- const std::string & **getThreadName** () const
- const bool & **getStopRequested** () const
- const bool & **getJoinable** () const
- const bool & **getWaitThread** () const
- const bool & **getRunning** () const
- const **u32** & **getCurrentFPS** () const
- const **u32** & **getExpectedFPS** () const
- const **u32** & **getIterationNumber** () const

Protected Attributes

Variables

- std::string **mThreadName**
- bool **mStopRequested**
- bool **mJoinable**
- bool **mWaitThread**
- pthread_t **mThread**
- bool **mRunning**
- **u32 mCurrentFPS**
- **u32 mExpectedFPS**
- **u32 mIterationNumber**

4.31.2 Constructor & Destructor Documentation

4.31.2.1 Thread (const std::string & *name*)

Thread(p. 135) constructor.

Parameters:
← ***name*** **Name**(p. 83) of the thread.

4.31.2.2 ~Thread () [virtual]

The destructor. This class has virtual destructor.

4.31.3 Member Function Documentation

4.31.3.1 void * entryPoint (void * *arg*) [static, protected]

Static entry point for the thread.

Parameters:
← ***arg*** This is only for thread.

Returns:
This is only for thread.

4.31.3.2 void finalize () [virtual]

Some finitiate step as thread.

Reimplemented in **Process** (p. 117).

4.31.3.3 const u32 & getCurrentFPS () const [inline]

Getter of mCurrentFPS. Returns value of mCurrentFPS.

Returns:
The value of mCurrentFPS

4.31.3.4 const u32 & getExpectedFPS () const [inline]

Getter of mExpectedFPS. Returns value of mExpectedFPS.

Returns:
The value of mExpectedFPS

4.31.3.5 const u32 & getIterationNumber () const [inline]

Getter of mIterationNumber. Returns value of mIterationNumber.

Returns:
The value of mIterationNumber

4.31.3.6 const bool & getJoinable () const [inline]

Getter of mJoinable. Returns value of mJoinable.

Returns:
The value of mJoinable

4.31.3.7 const bool & getRunning () const [inline]

Getter of mRunning. Returns value of mRunning.

Returns:
The value of mRunning

4.31.3.8 const bool & getStopRequested () const [inline]

Getter of mStopRequested. Returns value of mStopRequested.

Returns:
The value of mStopRequested

4.31.3.9 const std::string & getThreadName () const [inline]

Getter of mThreadName. Returns value of mThreadName.

Returns:
The value of mThreadName

4.31.3.10 Real getUSTime () [inline, static]

Get system time in us.

Returns:
System time in us.

4.31.3.11 const bool & getWaitThread () const [inline]

Getter of mWaitThread. Returns value of mWaitThread.

Returns:
The value of mWaitThread

4.31.3.12 void initialize () [virtual]

Some init step as thread.

Reimplemented in **Process** (p. 119).

4.31.3.13 void initThread (const u32 & *iterationNumber* = 0, const u32 & *expectedFPS* = 0, const bool & *joinable* = false, const bool & *waitThread* = false) [virtual]

Start thread.

Parameters:
← ***iterationNumber*** How much is the task running. Zero means nan.

← ***expectedFPS*** Expected FPS.

← ***joinable*** Joinable?

← ***waitThread*** Wait when destroy class?

4.31.3.14 bool iteration () [inline, protected, virtual]

One iteration step. This method calls the overridable task method.

Returns:
If true, the thread is stopped.

4.31.3.15 void joinThread () [virtual]

Join thread. Wait for its ending.

Remarks:
!!! You can join if the thread will stop (mIterationNumber != 0 or mStopRequested = true).

4.31.3.16 void shutDownThread () [virtual]

Immediately stop the thread.

4.31.3.17 void startThread () [virtual]

Start thread.

4.31.3.18 void stopThread () [virtual]

Request thread shut down.

4.31.3.19 void task () [protected, virtual]

It is the task of the thread. It have to be overiden.

Remarks:
This method should be abstact. It is not, because of unit testing of this class.

Reimplemented in **HandleClient** (p. 68), **Network** (p. 91), and **Process** (p. 121).

4.31.3.20 void thread () [protected, virtual]

This is the thread method containing one or more (or inifinite) iterations.

4.31.3.21 void yield () [inline, static]

Yield current thread.

4.31.4 Member Data Documentation

4.31.4.1 u32 mCurrentFPS [protected]

The thread is running.

Remarks:
This is own attribute of this class.

4.31.4.2 u32 mExpectedFPS [protected]

The thread is running. Zero means no limit.

Remarks:
This is own attribute of this class.

4.31.4.3 u32 mIterationNumber [protected]

Iteration number of task. Zero means nan.

Remarks:
This is own attribute of this class.

4.31.4.4 bool mJoinable [protected]

The thread is created as joinable or not.

Remarks:
This is own attribute of this class.

4.31.4.5 bool mRunning [protected]

The thread is running.

Remarks:
This is own attribute of this class.

4.31.4.6 bool mStopRequested [protected]

The thread is requested to stop.

Remarks:
This is own attribute of this class.

4.31.4.7 pthread_t mThread [protected]

Thread(p. 135) handle.

Remarks:
This is own attribute of this class.

4.31.4.8 std::string mThreadName [protected]

Name(p. 83) of the thread.

Remarks:
This is own attribute of this class.

4.31.4.9 bool mWaitThread [protected]

When class destructor call, wait thread or not.

Remarks:
This is own attribute of this class.

4.32 Timer Class Reference

4.32.1 Detailed Description

Collection of time handling methods.

Static Public Member Functions

Class methods

- static **Real getSystemTime** ()
- static void **sleep** (const **Real** &time)

Static Public Attributes

Class constants

- static const **Real EPSILONDELAY** = 0.000001

4.32.2 Member Function Documentation

4.32.2.1 Real getSystemTime () [inline, static]

Return time in secondes.

Returns:
Time in seconds

4.32.2.2 void sleep (const Real & *time*) [inline, static]

Sleep process for the specified seconds.

Parameters:
← ***time*** Time to sleep

4.32.3 Member Data Documentation

4.32.3.1 const Real EPSILONDELAY = 0.000001 [static]

Constant for minimal time delay (1us).

Remarks:
This is own attribute of this class.

4.33 XDisplay Class Reference

4.33.1 Detailed Description

Class that encapsulates an X Display.

Getters & setters

- const std::string & **getDisplayName** () const
- ::Display * **getDisplay** () const
- const bool & **getInitialized** () const
- const **u32** & **getWidth** () const
- const **u32** & **getHeight** () const
- static const bool & **getXMTInitialized** ()
- static const bool & **getXMTSupported** ()

Methods

- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual **s32** **findBestVisual** (const **s32** &screenNumber, const **s32** &multiSample=**XDisplay::IGNOREMULTISAMPLE**)
- virtual void **getVisualAttribs** (XVisualInfo *vInfo, **VisualAttribs** &attribs)
- virtual void **initializeMT** ()

Public Types

- typedef **Pointer**< **XDisplay**, **Mutex** > **Pointer**

Public Member Functions

Constructors & destructor

- **XDisplay** (const std::string &displayName="")
- virtual ~**XDisplay** ()

Static Public Attributes

Class constants

- static const **s32** **IGNOREMULTISAMPLE** = -1
- static const **s32** **UNKNOWNDIMENSION** = 0

Protected Attributes

Variables

- std::string **mDisplayName**
- ::Display * **mDisplay**
- bool **mInitialized**
- **u32 mWidth**
- **u32 mHeight**

Static Protected Attributes

Class variables

- static bool **mXMTInitialized** = false
- static bool **mXMTSupported** = false

Classes

- struct **VisualAttribs**

4.33.2 Member Typedef Documentation

4.33.2.1 typedef Pointer< XDisplay, Mutex > Pointer

Type for pointer on this class.

4.33.3 Constructor & Destructor Documentation

4.33.3.1 XDisplay (const std::string & *displayName* = "")

Create an X display.

Parameters:
← ***displayName*** X display name

4.33.3.2 ~XDisplay () [virtual]

The destructor. This class has virtual destructor.

4.33.4 Member Function Documentation

4.33.4.1 void finalize () [virtual]

Finalize X display.

4.33.4.2 s32 findBestVisual (const s32 & *screenNumber*, const s32 & *multiSample* = XDisplay::IGNOREMULTISAMPLE) [virtual]

Examine all visuals to find the so-called best one. We prefer deepest RGBA buffer with depth, stencil and accum that has no caveats. This will only choose formats with a multisample that equals multisample.

Parameters:

← ***screenNumber*** Screen number to test

← ***multiSample*** Select specific multisample value

Returns:

-1 in case of failure, otherwise a valid visual ID

4.33.4.3 inline::Display * getDisplay () const

Getter of mDisplay. Returns value of mDisplay.

Returns:

The value of mDisplay

4.33.4.4 const std::string & getDisplayName () const [inline]

Getter of mDisplayName. Returns value of mDisplayName.

Returns:

The value of mDisplayName

4.33.4.5 const u32 & getHeight () const [inline]

Getter of mHeight. Returns value of mHeight.

Returns:

The value of mHeight

4.33.4.6 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:

The value of mInitialized

4.33.4.7 void getVisualAttribs (XVisualInfo * *vInfo*, VisualAttribs & *attribs*) [virtual]

Get visual attributes for the specified visual info descriptor

Parameters:

← ***vInfo*** Visual info descriptor

→ ***attribs*** Variable to hold the restrieved attributes

4.33.4.8 const u32 & getWidth () const [inline]

Getter of mWidth. Returns value of mWidth.

Returns:
The value of mWidth

4.33.4.9 const bool & getXMTInitialized () [inline, static]

Getter of mXMTInitialized. Returns value of mXMTInitialized.

Returns:
The value of mXMTInitialized

4.33.4.10 const bool & getXMTSupported () [inline, static]

Getter of mXMTSupported. Returns value of mXMTSupported.

Returns:
The value of mXMTSupported

4.33.4.11 void initialize () [virtual]

Initialize X display.

4.33.4.12 void initializeMT () [protected, virtual]

Initialize X multithreading, enable Xlib support for concurrent threads. This function must be the first Xlib function a multi-threaded program calls, and it must complete before any other Xlib call is made. (Internally called at the first **XDisplay**(p. 142) initialization.)

4.33.5 Member Data Documentation

4.33.5.1 const s32 IGNOREMULTISAMPLE = -1 [static]

Constant for ignoring multisample.

Remarks:
This is own attribute of this class.

4.33.5.2 ::Display* mDisplay [protected]

Wrapped X Display.

Remarks:
This attribute references an attribute.

4.33.5.3 std::string mDisplayName [protected]

X display name.

Remarks:
This is own attribute of this class.

4.33.5.4 u32 mHeight [protected]

Height of the display in pixels (of the default screen).

Remarks:
This is own attribute of this class.

4.33.5.5 bool mInitialized [protected]

The display is initialized.

Remarks:
This is own attribute of this class.

4.33.5.6 u32 mWidth [protected]

Width of the display in pixels (of the default screen).

Remarks:
This is own attribute of this class.

4.33.5.7 bool mXMTInitialized = false [static, protected]

X multithreading is initialized.

Remarks:
This is own attribute of this class.

4.33.5.8 bool mXMTSupported = false [static, protected]

X multithreading is supported.

Remarks:
This is own attribute of this class.

4.33.5.9 const s32 UNKNOWNDIMENSION = 0 [static]

Constant for unknown display dimension.

Remarks:
This is own attribute of this class.

4.34 XDisplay::VisualAttribs Struct Reference

4.34.1 Detailed Description

Struct for visual attributes.

4.34.2 Member Data Documentation

4.34.2.1 int accumAlphaSize

4.34.2.2 int accumBlueSize

4.34.2.3 int accumGreenSize

4.34.2.4 int accumRedSize

4.34.2.5 int alphaSize

4.34.2.6 int auxBuffers

4.34.2.7 int bitsPerRGB

4.34.2.8 int blueMask

4.34.2.9 int blueSize

4.34.2.10 int bufferSize

4.34.2.11 int colormapSize

4.34.2.12 int depth

4.34.2.13 int depthSize

4.34.2.14 int doubleBuffer

4.34.2.15 int greenMask

4.34.2.16 int greenSize

4.34.2.17 int id

4.34.2.18 int klass

4.34.2.19 int level

4.34.2.20 int numMultisample

4.34.2.21 int numSamples

4.34.2.22 int redMask

4.34.2.23 int redSize

4.34.2.24 int rgba

4.34.2.25 int stencilSize

4.34.2.26 int stereo

4.34.2.27 int supportsGL

4.34.2.28 int transparentAlphaValue

4.34.2.29 int transparentBlueValue

4.34.2.30 int transparentGreenValue

Index