ParCompMark Reference Manual

v0.3

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 **Client**
- class **Cluster**
- class **ConfigOptions**
- class **Container**
- class **Context**
- class **CPU**
- class **DummyLock**
- class **DynLoad**
- class **Exception**
- class **FileSystemManager**
- class **GLXGLContext**
- class **GLXRenderWindow**
- class **GPU**
- class **HandleClient**
- class **Host**
- class **HostInfo**
- class **Lock**
- class **Logger**
- class **Mutex**
- class **Name**
- class **NetClient**

- class **NetServer**
- class **Network**
- class **Node**
- class **OpenGLRenderingEngine**
- class **OutputNode**
- class **Plugin**
- class **PluginManager**
- class **Pointer**
- class **Process**
- class **Renderer**
- class **RendererPlugin**
- class **Singleton**
- class **SqVM**
- class **StringConverter**
- 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
- const bool & **getQuiting** () const
- **OutputNode::Pointer** & **getOutputDocument** ()
- **ConfigOptions::Pointer** & **getConfigOptions** ()
- const std::string & **getLowLevelScript** () const
- const std::string & **getDynamicScript** () const
- **NetServer** ∗ **getServer** () const
- **NetClient** ∗ **getClient** () const
- static const std::string & **getLibrarySearchPath** ()
- static void **setLibrarySearchPath** (const std::string &librarysearchpath)
- 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** ()
- static const bool & **getCompRun** ()

Methods

- virtual void **setupHandlers** () const
- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual void **finalizeCommander** ()
- virtual void **finalizeSoldier** ()
- virtual void **setFromConfigOptions** ()
- virtual bool **startOperation** ()
- virtual void **NetworkTest** ()
- virtual void **writeOutput** ()
- virtual void **quit** ()
- virtual void **start** ()
- virtual void **stop** ()
- virtual void **autoDetection** ()
- virtual void **help** () const
- virtual bool **commanderOperation** ()
- virtual bool **soldierOperation** ()
- virtual bool **textUserInterface** ()
- virtual void **loadLowLevelScript** (const std::string &filename)
- virtual void **loadDynamicScript** (const std::string &filename)
- virtual void **createLowLevelScript** (const std::string &clusterDescription, const std::string &scenarioParameters)

Class constants

- static const std::string **PCAPINUT** = "scripts/framework/pcapi.nut"
- static const std::string **PCIMPLEMENTATIONS** [ ]
- static const **Application::CommandLineOption mCommandLineOptions** [ ]
- static const **u32 mCommandLineOptionCount** = sizeof(**mCommandLineOptions**) / sizeof(**mCommandLineOptions**[0])
- static const std::string **mUsageString** = "ParCompMark [options]"

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 **setLowLevelScriptFile** (const std::string &strarg)

- static void **setDynamicScriptFile** (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 bool **hasEnvironmentVariable** (const std::string &name)
- static std::string **getEnvironmentVariable** (const std::string &name)
- static void **terminateHandler** ()
- static void **terminateHandlerNOP** ()
- static void **unexpectedHandler** ()
- static void **segfaultHandler** (int value)
- static void **interruptHandler** (int value)

Protected Attributes

Variables

- bool **mInitialized**
- bool **mQuiting**
- **OutputNode::Pointer mOutputDocument**
- **ConfigOptions::Pointer mConfigOptions**
- std::string **mLowLevelScript**
- std::string **mDynamicScript**
- **NetServer** ∗ **mServer**
- **NetClient** ∗ **mClient**
- **SqVM::Pointer mDynamicScriptSqVM**

Static Protected Attributes

Class variables

- static std::string **mLibrarySearchPath** = "/usr/lib"
- 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**
- static bool **mCompRun**

Classes

- struct **CommandLineOption**

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Application ()

Default constructor.

4.1.2.2 ~Application () [virtual]

The destructor. This class has virtual destructor.

4.1.3 Member Function Documentation

4.1.3.1 void autoDetection () [virtual]

Auto detection of the cluster.

4.1.3.2 bool commanderOperation () [protected, virtual]

Operation of a commander mode application.

Returns:
Return true on error.

4.1.3.3 void createLowLevelScript (const std::string & *clusterDescription*, const std::string & *scenarioParameters*) [protected, virtual]

Create low-level script from dynamic script, cluster description, and scenario parameters.

Parameters:
← ***clusterDescription*** Squirrel cluster description.
← ***scenarioParameters*** Scenario parameters.

4.1.3.4 void finalize () [virtual]

Finalize the PCM application.

4.1.3.5 void finalizeCommander () [virtual]

Addintional finalization code for commander mode. This method is executed at the beginning of the common **finalize**()(p. 12) method.

4.1.3.6 void finalizeSoldier () [virtual]

Addintional finalization code for soldier mode. This method is executed at the beginning of the common **finalize**()(p. 12) method.

4.1.3.7 NetClient * getClient () const [inline]

Getter of mClient. Returns value of mClient.

Returns:
The value of mClient

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

Getter of mClusterDescription. Returns value of mClusterDescription.

Returns:
The value of mClusterDescription

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

Getter of mCommanderMode. Returns value of mCommanderMode.

Returns:
The value of mCommanderMode

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

Getter of mCompRun. Returns value of mCompRun.

Returns:
The value of mCompRun

4.1.3.11 ConfigOptions::Pointer & getConfigOptions () [inline]

Getter of mConfigOptions. Returns value of mConfigOptions.

Returns:
The value of mConfigOptions

4.1.3.12 const std::string & getDynamicScript () const [inline]

Getter of mDynamicScript. Returns value of mDynamicScript.

Returns:
The value of mDynamicScript

4.1.3.13 std::string getEnvironmentVariable (const std::string & *name*) [static]

Get environment variable.

Parameters:
← ***name*** **Name**(p. 134) of the variable.

Returns:
Value of the specified variable.

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

Getter of mGUIMode. Returns value of mGUIMode.

Returns:
The value of mGUIMode

4.1.3.15 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:
The value of mInitialized

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

Getter of mInput. Returns value of mInput.

Returns:
The value of mInput

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

Getter of mInteractiveParameters. Returns value of mInteractiveParameters.

Returns:
The value of mInteractiveParameters

4.1.3.18 const std::string & getLibrarySearchPath () [inline, static]

Getter of mLibrarySearchPath. Returns value of mLibrarySearchPath.

Returns:
The value of mLibrarySearchPath

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

Getter of mLowLevelMode. Returns value of mLowLevelMode.

Returns:
The value of mLowLevelMode

4.1.3.20 const std::string & getLowLevelScript () const [inline]

Getter of mLowLevelScript. Returns value of mLowLevelScript.

Returns:
The value of mLowLevelScript

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

Getter of mManualClusterDescription. Returns value of mManualClusterDescription.

Returns:
The value of mManualClusterDescription

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

Getter of mOutput. Returns value of mOutput.

Returns:
The value of mOutput

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

Getter of mOutputDocument. Returns value of mOutputDocument.

Returns:
The value of mOutputDocument

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

Getter of mParameters. Returns value of mParameters.

Returns:
The value of mParameters

4.1.3.25 const bool & getQuiting () const [inline]

Getter of mQuiting. Returns value of mQuiting.

Returns:
The value of mQuiting

4.1.3.26 NetServer * getServer () const [inline]

Getter of mServer. Returns value of mServer.

Returns:
The value of mServer

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

Getter of mUsageString. Returns value of mUsageString.

Returns:
The value of mUsageString

4.1.3.28 bool hasEnvironmentVariable (const std::string & *name*) [static]

Returns true if specified environment variable exists.

Parameters:
← ***name*** **Name**(p. 134) of the variable.

Returns:
Indicates that the specified environment variable exists.

4.1.3.29 void help () const [virtual]

Write commands.

4.1.3.30 void initialize () [virtual]

Initialize the PCM application.

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

Interrupt signal handler.

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

4.1.3.32 void loadDynamicScript (const std::string & *filename*) [protected, virtual]

Load dynamic script from the specified filename.

Parameters:
← ***filename*** File containing dynamic script (in application data directory).

4.1.3.33 void loadLowLevelScript (const std::string & *filename*) [protected, virtual]

Load low-level script from the specified filename.

Parameters:
← ***filename*** File containing low level script (in application data directory).

4.1.3.34 void NetworkTest () [virtual]

Network(p. 142) testing method.

4.1.3.35 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.

4.1.3.36 void quit () [virtual]

Quit from application.

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

Segmentation fault handler.

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

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

Set cluster description file.

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

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

Set commander mode.

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

4.1.3.40 void setDynamicScriptFile (const std::string & *strarg*) [static]

Set dynamic file.

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

4.1.3.41 void setFromConfigOptions () [virtual]

Call application-wide setters on config options.

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

Set GUI mode.

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

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

Set input file.

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

4.1.3.44 void setLibrarySearchPath (const std::string & *librarysearchpath*) [inline, static]

Setter of mLibrarySearchPath. Sets value of mLibrarySearchPath.

Parameters:
← ***librarysearchpath*** The value of mLibrarySearchPath

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

Set low-level scripting mode.

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

4.1.3.46 void setLowLevelScriptFile (const std::string & *strarg*) [static]

Set low-level file.

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

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

Set output file.

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

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

Set parameters description file.

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

4.1.3.49 void setupHandlers () const [virtual]

Setup special event handlers.

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

Write help to std out.

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

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

Write version to std out.

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

4.1.3.52 bool soldierOperation () [protected, virtual]

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

Returns:
Return true on error.

4.1.3.53 void start () [virtual]

Start compositing.

4.1.3.54 bool startOperation () [virtual]

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

Returns:
Return true on error.

4.1.3.55 void stop () [virtual]

Stop compositing.

4.1.3.56 void terminateHandler () [static]

Abnormal termination handler.

4.1.3.57 void terminateHandlerNOP () [static]

No operation termination handler. The **terminateHandler()**(p. 19) method sets this method as Abnormal termination handler during its run.

4.1.3.58 bool textUserInterface () [protected, virtual]

Start textual user interface.

Returns:
Return true on error.

4.1.3.59 void unexpectedHandler () [static]

Unexpected exception handler.

4.1.3.60 void writeOutput () [virtual]

Write collected output.

4.1.4 Member Data Documentation

4.1.4.1 NetClient* mClient [protected]

Client(p. 33) network.

Remarks:
This is own attribute of this class.

4.1.4.2 std::string mClusterDescription [static, protected]

Cluster(p. 37) description file name.

Remarks:
This is own attribute of this class.

4.1.4.3 bool mCommanderMode [static, protected]

Indicates commander mode (default false).

Remarks:
This is own attribute of this class.

4.1.4.4 const u32 mCommandLineOptionCount = sizeof(mCommandLineOptions) / sizeof(mCommandLineOptions[0]) [static, protected]

Number of command line options.

Remarks:
This is own attribute of this class.

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

Initial value:

```
{
        {'h', "help", " ........... : show help", false, Application::showHelp}
        ,
        {'v', "version", " ........ : show version", false, Application::showVersion}
        ,
        {'c', "commander", " ...... : enable commander mode", false, Application::setCommanderOn}
        ,



        {'l', "low-level", " ...... : set low-level script file", true, Application::setLowLevelScriptFile
        ,
        {'d', "dynamic", " ........ : set cluster dynamic script file", true, Application::setDynamicScrip
        ,
        {'i', "input", " .......... : set intput script file", true, Application::setInput}
        ,
        {'o', "output", " ......... : set output filename", true, Application::setOutput}
        ,
 }
```

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

Remarks:
This is own attribute of this class.

4.1.4.6 bool mCompRun [static, protected]

Output file name.

Remarks:
This is own attribute of this class.

4.1.4.7 ConfigOptions::Pointer mConfigOptions [protected]

Program configuration options.

Remarks:
This is own attribute of this class.

4.1.4.8 std::string mDynamicScript [protected]

Squirrel cluster dynamic script.

Remarks:
This is own attribute of this class.

4.1.4.9 SqVM::Pointer mDynamicScriptSqVM [protected]

Squirrel virtual machine for dynamic script execution.

Remarks:
This is own attribute of this class.

4.1.4.10 bool mGUIMode [static, protected]

Indicates GUI mode (default false).

Remarks:
This is own attribute of this class.

4.1.4.11 bool mInitialized [protected]

The application is initialized.

Remarks:
This is own attribute of this class.

4.1.4.12 std::string mInput [static, protected]

Input script file name.

Remarks:
This is own attribute of this class.

4.1.4.13 bool mInteractiveParameters [static, protected]

Indicates interactive parameter settings (default true).

Remarks:
This is own attribute of this class.

4.1.4.14 std::string mLibrarySearchPath = "/usr/lib" [static, protected]

PC dynamic library search path (default /usr/lib).

Remarks:
This is own attribute of this class.

4.1.4.15 bool mLowLevelMode [static, protected]

Indicates low-level scripting mode (default false).

Remarks:
This is own attribute of this class.

4.1.4.16 std::string mLowLevelScript [protected]

Squirrel low-level script.

Remarks:
This is own attribute of this class.

4.1.4.17 bool mManualClusterDescription [static, protected]

Indicates manual cluster description (default false).

Remarks:
This is own attribute of this class.

4.1.4.18 std::string mOutput [static, protected]

Output file name.

Remarks:
This is own attribute of this class.

4.1.4.19 OutputNode::Pointer mOutputDocument [protected]

Root of the output document.

Remarks:
This is own attribute of this class.

4.1.4.20 std::string mParameters [static, protected]

Parameters description file name.

Remarks:
This is own attribute of this class.

4.1.4.21 bool mQuiting [protected]

Flag indicating that the application is quitting now (to avoid multiple quit requests in **quit**()(p. 17) method).

Remarks:
This is own attribute of this class.

4.1.4.22 NetServer* mServer [protected]

Server network.

Remarks:
This is own attribute of this class.

4.1.4.23 const std::string mUsageString = "ParCompMark [options]" [static, protected]

Application(p. 9) usage string.

Remarks:
This is own attribute of this class.

4.1.4.24 const std::string PCAPINUT = "scripts/framework/pcapi.nut" [static]

PC API Squirrel script file (relative to the data directory).

Remarks:
This is own attribute of this class.

4.1.4.25 const std::string PCIMPLEMENTATIONS [static]

Initial value:

```
{
        "libpcapi.so", "libparcomp.so"
 }
```

List of supported PC implementation libraries.

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

Grapics memory buffer. **Buffer**(p. 26) object can store either colour and depth information of an image (PC result) or a fragment of an image (PC framelet).

Methods

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

Public Types

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

Public Member Functions

Constructors & destructor

- **Buffer** ()
- **Buffer** (const PCuint &left, const PCuint &top, const PCuint &width, const PCuint &height, const PCuint &depthFormat, **Node** *parent)
- virtual ~**Buffer** ()

Getters & setters

- **Node** * **getParent** () const
- const PCuint & **getLeft** () const
- void **setLeft** (const PCuint left)
- const PCuint & **getTop** () const
- void **setTop** (const PCuint top)
- const PCuint & **getWidth** () const
- void **setWidth** (const PCuint width)
- const PCuint & **getHeight** () const
- void **setHeight** (const PCuint height)
- const bool & **getOwnPointers** () const
- PCuint * **getColour** ()
- void **setColour** (const PCuint *colour)
- void * **getDepth** ()
- void **setDepth** (const void *depth)
- const PCint & **getDepthFormat** () const
- void **setDepthFormat** (const PCint depthformat)
- const PCint & **getOutputRowPixel** () const
- void **setOutputRowPixel** (const PCint &outputrowpixel)
- const bool & **getInitialized** () const

Static Public Member Functions

Scripting binding

- static void **squirrelGlue** ()

Protected Attributes

Variables

- **Node** ∗ **mParent**
- PCuint **mLeft**
- PCuint **mTop**
- PCuint **mWidth**
- PCuint **mHeight**
- bool **mOwnPointers**
- PCuint ∗ **mColour**
- void ∗ **mDepth**
- PCint **mDepthFormat**
- PCint **mOutputRowPixel**
- bool **mInitialized**

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 for Squirrel compatibility. Calling this constructor always raises an exception.

4.3.3.2 Buffer (const PCuint & *left*, const PCuint & *top*, const PCuint & *width*, const PCuint & *height*, const PCuint & *depthFormat*, Node ∗ *parent*)

Create a buffer by initializing its parameters.

Parameters:

← ***left*** X offset of the image (it is 0 for whole images).

← ***top*** Y offset of the image (it is 0 for whole images).

← ***width*** Width of the image.

← ***height*** Height of the image.

← ***depthFormat*** Depth format used by this image.

← ***parent*** Parent node

4.3.3.3 ∼Buffer () [virtual]

The destructor. This class has virtual destructor.

4.3.4 Member Function Documentation

4.3.4.1 void finalize () [virtual]

Finalize the buffer.

4.3.4.2 void freeBuffers () [protected, virtual]

Deallocate mColour and mDepth buffers.

4.3.4.3 PCuint * getColour () [inline]

Getter of mColour. Returns value of mColour.

Returns:
The value of mColour

4.3.4.4 void * getDepth () [inline]

Getter of mDepth. Returns value of mDepth.

Returns:
The value of mDepth

4.3.4.5 const PCint & getDepthFormat () const [inline]

Getter of mDepthFormat. Returns value of mDepthFormat.

Returns:
The value of mDepthFormat

4.3.4.6 const PCuint & getHeight () const [inline]

Getter of mHeight. Returns value of mHeight.

Returns:
The value of mHeight

4.3.4.7 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:
The value of mInitialized

4.3.4.8 const PCuint & getLeft () const [inline]

Getter of mLeft. Returns value of mLeft.

Returns:
The value of mLeft

4.3.4.9 const PCint & getOutputRowPixel () const [inline]

Getter of mOutputRowPixel. Returns value of mOutputRowPixel.

Returns:
The value of mOutputRowPixel

4.3.4.10 const bool & getOwnPointers () const [inline]

Getter of mOwnPointers. Returns value of mOwnPointers.

Returns:
The value of mOwnPointers

4.3.4.11 Node * getParent () const [inline]

Getter of mParent. Returns value of mParent.

Returns:
The value of mParent

4.3.4.12 const PCuint & getTop () const [inline]

Getter of mTop. Returns value of mTop.

Returns:
The value of mTop

4.3.4.13 const PCuint & getWidth () const [inline]

Getter of mWidth. Returns value of mWidth.

Returns:
The value of mWidth

4.3.4.14 void initialize () [virtual]

Initialize the buffer.

4.3.4.15 void setColour (const PCuint * *colour*) [inline]

Setter of mColour. Sets value of mColour.

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

4.3.4.16 void setDepth (const void * *depth*) [inline]

Setter of mDepth. Sets value of mDepth.

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

4.3.4.17 void setDepthFormat (const PCint *depthformat*) [inline]

Setter of mDepthFormat. Sets value of mDepthFormat.

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

4.3.4.18 void setHeight (const PCuint *height*) [inline]

Setter of mHeight. Sets value of mHeight.

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

4.3.4.19 void setLeft (const PCuint *left*) [inline]

Setter of mLeft. Sets value of mLeft.

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

4.3.4.20 void setOutputRowPixel (const PCint & *outputrowpixel*) [inline]

Setter of mOutputRowPixel. Sets value of mOutputRowPixel.

Parameters:
← ***outputrowpixel*** The value of mOutputRowPixel

4.3.4.21 void setTop (const PCuint *top*) [inline]

Setter of mTop. Sets value of mTop.

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

4.3.4.22 void setWidth (const PCuint *width*) [inline]

Setter of mWidth. Sets value of mWidth.

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

4.3.4.23 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.

4.3.5 Member Data Documentation

4.3.5.1 PCuint* mColour [protected]

Colour information buffer (RGBA32 or BGRA32 coded).

Remarks:
This is own attribute of this class.

4.3.5.2 void* mDepth [protected]

Depth information buffer (the coding is unknown).

Remarks:
This is own attribute of this class.

4.3.5.3 PCint mDepthFormat [protected]

Depth format. Needed for deallocating the buffer.

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 bool mInitialized [protected]

The buffer is initialized. None of its attribute can be modified after the buffer is initialized except for mOutputRowPixel, mColour, and mDepth.

Remarks:
This is own attribute of this class.

4.3.5.6 PCuint mLeft [protected]

Remarks:
This is own attribute of this class.

4.3.5.7 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.

4.3.5.8 bool mOwnPointers [protected]

The **Buffer**(p. 26) is responsible for deallocation of mColour and mDepth.

Remarks:
This is own attribute of this class.

4.3.5.9 Node* mParent [protected]

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

Remarks:
This attribute references an attribute.

4.3.5.10 PCuint mTop [protected]

Remarks:
This is own attribute of this class.

4.3.5.11 PCuint mWidth [protected]

Remarks:
This is own attribute of this class.

4.4 Client Class Reference

Inherits **Network**.

Inherited by **HandleClient**, and **NetClient**.

4.4.1 Detailed Description

Client(p. 33) functions.

Public Types

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

Public Member Functions

Constructors & destructor

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

Getters & setters

- const std::string & **getServerIP** () const
- void **setServerIP** (const std::string &serverip)
- const std::string & **getType** () const
- const std::string & **getMessage** () const
- **Network** * **getParent** () const

Methods

- virtual void **openConnection** ()
- virtual void **closeConnection** ()
- virtual void **sendMessage** (const std::string &type, const std::string &message)
- virtual void **recieveMessage** ()
- virtual void **handleMessage** ()

Protected Attributes

Variables

- std::string **mServerIP**
- std::string **mType**
- std::string **mMessage**
- **Network** * **mParent**

4.4.2 Member Typedef Documentation

4.4.2.1 typedef ParCompMark::Pointer< Client, DummyLock > Pointer

Type for pointer to this class.

Reimplemented from **Network** (p. 143).

Reimplemented in **HandleClient** (p. 109), and **NetClient** (p. 136).

4.4.3 Constructor & Destructor Documentation

4.4.3.1 Client (const std::string & *name*)

Create network class.

Parameters:
← ***name*** **Name**(p. 134) of the network.

4.4.3.2 ~Client () [virtual]

The destructor. This class has virtual destructor.

4.4.4 Member Function Documentation

4.4.4.1 void closeConnection () [virtual]

Close TCP/IP connection (mServerIP).

4.4.4.2 const std::string & getMessage () const [inline]

Getter of mMessage. Returns value of mMessage.

Returns:
The value of mMessage

4.4.4.3 Network * getParent () const [inline]

Getter of mParent. Returns value of mParent.

Returns:
The value of mParent

4.4.4.4 const std::string & getServerIP () const [inline]

Getter of mServerIP. Returns value of mServerIP.

Returns:
The value of mServerIP

4.4.4.5 const std::string & getType () const [inline]

Getter of mType. Returns value of mType.

Returns:
The value of mType

4.4.4.6 void handleMessage () [virtual]

Handle TCP/IP message.

4.4.4.7 void openConnection () [virtual]

Open TCP/IP connection (mServerIP).

Reimplemented in **HandleClient** (p. 110).

4.4.4.8 void recieveMessage () [virtual]

Wait for TCP/IP message.

4.4.4.9 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.

4.4.4.10 void setServerIP (const std::string & *serverip*) [inline]

Setter of mServerIP. Sets value of mServerIP.

Parameters:

← ***serverip*** The value of mServerIP

4.4.5 Member Data Documentation

4.4.5.1 std::string mMessage [protected]

IP address of the server host.

Remarks:

This is own attribute of this class.

4.4.5.2 Network* mParent [protected]

The parent network.

Remarks:

This attribute references an attribute.

4.4.5.3 std::string mServerIP [protected]

IP address of the server host.

Remarks:
This is own attribute of this class.

4.4.5.4 std::string mType [protected]

IP address of the server host.

Remarks:
This is own attribute of this class.

4.5 Cluster Class Reference

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

4.5.1 Detailed Description

Class contain hosts. Description of the physical grid.

Public Member Functions

Constructors & destructor

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

Getters & setters

- **Container**< **HostInfo**, **Mutex** >::**Pointer** & **getHosts** ()

Methods

- virtual std::string **serialize2Squirrel** ()

Protected Attributes

Variables

- **Container**< **HostInfo**, **Mutex** >::**Pointer mHosts**

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Cluster ()

Default constructor.

4.5.2.2 ~Cluster () [virtual]

The destructor. This class has virtual destructor.

4.5.3 Member Function Documentation

4.5.3.1 Container< HostInfo, Mutex >::Pointer & getHosts () [inline]

Getter of mHosts. Returns value of mHosts.

Returns:
The value of mHosts

4.5.3.2 std::string serialize2Squirrel () [virtual]

Serialize the cluster to Squirrel structure code snipplet.

Returns:
Serialized Squirrel structure code snipplet.

4.5.4 Member Data Documentation

4.5.4.1 Container< HostInfo, Mutex >::Pointer mHosts [protected]

Hosts in the cluster.

Remarks:
This is own attribute of this class.

4.6 ConfigOptions Class Reference

4.6.1 Detailed Description

Program configuration options. Stored in application ini file.

Methods

- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual bool **hasOption** (const std::string &name)
- virtual std::string **getString** (const std::string &name)
- virtual bool **getBool** (const std::string &name)
- virtual **s32 getInteger** (const std::string &name)
- virtual **Real getReal** (const std::string &name)
- virtual bool **testString** (const std::string &name, const std::string &value)
- virtual void **setString** (const std::string &name, const std::string &value, const std::string &description="")
- virtual void **setBool** (const std::string &name, const bool &value, const std::string &description="")
- virtual void **setInteger** (const std::string &name, const **s32** &value, const std::string &description="")
- virtual void **setReal** (const std::string &name, const **Real** &value, const std::string &description="")
- virtual void **removeOption** (const std::string &name)
- virtual void **saveToIniFile** (const std::string &filename)
- virtual void **loadFromIniFile** (const std::string &filename)
- virtual **ConfigOptions::Option::Pointer getValue** (const std::string &name, const **ConfigOptions::OptionType** &type)
- virtual void **setValue** (const std::string &name, const void ∗value, const **ConfigOptions::OptionType** &type, const std::string &description)
- virtual **u32 getOptionTypeSize** (const **ConfigOptions::OptionType** &type)
- virtual std::string **optionToString** (**ConfigOptions::Option::Pointer** &option)

Public Types

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

Public Member Functions

Constructors & destructor

- **ConfigOptions** ()
- virtual ∼**ConfigOptions** ()

Getters & setters

- const bool & **getInitialized** () const

Protected Types

- typedef **ParCompMark::ConfigOptions::Option Option**
- enum **OptionType** { **STRING**, **BOOL**, **INTEGER**, **REAL** }

Protected Attributes

Variables

- **Container**< **ConfigOptions::Option**, **Mutex** >::**Pointer mOptions**
- bool **mInitialized**

Classes

- struct **Option**

4.6.2 Member Typedef Documentation

4.6.2.1 typedef struct ParCompMark::ConfigOptions::Option Option [protected]

Struct for option.

4.6.2.2 typedef Pointer< ConfigOptions, Mutex > Pointer

Type for pointer on this class.

4.6.3 Member Enumeration Documentation

4.6.3.1 enum OptionType [protected]

Option(p. 46) type definitions.

Enumerator:

STRING STL string type

BOOL Bool type

INTEGER s32 integer type

REAL Real type

4.6.4 Constructor & Destructor Documentation

4.6.4.1 ConfigOptions ()

Default constructor.

4.6.4.2 ~ConfigOptions () [virtual]

The destructor. This class has virtual destructor.

4.6.5 Member Function Documentation

4.6.5.1 void finalize () [virtual]

Finalize option container.

4.6.5.2 bool getBool (const std::string & *name*) [virtual]

Get option.

Parameters:
← ***name*** **Name**(p. 134) of the option

Returns:
Bool value

4.6.5.3 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:
The value of mInitialized

4.6.5.4 s32 getInteger (const std::string & *name*) [virtual]

Get option.

Parameters:
← ***name*** **Name**(p. 134) of the option

Returns:
Integer value

4.6.5.5 u32 getOptionTypeSize (const ConfigOptions::OptionType & *type*) [protected, virtual]

Get size of option type

Parameters:
← ***type*** Type of the option

Returns:
Size of the specified type

4.6.5.6 Real getReal (const std::string & *name*) [virtual]

Get option.

Parameters:
← ***name*** **Name**(p. 134) of the option

Returns:
Real value

4.6.5.7 std::string getString (const std::string & *name*) [virtual]

Get option.

Parameters:
← ***name*** **Name**(p. 134) of the option

Returns:
STL string value

4.6.5.8 ConfigOptions::Option::Pointer getValue (const std::string & *name*, const ConfigOptions::OptionType & *type*) [protected, virtual]

Get option value for the specified type (internal method).

Parameters:
← ***name*** **Name**(p. 134) of the option

← ***type*** Type of the option

Returns:
Pointer(p. 177) to the specified option

4.6.5.9 bool hasOption (const std::string & *name*) [virtual]

Check for option.

Parameters:
← ***name*** **Name**(p. 134) of the option

Returns:
True if the container has the specified option.

4.6.5.10 void initialize () [virtual]

Initialize option container.

4.6.5.11 void loadFromIniFile (const std::string & *filename*) [virtual]

Load user-level ini file.

Parameters:
← ***filename*** **Name**(p. 134) of the config file

4.6.5.12 std::string optionToString (ConfigOptions::Option::Pointer & *option*) [protected, virtual]

Give string representation of the option value

Parameters:
→ ***option*** **Option**(p. 46)

Returns:
String representation

4.6.5.13 void removeOption (const std::string & *name*) [virtual]

Remove option.

Parameters:
← ***name*** **Name**(p. 134) of the option

4.6.5.14 void saveToIniFile (const std::string & *filename*) [virtual]

Create user-level ini file in the application directory (existing file will be overwritten).

Parameters:
← ***filename*** **Name**(p. 134) of the config file

4.6.5.15 void setBool (const std::string & *name*, const bool & *value*, const std::string & *description* = "") [virtual]

Set option.

Parameters:
← ***name*** **Name**(p. 134) of the option
← ***value*** Value of the option
← ***description*** Description of the option

4.6.5.16 void setInteger (const std::string & *name*, const s32 & *value*, const std::string & *description* = "") [virtual]

Set option.

Parameters:

← ***name*** **Name**(p. 134) of the option

← ***value*** Value of the option

← ***description*** Description of the option

4.6.5.17 void setReal (const std::string & *name*, const Real & *value*, const std::string & *description* = "") [virtual]

Set option.

Parameters:

← ***name*** **Name**(p. 134) of the option

← ***value*** Value of the option

← ***description*** Description of the option

4.6.5.18 void setString (const std::string & *name*, const std::string & *value*, const std::string & *description* = "") [virtual]

Set option.

Parameters:

← ***name*** **Name**(p. 134) of the option

← ***value*** Value of the option

← ***description*** Description of the option

4.6.5.19 void setValue (const std::string & *name*, const void * *value*, const ConfigOptions::OptionType & *type*, const std::string & *description*) [protected, virtual]

Set option value for the specified type (internal method).

Parameters:

← ***name*** **Name**(p. 134) of the option

← ***value*** Address of value

← ***type*** Type of the option

← ***description*** Description of the option

4.6.5.20 bool testString (const std::string & *name*, const std::string & *value*) [virtual]

Return true if the option exists with the given name and equals to the given value.

Parameters:

← ***name*** **Name**(p. 134) of the option

← ***value*** Value to test

Returns:

True if the option exists with the given name and equals to the given value

4.6.6 Member Data Documentation

4.6.6.1 bool mInitialized [protected]

The option container is initialized.

Remarks:
This is own attribute of this class.

4.6.6.2 Container< ConfigOptions::Option, Mutex >::Pointer mOptions [protected]

Container(p. 47) for application options.

Remarks:
This is own attribute of this class.

4.7 ConfigOptions::Option Struct Reference

4.7.1 Detailed Description

Struct for option.

Public Types

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

Public Member Functions

- ~**Option** ()

Public Attributes

- **OptionType type**
- void * **value**
- std::string **description**

4.7.2 Member Typedef Documentation

4.7.2.1 typedef ParCompMark::Pointer< Option, DummyLock > Pointer

Smart pointer on this struct

4.7.3 Constructor & Destructor Documentation

4.7.3.1 ~Option () [inline]

Destructor of the option

4.7.4 Member Data Documentation

4.7.4.1 std::string description

Description of the option (optional)

4.7.4.2 OptionType type

Type of the option

4.7.4.3 void* value

Value of the option

4.8 Container Class Template Reference

4.8.1 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 bool **isEmpty** ()
- virtual **Iterator begin** ()
- virtual **Iterator end** ()

Protected Attributes

Variables

- **ElementsMap mElements**

4.8.2 Member Typedef Documentation

4.8.2.1 typedef ElementType::Pointer ElementPointer

Type definition for pointer on elements.

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

Type definition for map of elements.

4.8.2.3 typedef ElementsMap::iterator Iterator

Type definition for iterator on map of elements.

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

Type for pointer on this class.

4.8.3 Constructor & Destructor Documentation

4.8.3.1 Container () [inline]

Default constructor.

4.8.3.2 ~Container () [inline, virtual]

The destructor. This class has virtual destructor.

4.8.4 Member Function Documentation

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

Add an element.

Parameters:

← ***name*** **Name**(p. 134) of the element

← ***element*** Element

4.8.4.2 std::map< std::string, typename ElementType::Pointer >::iterator begin () [virtual]

Begin iterator on the container.

Returns:

Iterator on the first element.

4.8.4.3 std::map< std::string, typename ElementType::Pointer >::iterator end () [virtual]

End iterator on the container.

Returns:

Iterator on the element after the last element.

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

Get an element by name.

Parameters:
← ***name*** **Name**(p. 134) of the element

Returns:
Pointer(p. 177) to the element

4.8.4.5 u32 getSize () [virtual]

Return the number of elements.

Returns:
Pointer(p. 177) to the element

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

Search for an element by name.

Parameters:
← ***name*** **Name**(p. 134) of the element

Returns:
The container has the element.

4.8.4.7 bool isEmpty () [virtual]

Return true if the container is empty.

Returns:
True if the container is empty

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

Remove an element by name.

Parameters:
← ***name*** **Name**(p. 134) of the element

4.8.5 Member Data Documentation

4.8.5.1 ElementsMap mElements [protected]

Map of elements.

Remarks:
This is own attribute of this class.

4.9 Context Class Reference

4.9.1 Detailed Description

Class containing PC context information.

Public Types

- typedef **Pointer**< **Context**, **DummyLock** > **Pointer**
- enum **ContextType** { **MASTER** = 0, **SLAVE** = 1 }

Public Member Functions

Constructors & destructor

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

Getters & setters

- bool & **getUseGLFrameletEXT** ()
- void **setUseGLFrameletEXT** (const bool useglframeletext)
- **Context::ContextType** & **getContextType** ()
- void **setContextType** (const **Context::ContextType** &contexttype)
- PCint & **getFrameWidth** ()
- void **setFrameWidth** (const PCint framewidth)
- PCint & **getFrameHeight** ()
- void **setFrameHeight** (const PCint frameheight)
- PCint & **getColourFormat** ()
- void **setColourFormat** (const PCint colourformat)
- PCint & **getDepthFormat** ()
- void **setDepthFormat** (const PCint depthformat)
- const PCint & **getPixelFormat** () const
- PCint & **getCompositeType** ()
- void **setCompositeType** (const PCint compositetype)
- PCint & **getCompressionHint** ()
- void **setCompressionHint** (const PCint compressionhint)
- PCint & **getRetainOutputCount** ()
- void **setRetainOutputCount** (const PCint retainoutputcount)
- PCint & **getVolatileFrameletCount** ()
- void **setVolatileFrameletCount** (const PCint volatileframeletcount)
- bool & **getOutputDepth** ()
- void **setOutputDepth** (const bool outputdepth)
- PCstring * **getProcesses** () const
- const PCint & **getProcessCount** () const
- int & **getProcessIndex** ()
- void **setProcessIndex** (const int processindex)
- const PCint & **getHostIndex** () const
- PCint & **getNetworkID** ()
- void **setNetworkID** (const PCint networkid)
- const PCcontext & **getContext** () const
- **Process** * **getParent** ()
- const bool & **getInitialized** () const

Methods

- virtual void **setProcesses** (const char *processList)
- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual void **setContextTypeSq** (const int contextType)
- virtual int **getContextTypeSq** ()

Static Public Member Functions

Scripting binding

- static void **squirrelGlue** ()

Protected Attributes

Variables

- bool **mUseGLFrameletEXT**
- **ContextType mContextType**
- PCint **mFrameWidth**
- PCint **mFrameHeight**
- PCint **mColourFormat**
- PCint **mDepthFormat**
- PCint **mPixelFormat**
- PCint **mCompositeType**
- PCint **mCompressionHint**
- PCint **mRetainOutputCount**
- PCint **mVolatileFrameletCount**
- bool **mOutputDepth**
- PCstring * **mProcesses**
- PCint **mProcessCount**
- int **mProcessIndex**
- PCint **mHostIndex**
- PCint **mNetworkID**
- PCcontext **mContext**
- **Process** * **mParent**
- bool **mInitialized**

4.9.2 Member Typedef Documentation

4.9.2.1 typedef Pointer< Context, DummyLock > Pointer

Type for pointer on this class.

4.9.3 Member Enumeration Documentation

4.9.3.1 enum ContextType

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

Enumerator:

MASTER Master context.

SLAVE Slave context.

4.9.4 Constructor & Destructor Documentation

4.9.4.1 Context ()

Default constructor for Squirrel compatibility. Calling this constructor always raises an exception.

4.9.4.2 Context (Process * *parent*)

Create **Context**(p. 50). Normally **Process**(p. 185) calls this constructor.

Parameters:
← ***parent*** Parent host

4.9.4.3 ~Context () [virtual]

The destructor. This class has virtual destructor.

4.9.5 Member Function Documentation

4.9.5.1 void finalize () [virtual]

Finalize the PC context.

4.9.5.2 PCint & getColourFormat () [inline]

Getter of mColourFormat. Returns value of mColourFormat.

Returns:
The value of mColourFormat

4.9.5.3 PCint & getCompositeType () [inline]

Getter of mCompositeType. Returns value of mCompositeType.

Returns:
The value of mCompositeType

4.9.5.4 PCint & getCompressionHint () [inline]

Getter of mCompressionHint. Returns value of mCompressionHint.

Returns:
The value of mCompressionHint

4.9.5.5 const PCcontext & getContext () const [inline]

Getter of mContext. Returns value of mContext.

Returns:
The value of mContext

4.9.5.6 Context::ContextType & getContextType () [inline]

Getter of mContextType. Returns value of mContextType.

Returns:
The value of mContextType

4.9.5.7 int getContextTypeSq () [inline, virtual]

Alternate squirrel setter of mContextType.

Returns:
Context(p. 50) type int value (0 means MASTER, 1 means SLAVE).

4.9.5.8 PCint & getDepthFormat () [inline]

Getter of mDepthFormat. Returns value of mDepthFormat.

Returns:
The value of mDepthFormat

4.9.5.9 PCint & getFrameHeight () [inline]

Getter of mFrameHeight. Returns value of mFrameHeight.

Returns:
The value of mFrameHeight

4.9.5.10 PCint & getFrameWidth () [inline]

Getter of mFrameWidth. Returns value of mFrameWidth.

Returns:
The value of mFrameWidth

4.9.5.11 const PCint & getHostIndex () const [inline]

Getter of mHostIndex. Returns value of mHostIndex.

Returns:
The value of mHostIndex

4.9.5.12 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:
The value of mInitialized

4.9.5.13 PCint & getNetworkID () [inline]

Getter of mNetworkID. Returns value of mNetworkID.

Returns:
The value of mNetworkID

4.9.5.14 bool & getOutputDepth () [inline]

Getter of mOutputDepth. Returns value of mOutputDepth.

Returns:
The value of mOutputDepth

4.9.5.15 Process * getParent () [inline]

Getter of mParent. Returns value of mParent.

Returns:
The value of mParent

4.9.5.16 const PCint & getPixelFormat () const [inline]

Getter of mPixelFormat. Returns value of mPixelFormat.

Returns:
The value of mPixelFormat

4.9.5.17 const PCint & getProcessCount () const [inline]

Getter of mProcessCount. Returns value of mProcessCount.

Returns:
The value of mProcessCount

4.9.5.18 PCstring * getProcesses () const [inline]

Getter of mProcesses. Returns value of mProcesses.

Returns:
The value of mProcesses

4.9.5.19 int & getProcessIndex () [inline]

Getter of mProcessIndex. Returns value of mProcessIndex.

Returns:
The value of mProcessIndex

4.9.5.20 PCint & getRetainOutputCount () [inline]

Getter of mRetainOutputCount. Returns value of mRetainOutputCount.

Returns:
The value of mRetainOutputCount

4.9.5.21 bool & getUseGLFrameletEXT () [inline]

Getter of mUseGLFrameletEXT. Returns value of mUseGLFrameletEXT.

Returns:
The value of mUseGLFrameletEXT

4.9.5.22 PCint & getVolatileFrameletCount () [inline]

Getter of mVolatileFrameletCount. Returns value of mVolatileFrameletCount.

Returns:
The value of mVolatileFrameletCount

4.9.5.23 void initialize () [virtual]

Init the PC context.

4.9.5.24 void setColourFormat (const PCint *colourformat*) [inline]

Setter of mColourFormat. Sets value of mColourFormat.

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

4.9.5.25 void setCompositeType (const PCint *compositetype*) [inline]

Setter of mCompositeType. Sets value of mCompositeType.

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

4.9.5.26 void setCompressionHint (const PCint *compressionhint*) [inline]

Setter of mCompressionHint. Sets value of mCompressionHint.

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

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

Setter of mContextType. Sets value of mContextType.

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

4.9.5.28 void setContextTypeSq (const int *contextType*) [inline, virtual]

Alternate squirrel setter of mContextType.

Parameters:
← ***contextType*** **Context**(p. 50) type int value (0 means MASTER, 1 means SLAVE).

4.9.5.29 void setDepthFormat (const PCint *depthformat*) [inline]

Setter of mDepthFormat. Sets value of mDepthFormat.

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

4.9.5.30 void setFrameHeight (const PCint *frameheight*) [inline]

Setter of mFrameHeight. Sets value of mFrameHeight.

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

4.9.5.31 void setFrameWidth (const PCint *framewidth*) [inline]

Setter of mFrameWidth. Sets value of mFrameWidth.

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

4.9.5.32 void setNetworkID (const PCint *networkid*) [inline]

Setter of mNetworkID. Sets value of mNetworkID.

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

4.9.5.33 void setOutputDepth (const bool *outputdepth*) [inline]

Setter of mOutputDepth. Sets value of mOutputDepth.

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

4.9.5.34 void setProcesses (const char * *processList*) [virtual]

Create process array from string separated by semicolons.

Parameters:
← ***processList*** **Process**(p. 185) list separated by semicolons (C string for squirrel compatibility).

4.9.5.35 void setProcessIndex (const int *processindex*) [inline]

Setter of mProcessIndex. Sets value of mProcessIndex.

Parameters:
← ***processindex*** The value of mProcessIndex

4.9.5.36 void setRetainOutputCount (const PCint *retainoutputcount*) [inline]

Setter of mRetainOutputCount. Sets value of mRetainOutputCount.

Parameters:
← ***retainoutputcount*** The value of mRetainOutputCount

4.9.5.37 void setUseGLFrameletEXT (const bool *useglframeletext*) [inline]

Setter of mUseGLFrameletEXT. Sets value of mUseGLFrameletEXT.

Parameters:
← ***useglframeletext*** The value of mUseGLFrameletEXT

4.9.5.38 void setVolatileFrameletCount (const PCint *volatileframeletcount*) [inline]

Setter of mVolatileFrameletCount. Sets value of mVolatileFrameletCount.

Parameters:
← ***volatileframeletcount*** The value of mVolatileFrameletCount

4.9.5.39 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.

4.9.6 Member Data Documentation

4.9.6.1 PCint mColourFormat [protected]

Colour format of rendered frame by the context.

Remarks:
This is own attribute of this class.

4.9.6.2 PCint mCompositeType [protected]

Type of composition.

Remarks:
This is own attribute of this class.

4.9.6.3 PCint mCompressionHint [protected]

Compression hint.

Remarks:
This is own attribute of this class.

4.9.6.4 PCcontext mContext [protected]

The PC context.

Remarks:
This is own attribute of this class.

4.9.6.5 ContextType mContextType [protected]

Type of the context.

Remarks:
This is own attribute of this class.

4.9.6.6 PCint mDepthFormat [protected]

Depth format of rendered frame by the context.

Remarks:
This is own attribute of this class.

4.9.6.7 PCint mFrameHeight [protected]

Height of rendered frame by the context.

Remarks:
This is own attribute of this class.

4.9.6.8 PCint mFrameWidth [protected]

Width of rendered frame by the context.

Remarks:
This is own attribute of this class.

4.9.6.9 PCint mHostIndex [protected]

The own host index.

Remarks:
This is own attribute of this class.

4.9.6.10 bool mInitialized [protected]

The context is initialized.

Remarks:
This is own attribute of this class.

4.9.6.11 PCint mNetworkID [protected]

The network ID.

Remarks:
This is own attribute of this class.

4.9.6.12 bool mOutputDepth [protected]

Needs depth information of the composition.

Remarks:
This is own attribute of this class.

4.9.6.13 Process* mParent [protected]

Parent **Process**(p. 185) of this **Context**(p. 50).

Remarks:
This attribute references an attribute.

4.9.6.14 PCint mPixelFormat [protected]

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

Remarks:
This is own attribute of this class.

4.9.6.15 PCint mProcessCount [protected]

Number of processes (PC nomenclature nodes) in this context.

Remarks:
This is own attribute of this class.

4.9.6.16 PCstring* mProcesses [protected]

Processes in the context. The format of the strings is "hostname:localid".

Remarks:
This is own attribute of this class.

4.9.6.17 int mProcessIndex [protected]

Position of the parent process in mProcesses array.

Remarks:
This is own attribute of this class.

4.9.6.18 PCint mRetainOutputCount [protected]

The output of a frame is available between a pcFrameEnd and the following "mRetainOutput"th pcFrameBegin.

Remarks:
This is own attribute of this class.

4.9.6.19 bool mUseGLFrameletEXT [protected]

Use OpenGL framelet PC library extension.

Remarks:
This is own attribute of this class.

4.9.6.20 PCint mVolatileFrameletCount [protected]

How many frames in the future the application will resuse the buffer.

Remarks:
This is own attribute of this class.

4.10 CPU Class Reference

Inherits **OutputNode**.

4.10.1 Detailed Description

Class contain **CPU**(p. 62) information.

Public Types

- typedef **ParCompMark::Pointer**< **CPU**, **Mutex** > **Pointer**

Public Member Functions

Constructors & destructor

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

Getters & setters

- const std::string & **getVendor** () const
- void **setVendor** (const std::string &vendor)
- const std::string & **getModel** () const
- void **setModel** (const std::string &model)
- const **Real** & **getClock** () const
- void **setClock** (const **Real** &clock)
- const **u32** & **getCache** () const
- void **setCache** (const **u32** &cache)
- const **Real** & **getBogomips** () const
- void **setBogomips** (const **Real** &bogomips)
- const std::string & **getFlags** () const
- void **setFlags** (const std::string &flags)

Methods

- virtual void **refreshData** ()

Protected Attributes

Variables

- std::string **mVendor**
- std::string **mModel**
- **Real mClock**
- **u32 mCache**
- **Real mBogomips**
- std::string **mFlags**

4.10.2 Member Typedef Documentation

4.10.2.1 typedef ParCompMark::Pointer< CPU, Mutex > Pointer

Type for pointer on this class.

Reimplemented from **OutputNode** (p. 161).

4.10.3 Constructor & Destructor Documentation

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

Creates **CPU**(p. 62) node.

Parameters:

← ***name*** **Name**(p. 134) of the node.

← ***type*** Type of the node.

4.10.3.2 ~CPU () [virtual]

The destructor. This class has virtual destructor.

4.10.4 Member Function Documentation

4.10.4.1 const Real & getBogomips () const [inline]

Getter of mBogomips. Returns value of mBogomips.

Returns:

The value of mBogomips

4.10.4.2 const u32 & getCache () const [inline]

Getter of mCache. Returns value of mCache.

Returns:

The value of mCache

4.10.4.3 const Real & getClock () const [inline]

Getter of mClock. Returns value of mClock.

Returns:

The value of mClock

4.10.4.4 const std::string & getFlags () const [inline]

Getter of mFlags. Returns value of mFlags.

Returns:
The value of mFlags

4.10.4.5 const std::string & getModel () const [inline]

Getter of mModel. Returns value of mModel.

Returns:
The value of mModel

4.10.4.6 const std::string & getVendor () const [inline]

Getter of mVendor. Returns value of mVendor.

Returns:
The value of mVendor

4.10.4.7 void refreshData () [virtual]

Refresh **CPU**(p. 62) datas.

Reimplemented from **OutputNode** (p. 164).

4.10.4.8 void setBogomips (const Real & *bogomips*) [inline]

Setter of mBogomips. Sets value of mBogomips.

Parameters:
← ***bogomips*** The value of mBogomips

4.10.4.9 void setCache (const u32 & *cache*) [inline]

Setter of mCache. Sets value of mCache.

Parameters:
← ***cache*** The value of mCache

4.10.4.10 void setClock (const Real & *clock*) [inline]

Setter of mClock. Sets value of mClock.

Parameters:
← ***clock*** The value of mClock

4.10.4.11 void setFlags (const std::string & *flags*) [inline]

Setter of mFlags. Sets value of mFlags.

Parameters:
← ***flags*** The value of mFlags

4.10.4.12 void setModel (const std::string & *model*) [inline]

Setter of mModel. Sets value of mModel.

Parameters:
← ***model*** The value of mModel

4.10.4.13 void setVendor (const std::string & *vendor*) [inline]

Setter of mVendor. Sets value of mVendor.

Parameters:
← ***vendor*** The value of mVendor

4.10.5 Member Data Documentation

4.10.5.1 Real mBogomips [protected]

BogoMIPS value for the **CPU**(p. 62). MIPS is short for Millions of Instructions Per Second. It is a measure for the computation speed of a program.

Remarks:
This is own attribute of this class.

4.10.5.2 u32 mCache [protected]

CPU(p. 62) cache in KB.

Remarks:
This is own attribute of this class.

4.10.5.3 Real mClock [protected]

CPU(p. 62) clock in MHz.

Remarks:
This is own attribute of this class.

4.10.5.4 std::string mFlags [protected]

CPU(p. 62) flags.

Remarks:
This is own attribute of this class.

4.10.5.5 std::string mModel [protected]

Name(p. 134) of the **CPU**(p. 62) model.

Remarks:
This is own attribute of this class.

4.10.5.6 std::string mVendor [protected]

Name(p. 134) of the vendor of the **CPU**(p. 62).

Remarks:
This is own attribute of this class.

4.11 DummyLock Class Reference

Inherits **Lock**.

4.11.1 Detailed Description

Dummy lock implementing the **Lock**(p. 124) interface. Does not do anything. The **DummyLock**(p. 67) can always be locked.

Public Member Functions

Constructors & destructor

- virtual ~**DummyLock** ()

Methods

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

4.11.2 Constructor & Destructor Documentation

4.11.2.1 ~DummyLock () [inline, virtual]

The destructor. This class has virtual destructor.

4.11.3 Member Function Documentation

4.11.3.1 void lock () [inline, virtual]

Lock(p. 124) the lock. Does not do anything.

Implements **Lock** (p. 125).

4.11.3.2 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. 125).

4.11.3.3 void unlock () [inline, virtual]

Unlock the lock. Does not do anything.

Implements **Lock** (p. 125).

4.12 DynLoad Class Reference

Inherited by **Plugin**.

4.12.1 Detailed Description

Dynamic load library.

Methods

- virtual bool **hasFunction** (const std::string &funcName) const
- virtual void * **getFunction** (const std::string &funcName) const
- virtual void **load** ()
- virtual void **unload** ()

Public Types

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

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**

4.12.2 Member Typedef Documentation

4.12.2.1 typedef Pointer< DynLoad, Mutex > Pointer

Type for pointer on this class.

Reimplemented in **Plugin** (p. 169), and **RendererPlugin** (p. 205).

4.12.3 Constructor & Destructor Documentation

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

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

Parameters:
← ***libName*** **Name**(p. 134) of the loaded library

4.12.3.2 ~DynLoad () [virtual]

The destructor. This class has virtual destructor.

4.12.4 Member Function Documentation

4.12.4.1 void * getFunction (const std::string & *funcName*) const [virtual]

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

Parameters:
← ***funcName*** **Name**(p. 134) of function

Returns:
Pointer(p. 177) to the function

4.12.4.2 void * getHandle () const [inline]

Getter of mHandle. Returns value of mHandle.

Returns:
The value of mHandle

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

Getter of mLibraryName. Returns value of mLibraryName.

Returns:
The value of mLibraryName

4.12.4.4 bool hasFunction (const std::string & *funcName*) const [virtual]

Return true when the dynamic library has a callable function with the specified name.

Parameters:
← ***funcName*** **Name**(p. 134) of function

Returns:
True when the dynamic library has a callable function with the specified name

4.12.4.5 void load () [protected, virtual]

Load a library. Called by constructor.

4.12.4.6 void unload () [protected, virtual]

Load a library. Called by constructor.

4.12.5 Member Data Documentation

4.12.5.1 void* mHandle [protected]

The dynamic library handler.

Remarks:
This is own attribute of this class.

4.12.5.2 std::string mLibraryName [protected]

The dynamic library name.

Remarks:
This is own attribute of this class.

4.13 Exception Class Reference

4.13.1 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, **SCRIPT_ERROR**, **XLIB_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

4.13.2 Member Enumeration Documentation

4.13.2.1 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.

SCRIPT_ERROR Error in a script file.

XLIB_ERROR Xlib error.

4.13.3 Constructor & Destructor Documentation

4.13.3.1 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. 134) of the file where the exception was thrown.

← ***functionName*** **Name**(p. 134) of the function where the exception was thrown.

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

4.13.4 Member Function Documentation

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

Getter of mDescription. Returns value of mDescription.

Returns:
The value of mDescription

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

Getter of mFileName. Returns value of mFileName.

Returns:
The value of mFileName

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

Getter of mFunctionName. Returns value of mFunctionName.

Returns:
The value of mFunctionName

4.13.4.4 Exception * getLastException () [inline, static]

Getter of mLastException. Returns value of mLastException.

Returns:
The value of mLastException

4.13.4.5 const u32 & getLineNumber () const [inline]

Getter of mLineNumber. Returns value of mLineNumber.

Returns:
The value of mLineNumber

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

Getter of mType. Returns value of mType.

Returns:
The value of mType

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

Translate type to human readable format.

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

Returns:

4.13.5 Member Data Documentation

4.13.5.1 std::string mDescription [protected]

Textual description of the exception.

Remarks:
This is own attribute of this class.

4.13.5.2 std::string mFileName [protected]

Name(p. 134) of the file where the exception was thrown.

Remarks:
This is own attribute of this class.

4.13.5.3 std::string mFunctionName [protected]

Name(p. 134) of the function where the exception was thrown.

Remarks:
This is own attribute of this class.

4.13.5.4 Exception * mLastException = 0 [static, protected]

Pointer(p. 177) to the last raised exception.

Remarks:
This attribute references an attribute.

4.13.5.5 u32 mLineNumber [protected]

Number of the line where the exception was thrown.

Remarks:
This is own attribute of this class.

4.13.5.6 ExceptionType mType [protected]

Type of the exception.

Remarks:
This is own attribute of this class.

4.14 FileSystemManager Class Reference

Inherits **Singleton**< **FileSystemManager** >.

4.14.1 Detailed Description

Singleton(p. 209) class managing file system. This class handles file opening, config file handling.

Methods

- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual bool **existsDirectory** (const std::string &name) const
- virtual bool **createDirectory** (const std::string &name) const
- virtual std::string **findDataDirectory** () const
- virtual bool **existsLibrary** (const std::string &library, const std::vector< std::string > &additionalPaths=std::vector< std::string >()) const
- virtual std::string **findLibraryPath** (const std::string &library, const std::vector< std::string > &additionalPaths=std::vector< std::string >()) const
- virtual bool **existsFile** (const std::string &name) const
- virtual bool **existsAppFile** (const std::string &name) const
- virtual **FileSystemManager::CFilePointer openFileC** (const std::string &name, const **FileOperation** &operation=FileSystemManager::READ) const
- virtual **FileSystemManager::CFilePointer openAppFileC** (const std::string &name, const **FileOperation** &operation=FileSystemManager::READ) const
- virtual **FileSystemManager::CFilePointer openDataFileC** (const std::string &name, const **FileOperation** &operation=FileSystemManager::READ) const
- virtual **FileSystemManager::CppFilePointer openFileCpp** (const std::string &name, const **FileOperation** &operation=FileSystemManager::READ) const
- virtual **FileSystemManager::CppFilePointer openAppFileCpp** (const std::string &name, const **FileOperation** &operation=FileSystemManager::READ) const
- virtual **FileSystemManager::CppFilePointer openDataFileCpp** (const std::string &name, const **FileOperation** &operation=FileSystemManager::READ) const
- virtual std::string **getPathDataFile** (const std::string &name) const
- virtual std::string **readTextFile** (const std::string &name) const
- virtual std::string **readAppTextFile** (const std::string &name) const
- virtual std::string **readDataTextFile** (const std::string &name) const
- virtual void **_findHomeDirectory** ()
- virtual std::string **_replaceHomeChar** (const std::string &path) const
- virtual std::string **_translateToAbsolutePath** (const std::string &path) const
- virtual void **_createAppDirectory** ()

Public Types

- typedef **Pointer**< FILE, **Mutex** > **CFilePointer**
- typedef **Pointer**< std::fstream, **Mutex** > **CppFilePointer**
- enum **FileOperation** { **READ**, **WRITE**, **APPEND** }

Public Member Functions

Constructors & destructor

- **FileSystemManager** (const std::string &appDirectory="~/.ParCompMark/", const std::string &iniFile="parcompmark.ini")
- virtual ~**FileSystemManager** ()

Getters & setters

- const bool & **getInitialized** () const
- const std::string & **getAppDirectory** () const
- void **setAppDirectory** (const std::string &appdirectory)
- const std::string & **getDataDirectory** () const
- void **setDataDirectory** (const std::string &datadirectory)
- const std::string & **getHomeDirectory** () const
- const std::string & **getCurrentDirectory** () const
- const std::string & **getIniFile** () const

Static Public Member Functions

Class methods

- static void **closeFile** (**FileSystemManager::CppFilePointer** &fp)
- static void **closeFile** (**FileSystemManager::CFilePointer** &fp)

Static Public Attributes

Class constants

- static const **u32 MAX_PATH** = 1024
- static const std::string **PATH_SEPARATOR** = "/"

Protected Attributes

Variables

- bool **mInitialized**
- std::string **mAppDirectory**
- std::string **mDataDirectory**
- std::string **mHomeDirectory**
- std::string **mCurrentDirectory**
- std::string **mIniFile**

4.14.2 Member Typedef Documentation

4.14.2.1 typedef Pointer< FILE, Mutex > CFilePointer

Type for pointer on a C-style file pointer.

4.14.2.2 typedef Pointer< std::fstream, Mutex > CppFilePointer

Type for pointer on a std::fstream object.

4.14.3 Member Enumeration Documentation

4.14.3.1 enum FileOperation

Type of file operation.

Enumerator:

READ Read.

WRITE Write.

APPEND Append.

4.14.4 Constructor & Destructor Documentation

4.14.4.1 FileSystemManager (const std::string & *appDirectory* = "∼/.ParCompMark/", const std::string & *iniFile* = "parcompmark.ini")

Creates file system manager.

Parameters:

← ***appDirectory*** Path of application directory.

← ***iniFile*** **Name**(p. 134) of user-level ini file.

4.14.4.2 ∼FileSystemManager () [virtual]

The destructor. This class has virtual destructor.

4.14.5 Member Function Documentation

4.14.5.1 void _createAppDirectory () [protected, virtual]

Create user-level application directory. Do nothing if already exists.

4.14.5.2 void _findHomeDirectory () [protected, virtual]

Find the home directory of the user on the filesystem.

4.14.5.3 std::string _replaceHomeChar (const std::string & *path*) const [protected, virtual]

Replace '∼' char in the specified path string.

Parameters:

← ***path*** Path string.

Returns:

Replaced path string.

4.14.5.4 std::string _translateToAbsolutePath (const std::string & *path*) const [protected, virtual]

Translate relative path to absolute (also replace '∼' char in the specified path string). Original code: http://www.codeproject.com/useritems/path_conversion.asp

Parameters:
← ***path*** Path string.

Returns:
Replaced path string.

4.14.5.5 void closeFile (FileSystemManager::CFilePointer & *fp*) [static]

Close file from C-style pointer. This method is static to be able to close the logfile.

Parameters:
→ ***fp*** Smart pointer on a C-style file pointer.

4.14.5.6 void closeFile (FileSystemManager::CppFilePointer & *fp*) [static]

Close file from fstream object. This method is static to be able to close the logfile.

Parameters:
→ ***fp*** Smart pointer on a fstream object.

4.14.5.7 bool createDirectory (const std::string & *name*) const [virtual]

Create directory.

Parameters:
← ***name*** **Name**(p. 134) of the directory.

Returns:
True if the directory did not exist, and had to be created.

4.14.5.8 bool existsAppFile (const std::string & *name*) const [virtual]

Return true if the specified file exists in the application directory.

Parameters:
← ***name*** **Name**(p. 134) of the file.

Returns:
True if the specified file exists.

4.14.5.9 bool existsDirectory (const std::string & *name*) const [virtual]

Return true if the specified directory exists.

Parameters:
← ***name*** **Name**(p. 134) of the directory.

Returns:
True if the specified directory exists.

4.14.5.10 bool existsFile (const std::string & *name*) const [virtual]

Return true if the specified file exists.

Parameters:
← ***name*** **Name**(p. 134) of the file.

Returns:
True if the specified file exists.

4.14.5.11 bool existsLibrary (const std::string & *library*, const std::vector< std::string > & *additionalPaths* = std::vector< std::string >()) const [virtual]

Returns true if the library can be found.

Parameters:
← ***library*** **Name**(p. 134) of the library file.

← ***additionalPaths*** Array if additional paths to seach

Returns:
The library can be found.

4.14.5.12 void finalize () [virtual]

Finalize the file system manager.

4.14.5.13 std::string findDataDirectory () const [virtual]

Find data directory path. Throw exception if not found.

Returns:
Path of data directory.

4.14.5.14 std::string findLibraryPath (const std::string & *library*, const std::vector< std::string > & *additionalPaths* = std::vector< std::string >()) const [virtual]

Find path for library. Throw exception if not found.

Parameters:

← ***library*** **Name**(p. 134) of the library file.

← ***additionalPaths*** Array if additional paths to seach

Returns:

Path of the library.

4.14.5.15 const std::string & getAppDirectory () const [inline]

Getter of mAppDirectory. Returns value of mAppDirectory.

Returns:

The value of mAppDirectory

4.14.5.16 const std::string & getCurrentDirectory () const [inline]

Getter of mCurrentDirectory. Returns value of mCurrentDirectory.

Returns:

The value of mCurrentDirectory

4.14.5.17 const std::string & getDataDirectory () const [inline]

Getter of mDataDirectory. Returns value of mDataDirectory.

Returns:

The value of mDataDirectory

4.14.5.18 const std::string & getHomeDirectory () const [inline]

Getter of mHomeDirectory. Returns value of mHomeDirectory.

Returns:

The value of mHomeDirectory

4.14.5.19 const std::string & getIniFile () const [inline]

Getter of mIniFile. Returns value of mIniFile.

Returns:

The value of mIniFile

4.14.5.20 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:
The value of mInitialized

4.14.5.21 std::string getPathDataFile (const std::string & *name*) const [virtual]

Get full path of the specified data file.

Parameters:
← ***name*** **Name**(p. 134) of the file.

Returns:
Full path for the data file.

4.14.5.22 void initialize () [virtual]

Initialize the file system manager.

4.14.5.23 FileSystemManager::CFilePointer openAppFileC (const std::string & *name*, const FileOperation & *operation* = FileSystemManager::READ) const [virtual]

Open file for reading or writing in the application directory and return a C-style file pointer.

Parameters:
← ***name*** **Name**(p. 134) of the file.

← ***operation*** File operation.

Returns:
Smart pointer on a C-style file pointer.

4.14.5.24 FileSystemManager::CppFilePointer openAppFileCpp (const std::string & *name*, const FileOperation & *operation* = FileSystemManager::READ) const [virtual]

Open file for reading or writing in the application directory and return a fstream object.

Parameters:
← ***name*** **Name**(p. 134) of the file.

← ***operation*** File operation.

Returns:
Smart pointer on a fstream object.

4.14.5.25 FileSystemManager::CFilePointer openDataFileC (const std::string & *name*, const FileOperation & *operation* = FileSystemManager::READ) const [virtual]

Open file for reading or writing in the application data directory and return a C-style file pointer.

Parameters:

← ***name*** **Name**(p. 134) of the file.

← ***operation*** File operation.

Returns:

Smart pointer on a C-style file pointer.

4.14.5.26 FileSystemManager::CppFilePointer openDataFileCpp (const std::string & *name*, const FileOperation & *operation* = FileSystemManager::READ) const [virtual]

Open file for reading or writing in the application data directory and return a fstream object.

Parameters:

← ***name*** **Name**(p. 134) of the file.

← ***operation*** File operation.

Returns:

Smart pointer on a fstream object.

4.14.5.27 FileSystemManager::CFilePointer openFileC (const std::string & *name*, const FileOperation & *operation* = FileSystemManager::READ) const [virtual]

Open file for reading or writing and return a C-style file pointer.

Parameters:

← ***name*** **Name**(p. 134) of the file.

← ***operation*** File operation.

Returns:

Smart pointer on a C-style file pointer.

4.14.5.28 FileSystemManager::CppFilePointer openFileCpp (const std::string & *name*, const FileOperation & *operation* = FileSystemManager::READ) const [virtual]

Open file for reading or writing and return a fstream object.

Parameters:

← ***name*** **Name**(p. 134) of the file.

← ***operation*** File operation.

Returns:

Smart pointer on a fstream object.

4.14.5.29 std::string readAppTextFile (const std::string & *name*) const [virtual]

Read contents of a text file in the application directory.

Parameters:
← ***name*** **Name**(p. 134) of the file.

Returns:
Content of the file.

4.14.5.30 std::string readDataTextFile (const std::string & *name*) const [virtual]

Read contents of a text file in the application data directory.

Parameters:
← ***name*** **Name**(p. 134) of the file.

Returns:
Content of the file.

4.14.5.31 std::string readTextFile (const std::string & *name*) const [virtual]

Read contents of a text file.

Parameters:
← ***name*** **Name**(p. 134) of the file.

Returns:
Content of the file.

4.14.5.32 void setAppDirectory (const std::string & *appdirectory*) [inline]

Setter of mAppDirectory. Sets value of mAppDirectory.

Parameters:
← ***appdirectory*** The value of mAppDirectory

4.14.5.33 void setDataDirectory (const std::string & *datadirectory*) [inline]

Setter of mDataDirectory. Sets value of mDataDirectory.

Parameters:
← ***datadirectory*** The value of mDataDirectory

4.14.6 Member Data Documentation

4.14.6.1 std::string mAppDirectory [protected]

Path of user-level application directory.

Remarks:
This is own attribute of this class.

4.14.6.2 const u32 MAX_PATH = 1024 [static]

Maximal number of chars in paths.

Remarks:
This is own attribute of this class.

4.14.6.3 std::string mCurrentDirectory [protected]

Path of current directory.

Remarks:
This is own attribute of this class.

4.14.6.4 std::string mDataDirectory [protected]

Path of application data directory.

Remarks:
This is own attribute of this class.

4.14.6.5 std::string mHomeDirectory [protected]

Home directory of the user.

Remarks:
This is own attribute of this class.

4.14.6.6 std::string mIniFile [protected]

Name(p. 134) of user-level ini file.

Remarks:
This is own attribute of this class.

4.14.6.7 bool mInitialized [protected]

The application is initialized.

Remarks:
This is own attribute of this class.

4.14.6.8 const std::string PATH_SEPARATOR = "/" [static]

Path separator on the host OS.

Remarks:
This is own attribute of this class.

4.15 GLXGLContext Class Reference

4.15.1 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
- const bool & **getTainted** () const
- void **setTainted** (const bool &tainted)

Methods

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

Protected Attributes

Variables

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

4.15.2 Member Typedef Documentation

4.15.2.1 typedef Pointer< GLXGLContext, Mutex > Pointer

Type for pointer on this class.

4.15.3 Constructor & Destructor Documentation

4.15.3.1 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

4.15.3.2 ~GLXGLContext () `[virtual]`

The destructor. This class has virtual destructor.

4.15.4 Member Function Documentation

4.15.4.1 void finalize () `[virtual]`

Finalize the GL context.

4.15.4.2 const XDisplay::Pointer & getDisplay () const `[inline]`

Getter of mDisplay. Returns value of mDisplay.

Returns:

The value of mDisplay

4.15.4.3 const ::GLXContext & getGLXContext () const `[inline]`

Getter of mGLXContext. Returns value of mGLXContext.

Returns:

The value of mGLXContext

4.15.4.4 GLXRenderWindow * getGLXWindow () const `[inline]`

Getter of mGLXWindow. Returns value of mGLXWindow.

Returns:

The value of mGLXWindow

4.15.4.5 const bool & getInitialized () const `[inline]`

Getter of mInitialized. Returns value of mInitialized.

Returns:

The value of mInitialized

4.15.4.6 const bool & getTainted () const [inline]

Getter of mTainted. Returns value of mTainted.

Returns:
The value of mTainted

4.15.4.7 inline::XVisualInfo * getVisualInfo () const

Getter of mVisualInfo. Returns value of mVisualInfo.

Returns:
The value of mVisualInfo

4.15.4.8 void initialize () [virtual]

Initialize the GL context.

4.15.4.9 void releaseCurrent () [virtual]

Release the current context without assigning a new one.

4.15.4.10 void setCurrent () [virtual]

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

4.15.4.11 void setTainted (const bool & *tainted*) [inline]

Setter of mTainted. Sets value of mTainted.

Parameters:
← ***tainted*** The value of mTainted

4.15.5 Member Data Documentation

4.15.5.1 XDisplay::Pointer mDisplay [protected]

Corresponding X Display.

Remarks:
This is own attribute of this class.

4.15.5.2 ::GLXContext mGLXContext [protected]

Wrapped GLX context.

Remarks:
This is own attribute of this class.

4.15.5.3 GLXRenderWindow* mGLXWindow [protected]

Corresponding GLX rendering window.

Remarks:
This attribute references an attribute.

4.15.5.4 bool mInitialized [protected]

The context is initialized.

Remarks:
This is own attribute of this class.

4.15.5.5 bool mTainted [protected]

The GLXContext attribute is maintaned by another code (In our case the PC library).

Remarks:
This is own attribute of this class.

4.15.5.6 ::XVisualInfo* mVisualInfo [protected]

Visual info for the context.

Remarks:
This is own attribute of this class.

4.16 GLXRenderWindow Class Reference

4.16.1 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 **releaseCurrent** ()
- 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
- static const **u32 DEFAULTWINDOWWIDTH** = 128
- static const **u32 DEFAULTWINDOWHEIGHT** = 128

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**

4.16.2 Member Typedef Documentation

4.16.2.1 typedef Pointer< GLXRenderWindow, Mutex > Pointer

Type for pointer on this class.

4.16.3 Constructor & Destructor Documentation

4.16.3.1 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!)

4.16.3.2 ~GLXRenderWindow () [virtual]

The destructor. This class has virtual destructor.

4.16.4 Member Function Documentation

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

Efficiently set window position (for internal use).

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

Efficiently set window sizes (for internal use).

4.16.4.3 void _setCaption () [protected, virtual]

Efficiently set window caption (for internal use).

4.16.4.4 void createWindow () [protected, virtual]

Create GLX Window. Protected method for internal use.

4.16.4.5 void destroyWindow () [protected, virtual]

Destroy GLX Window. Protected method for internal use.

4.16.4.6 void finalize () [virtual]

Finalize the GLX RenderWindow. Protected method for internal use.

4.16.4.7 void finishFrame () [inline, virtual]

Finish current frame.

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

Getter of mCaption. Returns value of mCaption.

Returns:
The value of mCaption

4.16.4.9 const u32 & getColourDepth () const [inline]

Getter of mColourDepth. Returns value of mColourDepth.

Returns:
The value of mColourDepth

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

Getter of mDisplay. Returns value of mDisplay.

Returns:
The value of mDisplay

4.16.4.11 const u32 & getFrameNumber () const [inline]

Getter of mFrameNumber. Returns value of mFrameNumber.

Returns:
The value of mFrameNumber

4.16.4.12 const u32 & getFSAASamples () const [inline]

Getter of mFSAASamples. Returns value of mFSAASamples.

Returns:
The value of mFSAASamples

4.16.4.13 const bool & getFullScreen () const [inline]

Getter of mFullScreen. Returns value of mFullScreen.

Returns:
The value of mFullScreen

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

Getter of mGLXGLContext. Returns value of mGLXGLContext.

Returns:
The value of mGLXGLContext

4.16.4.15 const u32 & getHeight () const [inline]

Getter of mHeight. Returns value of mHeight.

Returns:
The value of mHeight

4.16.4.16 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:
The value of mInitialized

4.16.4.17 const s32 & getLeft () const [inline]

Getter of mLeft. Returns value of mLeft.

Returns:
The value of mLeft

4.16.4.18 Process * getProcess () const [inline]

Getter of mProcess. Returns value of mProcess.

Returns:
The value of mProcess

4.16.4.19 const s32 & getTop () const [inline]

Getter of mTop. Returns value of mTop.

Returns:
The value of mTop

4.16.4.20 const bool & getVisible () const [inline]

Getter of mVisible. Returns value of mVisible.

Returns:
The value of mVisible

4.16.4.21 const u32 & getWidth () const [inline]

Getter of mWidth. Returns value of mWidth.

Returns:
The value of mWidth

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

Getter of mWindow. Returns value of mWindow.

Returns:
The value of mWindow

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

Getter of mWindowStatistics. Returns value of mWindowStatistics.

Returns:
The value of mWindowStatistics

4.16.4.24 void initialize () [virtual]

Initialize the GLX RenderWindow. Protected method for internal use.

4.16.4.25 void releaseCurrent () [virtual]

Disable the context of the window. Release the current context without assigning a new one.

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

Set new window position.

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

4.16.4.27 void resetStatistics () [protected, virtual]

Reset statistics. Protected method for internal use.

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

Set new window sizes.

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

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

Setter of mCaption. Sets value of mCaption.

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

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

Setter of mColourDepth. Sets value of mColourDepth.

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

4.16.4.31 void setCurrent () [virtual]

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

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

Setter of mFSAASamples. Sets value of mFSAASamples.

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

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

Setter of mFullScreen. Sets value of mFullScreen.

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

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

Setter of mHeight. Sets value of mHeight.

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

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

Setter of mLeft. Sets value of mLeft.

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

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

Setter of mTop. Sets value of mTop.

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

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

Setter of mVisible. Sets value of mVisible.

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

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

Setter of mWidth. Sets value of mWidth.

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

4.16.4.39 void startFrame () [inline, virtual]

Start a frame.

4.16.4.40 void updateStatistics () [protected, virtual]

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

4.16.5 Member Data Documentation

4.16.5.1 const s32 CENTERED = -1 [static]

Constant for centered position.

Remarks:
This is own attribute of this class.

4.16.5.2 const u32 DEFAULTWINDOWHEIGHT = 128 [static]

Default window height.

Remarks:
This is own attribute of this class.

4.16.5.3 const u32 DEFAULTWINDOWWIDTH = 128 [static]

Default window width.

Remarks:
This is own attribute of this class.

4.16.5.4 ::Atom mAtomDeleteWindow [protected]

Atom to recognize window close events.

Remarks:
This is own attribute of this class.

4.16.5.5 const u32 MAXIMALSIZE = 0 [static]

Constant for maximal size.

Remarks:
This is own attribute of this class.

4.16.5.6 std::string mCaption [protected]

Window caption.

Remarks:
This is own attribute of this class.

4.16.5.7 u32 mColourDepth [protected]

Colour depth of the window.

Remarks:
This is own attribute of this class.

4.16.5.8 XDisplay::Pointer mDisplay [protected]

Corresponding X Display.

Remarks:
This is own attribute of this class.

4.16.5.9 Real mFrameBeginTime [protected]

Time of stating a new frame.

Remarks:
This is own attribute of this class.

4.16.5.10 u32 mFrameNumber [protected]

Number of current frame.

Remarks:
This is own attribute of this class.

4.16.5.11 u32 mFSAASamples [protected]

Number of fullscreen antialasing samples.

Remarks:
This is own attribute of this class.

4.16.5.12 bool mFullScreen [protected]

The window appears in full screen mode.

Remarks:
This is own attribute of this class.

4.16.5.13 GLXGLContext::Pointer mGLXGLContext [protected]

GLX context of the render window.

Remarks:
This is own attribute of this class.

4.16.5.14 u32 mHeight [protected]

Vertical size of the window.

Remarks:
This is own attribute of this class.

4.16.5.15 bool mInitialized [protected]

The window is initialized.

Remarks:
This is own attribute of this class.

4.16.5.16 s32 mLeft [protected]

Horizontal position of the window.

Remarks:
This is own attribute of this class.

4.16.5.17 s32 mOriginalXRRConfiguation [protected]

For storing original XRR configuration mode.

Remarks:
This is own attribute of this class.

4.16.5.18 Process* mProcess [protected]

Corresponding process.

Remarks:
This attribute references an attribute.

4.16.5.19 Real mSumFPS [protected]

Summarized frame rates for all frames for calculating avgFPS statistics.

Remarks:
This is own attribute of this class.

4.16.5.20 Real mSumTriangleCount [protected]

Summarized triangle count for all frames for calculating avgTriangleCount statistics.

Remarks:
This is own attribute of this class.

4.16.5.21 s32 mTop [protected]

Vertical position of the window.

Remarks:
This is own attribute of this class.

4.16.5.22 bool mVisible [protected]

The window is visible.

Remarks:
This is own attribute of this class.

4.16.5.23 u32 mWidth [protected]

Horizontal size of the window.

Remarks:
This is own attribute of this class.

4.16.5.24 ::Window mWindow [protected]

Wrapped GLX Window.

Remarks:
This is own attribute of this class.

4.16.5.25 WindowStatistics mWindowStatistics [protected]

Current window statistics.

Remarks:
This is own attribute of this class.

4.16.5.26 const Real UNDEFINEDSTATISTICS = -1.0 [static]

Undefined statistics value.

Remarks:
This is own attribute of this class.

4.16.5.27 const s32 UNDEFINEDXRRCONFIGURATION = -1 [static]

Constant for undefined XRR configuration.

Remarks:
This is own attribute of this class.

4.17 GLXRenderWindow::WindowStatistics Struct Reference

4.17.1 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**

4.17.2 Member Data Documentation

4.17.2.1 Real avgFPS

Average frame rate

4.17.2.2 Real avgTriangleCount

Average triangle count

4.17.2.3 Real bestFPS

Frame rate for the frame with the best frame time

4.17.2.4 Real bestFrameTime

Best frame time

4.17.2.5 Real lastFPS

Frame rate for the last frame

4.17.2.6 Real lastFrameTime

Last frame time

4.17.2.7 Real lastTriangleCount

Triangle count for the last frame

4.17.2.8 Real maxTriangleCount

Maximal triangle count

4.17.2.9 Real minTriangleCount

Minimal triangle count

4.17.2.10 Real worstFPS

Frame rate for the frame with the worst frame time

4.17.2.11 Real worstFrameTime

Worst frame time

4.18 GPU Class Reference

Inherits **OutputNode**.

4.18.1 Detailed Description

Class contain **GPU**(p. 106) information.

Public Types

- typedef **ParCompMark::Pointer**< **GPU**, **Mutex** > **Pointer**

Public Member Functions

Constructors & destructor

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

Getters & setters

- const std::string & **getVendor** () const
- const std::string & **getModel** () const
- const **u32** & **getMemorySize** () const
- const **Real** & **getLoadPerSecond** () const

Methods

- virtual void **refreshData** ()

Protected Attributes

Variables

- std::string **mVendor**
- std::string **mModel**
- **u32 mMemorySize**
- **Real mLoadPerSecond**

4.18.2 Member Typedef Documentation

4.18.2.1 typedef ParCompMark::Pointer< GPU, Mutex > Pointer

Type for pointer on this class.

Reimplemented from **OutputNode** (p. 161).

4.18.3 Constructor & Destructor Documentation

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

Creates **GPU**(p. 106) node.

Parameters:

← ***name*** **Name**(p. 134) of the node.

← ***type*** Type of the node.

4.18.3.2 ~GPU () [virtual]

The destructor. This class has virtual destructor.

4.18.4 Member Function Documentation

4.18.4.1 const Real & getLoadPerSecond () const [inline]

Getter of mLoadPerSecond. Returns value of mLoadPerSecond.

Returns:

The value of mLoadPerSecond

4.18.4.2 const u32 & getMemorySize () const [inline]

Getter of mMemorySize. Returns value of mMemorySize.

Returns:

The value of mMemorySize

4.18.4.3 const std::string & getModel () const [inline]

Getter of mModel. Returns value of mModel.

Returns:

The value of mModel

4.18.4.4 const std::string & getVendor () const [inline]

Getter of mVendor. Returns value of mVendor.

Returns:

The value of mVendor

4.18.4.5 void refreshData () [virtual]

Refresh **GPU**(p. 106) datas.

Reimplemented from **OutputNode** (p. 164).

4.18.5 Member Data Documentation

4.18.5.1 Real mLoadPerSecond [protected]

Triangles/second or texels/second. **GPU**(p. 106) load=rendering capacity can be only get per second.

Remarks:
This is own attribute of this class.

4.18.5.2 u32 mMemorySize [protected]

GPU(p. 106) memory size.

Remarks:
This is own attribute of this class.

4.18.5.3 std::string mModel [protected]

GPU(p. 106) model name.

Remarks:
This is own attribute of this class.

4.18.5.4 std::string mVendor [protected]

Name(p. 134) of the vendor of the **GPU**(p. 106).

Remarks:
This is own attribute of this class.

4.19 HandleClient Class Reference

Inherits **Client**.

4.19.1 Detailed Description

Class for handle client messages.

Methods

- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual void **openConnection** ()
- virtual void **task** ()

Public Types

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

Public Member Functions

Constructors & destructor

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

4.19.2 Member Typedef Documentation

4.19.2.1 typedef ParCompMark::Pointer< HandleClient, DummyLock > Pointer

Type for pointer or this class.

Reimplemented from **Client** (p. 33).

4.19.3 Constructor & Destructor Documentation

4.19.3.1 HandleClient (const NetServer * *net*)

Create **HandleClient**(p. 109) class.

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

4.19.3.2 ~HandleClient () [virtual]

The destructor. This class has virtual destructor.

4.19.4 Member Function Documentation

4.19.4.1 void finalize () [virtual]

Initialize the class (broadcast recieve thread).

4.19.4.2 void initialize () [virtual]

Initialize the class (broadcast recieve thread).

4.19.4.3 void openConnection () [virtual]

Open TCP/IP connection (do nothing).

Reimplemented from **Client** (p. 35).

4.19.4.4 void task () [protected, virtual]

Handle a client.

Reimplemented from **Thread** (p. 230).

4.20 Host Class Reference

Inherits **Singleton**< **Host** >.

4.20.1 Detailed Description

Class for describe a hosts.

Public Types

- typedef **ParCompMark::Pointer**< int, **Mutex** > **IntPointer**

Public Member Functions

Constructors & destructor

- **Host** (const std::string &name="(unknown host)")
- virtual ~**Host** ()

Getters & setters

- const bool & **getWait** () const
- const std::string & **getName** () const
- void **setName** (const std::string &name)
- const std::string & **getLowLevelScript** () const
- void **setLowLevelScript** (const std::string &lowlevelscript)
- **Container**< **Node**, **Mutex** >::**Pointer** & **getNodes** ()
- const bool & **getInitialized** () const
- const **Real** & **getTimeCorrection** () const
- **OutputNode::Pointer** & **getOutputDocument** ()
- const PCid & **getSessionID** () const

Methods

- virtual **XDisplay::Pointer openXDisplay** (const std::string &displayName="")
- virtual void **openXDisplays** ()
- virtual void **closeXDisplays** ()
- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual **Node** * **createNode** (const char *nodeName)
- virtual void **start** ()
- virtual **u32 stop** ()
- virtual void **collectData** ()
- virtual void **setFrameID** (const **u32** &frameID)

Static Public Member Functions

Scripting binding

- static void **squirrelGlue** ()

Class methods

- static **Host** * **getInstance** ()

Static Public Attributes

Class constants

- static const std::string **HOSTINITNUT** = "scripts/framework/host-init.nut"

Protected Attributes

Variables

- bool **mWait**
- std::string **mName**
- **SqVM::Pointer mSqVM**
- std::string **mLowLevelScript**
- **Container**< **XDisplay**, **Mutex** >::**Pointer mXDisplays**
- **Container**< **Node**, **Mutex** >::**Pointer mNodes**
- bool **mInitialized**
- **Real mTimeCorrection**
- **OutputNode::Pointer mOutputDocument**
- PCid **mSessionID**

4.20.2 Member Typedef Documentation

4.20.2.1 typedef ParCompMark::Pointer< int, Mutex > IntPointer

Type for pointer to an int.

4.20.3 Constructor & Destructor Documentation

4.20.3.1 Host (const std::string & *name* = "(unknown host)")

Creates a host with a specified name.

Parameters:
← ***name*** **Name**(p. 134) of the host.

4.20.3.2 ~Host () [virtual]

The destructor. This class has virtual destructor.

4.20.4 Member Function Documentation

4.20.4.1 void closeXDisplays () [virtual]

Close all opened X displays.

4.20.4.2 void collectData () [virtual]

Collect data from nodes.

4.20.4.3 Node * createNode (const char * *nodeName*) [virtual]

Create node on this host.

Parameters:
← *nodeName* **Name**(p. 134) of the node (C string for squirrel compatibility).

Returns:
Pointer(p. 177) to the created node.

4.20.4.4 void finalize () [virtual]

Finalize the host.

4.20.4.5 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:
The value of mInitialized

4.20.4.6 Host * getInstance () [static]

Gets the instance of the singleton class. Reimplemented for squirrel usage.

Returns:
Instance of the class

Reimplemented from **Singleton**< **Host** > (p. 210).

4.20.4.7 const std::string & getLowLevelScript () const [inline]

Getter of mLowLevelScript. Returns value of mLowLevelScript.

Returns:
The value of mLowLevelScript

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

Getter of mName. Returns value of mName.

Returns:
The value of mName

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

Getter of mNodes. Returns value of mNodes.

Returns:
The value of mNodes

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

Getter of mOutputDocument. Returns value of mOutputDocument.

Returns:
The value of mOutputDocument

4.20.4.11 const PCid & getSessionID () const [inline]

Getter of mSessionID. Returns value of mSessionID.

Returns:
The value of mSessionID

4.20.4.12 const Real & getTimeCorrection () const [inline]

Getter of mTimeCorrection. Returns value of mTimeCorrection.

Returns:
The value of mTimeCorrection

4.20.4.13 const bool & getWait () const [inline]

Getter of mWait. Returns value of mWait.

Returns:
The value of mWait

4.20.4.14 void initialize () [virtual]

Initialize host from low level script.

4.20.4.15 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

4.20.4.16 void openXDisplays () [virtual]

Automatically scan available X displays on this host and open them.

4.20.4.17 void setFrameID (const u32 & *frameID*) [virtual]

Set the ending frameID of processes.

Parameters:
← ***frameID*** StopID.

4.20.4.18 void setLowLevelScript (const std::string & *lowlevelscript*) [inline]

Setter of mLowLevelScript. Sets value of mLowLevelScript.

Parameters:
← ***lowlevelscript*** The value of mLowLevelScript

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

Setter of mName. Sets value of mName.

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

4.20.4.20 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.

4.20.4.21 void start () [virtual]

Start the nodes.

4.20.4.22 u32 stop () [virtual]

Stop the nodes.

Returns:
frameID

4.20.5 Member Data Documentation

4.20.5.1 const std::string HOSTINITNUT = "scripts/framework/host-init.nut" [static]

Host(p. 111) initializer Squirrel script file (relative to the data directory).

Remarks:
This is own attribute of this class.

4.20.5.2 bool mInitialized [protected]

The host is initialized.

Remarks:
This is own attribute of this class.

4.20.5.3 std::string mLowLevelScript [protected]

Squirrel low level script for initialization.

Remarks:
This is own attribute of this class.

4.20.5.4 std::string mName [protected]

Name(p. 134) of the host (IP address).

Remarks:
This is own attribute of this class.

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

Nodes on this host.

Remarks:
This is own attribute of this class.

4.20.5.6 OutputNode::Pointer mOutputDocument [protected]

Root of the output document on this host.

Remarks:
This is own attribute of this class.

4.20.5.7 PCid mSessionID [protected]

PC session ID.

Remarks:
This is own attribute of this class.

4.20.5.8 SqVM::Pointer mSqVM [protected]

Squirrel virtual machine.

Remarks:
This is own attribute of this class.

4.20.5.9 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.

4.20.5.10 bool mWait [protected]

Node(p. 148) wait?.

Remarks:
This is own attribute of this class.

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

X Displays on this host.

Remarks:
This is own attribute of this class.

4.21 HostInfo Class Reference

Inherits **OutputNode**.

4.21.1 Detailed Description

Class for contain host information.

Public Types

- typedef **ParCompMark::HostInfo::NetIDName NetIDName**
- typedef **ParCompMark::Pointer**< **HostInfo**, **Mutex** > **Pointer**

Public Member Functions

Constructors & destructor

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

Getters & setters

- **Container**< **CPU**, **Mutex** >::**Pointer** & **getCPUs** ()
- **Container**< **GPU**, **Mutex** >::**Pointer** & **getGPUs** ()
- **Container**< **HostInfo::NetIDName**, **DummyLock** >::**Pointer** & **getNetIDNames** ()
- const std::string & **getPCLibVersion** () const
- const PCint & **getPCNumNetworks** () const
- const std::string & **getPCVendor** () const
- const std::string & **getPCExtensions** () const
- const PCint & **getVolatileFrameletLimit** () const
- const PCint & **getRetainOutputLimit** () const

Methods

- virtual void **refreshData** ()
- virtual void **refreshCPUs** (**OutputNode::Pointer** &cpus)
- virtual void **refreshGPUs** (**OutputNode::Pointer** &gpus)

Protected Attributes

Variables

- **Container**< **CPU**, **Mutex** >::**Pointer mCPUs**
- **Container**< **GPU**, **Mutex** >::**Pointer mGPUs**
- **Container**< **HostInfo::NetIDName**, **DummyLock** >::**Pointer mNetIDNames**
- std::string **mPCLibVersion**
- PCint **mPCNumNetworks**
- std::string **mPCVendor**
- std::string **mPCExtensions**
- PCint **mVolatileFrameletLimit**
- PCint **mRetainOutputLimit**

Classes

- struct **NetIDName**

4.21.2 Member Typedef Documentation

4.21.2.1 typedef struct ParCompMark::HostInfo::NetIDName NetIDName

Struct contain network layer and ID what PC can control.

4.21.2.2 typedef ParCompMark::Pointer< HostInfo, Mutex > Pointer

Type for pointer on this class.

Reimplemented from **OutputNode** (p. 161).

4.21.3 Constructor & Destructor Documentation

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

Creates a host with a specified name.

Parameters:

← ***name*** **Name**(p. 134) of the host.

← ***type*** Type of the host node.

4.21.3.2 ~HostInfo () [virtual]

The destructor. This class has virtual destructor.

4.21.4 Member Function Documentation

4.21.4.1 Container< CPU, Mutex >::Pointer & getCPUs () [inline]

Getter of mCPUs. Returns value of mCPUs.

Returns:

The value of mCPUs

4.21.4.2 Container< GPU, Mutex >::Pointer & getGPUs () [inline]

Getter of mGPUs. Returns value of mGPUs.

Returns:

The value of mGPUs

4.21.4.3 Container< HostInfo::NetIDName, DummyLock >::Pointer & getNetIDNames () [inline]

Getter of mNetIDNames. Returns value of mNetIDNames.

Returns:
The value of mNetIDNames

4.21.4.4 const std::string & getPCExtensions () const [inline]

Getter of mPCExtensions. Returns value of mPCExtensions.

Returns:
The value of mPCExtensions

4.21.4.5 const std::string & getPCLibVersion () const [inline]

Getter of mPCLibVersion. Returns value of mPCLibVersion.

Returns:
The value of mPCLibVersion

4.21.4.6 const PCint & getPCNumNetworks () const [inline]

Getter of mPCNumNetworks. Returns value of mPCNumNetworks.

Returns:
The value of mPCNumNetworks

4.21.4.7 const std::string & getPCVendor () const [inline]

Getter of mPCVendor. Returns value of mPCVendor.

Returns:
The value of mPCVendor

4.21.4.8 const PCint & getRetainOutputLimit () const [inline]

Getter of mRetainOutputLimit. Returns value of mRetainOutputLimit.

Returns:
The value of mRetainOutputLimit

4.21.4.9 const PCint & getVolatileFrameletLimit () const [inline]

Getter of mVolatileFrameletLimit. Returns value of mVolatileFrameletLimit.

Returns:
The value of mVolatileFrameletLimit

4.21.4.10 void refreshCPUs (OutputNode::Pointer & *cpus*) [virtual]

Refresh CPUs data on the hosts.

Parameters:
→ ***cpus*** CPUs root.

4.21.4.11 void refreshData () [virtual]

Refresh **Host**(p. 111) datas.

Reimplemented from **OutputNode** (p. 164).

4.21.4.12 void refreshGPUs (OutputNode::Pointer & *gpus*) [virtual]

Refresh GPUs data on the hosts.

Parameters:
→ ***gpus*** GPUs root.

4.21.5 Member Data Documentation

4.21.5.1 Container< CPU, Mutex >::Pointer mCPUs [protected]

CPUs on this host.

Remarks:
This is own attribute of this class.

4.21.5.2 Container< GPU, Mutex >::Pointer mGPUs [protected]

GPUs on this host.

Remarks:
This is own attribute of this class.

4.21.5.3 Container< HostInfo::NetIDName, DummyLock >::Pointer mNetIDNames [protected]

Network(p. 142) layers and IDs what PC can control.

Remarks:
This is own attribute of this class.

4.21.5.4 std::string mPCExtensions [protected]

Loaded PC extensions.

Remarks:
This is own attribute of this class.

4.21.5.5 std::string mPCLibVersion [protected]

PC library version.

Remarks:
This is own attribute of this class.

4.21.5.6 PCint mPCNumNetworks [protected]

Number of network transprot layers available for the PC API.

Remarks:
This is own attribute of this class.

4.21.5.7 std::string mPCVendor [protected]

PC vendor.

Remarks:
This is own attribute of this class.

4.21.5.8 PCint mRetainOutputLimit [protected]

The maximum value of retain output supported by implementation.

Remarks:
This is own attribute of this class.

4.21.5.9 PCint mVolatileFrameletLimit [protected]

The maximum value of volatile framelet supported by implementation.

Remarks:
This is own attribute of this class.

4.22 HostInfo::NetIDName Struct Reference

4.22.1 Detailed Description

Struct contain network layer and ID what PC can control.

Public Types

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

Public Attributes

- std::string **name**
- PCint **ID**

4.22.2 Member Typedef Documentation

4.22.2.1 typedef ParCompMark::Pointer< NetIDName, DummyLock > Pointer

Smart pointer on this struct

4.22.3 Member Data Documentation

4.22.3.1 PCint ID

ID of the given layer

4.22.3.2 std::string name

Network(p. 142) layer name

4.23 Lock Class Reference

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

4.23.1 Detailed Description

Lock(p. 124) interface. Provides lock, trylock and unlock methods.

Public Member Functions

Constructors & destructor

- **Lock** ()
- virtual ∼**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**

4.23.2 Constructor & Destructor Documentation

4.23.2.1 Lock () [inline]

Default constructor.

4.23.2.2 ∼Lock () [inline, virtual]

The destructor. This class has virtual destructor.

4.23.3 Member Function Documentation

4.23.3.1 const bool & getLocked () const [inline]

Getter of mLocked. Returns value of mLocked.

Returns:
The value of mLocked

4.23.3.2 virtual void lock () [pure virtual]

Lock(p. 124) the lock.

Implemented in **DummyLock** (p. 67), and **Mutex** (p. 132).

4.23.3.3 virtual bool trylock () [pure virtual]

Try locking the lock.

Returns:
True if the locking was successful.

Implemented in **DummyLock** (p. 67), and **Mutex** (p. 133).

4.23.3.4 virtual void unlock () [pure virtual]

Unlock the lock.

Implemented in **DummyLock** (p. 67), and **Mutex** (p. 133).

4.23.4 Member Data Documentation

4.23.4.1 bool mLocked [protected]

Indicates that the lock is locked.

Remarks:
This is own attribute of this class.

4.24 Logger Class Reference

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

4.24.1 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="parcompmark.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 **initialize** ()
- 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 std::string **translateException** (const **Exception** &exception)

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**

4.24.2 Member Enumeration Documentation

4.24.2.1 enum LogLevel

Definitions logging levels.

Enumerator:

FATAL Fatal error.

ERROR Error.

WARNING Warning.

NOTICE Notice.

DEBUG Debug.

4.24.3 Constructor & Destructor Documentation

4.24.3.1 Logger (const std::string & *logFileName* = "parcompmark.log")

Create logger.

Parameters:

← ***logFileName*** **Name**(p. 134) of the log file In the application directory.

4.24.3.2 ~Logger () [virtual]

The destructor. This class has virtual destructor.

4.24.4 Member Function Documentation

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

Getter of mConsoleLogLevel. Returns value of mConsoleLogLevel.

Returns:

The value of mConsoleLogLevel

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

Getter of mFileLogLevel. Returns value of mFileLogLevel.

Returns:
The value of mFileLogLevel

4.24.4.3 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:
The value of mInitialized

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

Getter of mLogFileName. Returns value of mLogFileName.

Returns:
The value of mLogFileName

4.24.4.5 const u8 & getLogMode () const [inline]

Getter of mLogMode. Returns value of mLogMode.

Returns:
The value of mLogMode

4.24.4.6 void initialize () [virtual]

Initializes the logger.

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

Logs an exception.

Parameters:
← ***exception*** **Exception**(p. 71).

4.24.4.8 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.

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

Logs a message. Handles multiline messages too.

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

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

Setter of mConsoleLogLevel. Sets value of mConsoleLogLevel.

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

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

Setter of mFileLogLevel. Sets value of mFileLogLevel.

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

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

Setter of mLogFileName. Sets value of mLogFileName.

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

4.24.4.13 std::string translateException (const Exception & *exception*) [inline, static]

Translate an exception to human readable.

Parameters:
← ***exception*** **Exception**(p. 71).

Returns:

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

Translate log level to human readable.

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

Returns:

4.24.5 Member Data Documentation

4.24.5.1 const u8 LOGTOCONSOLE = 1 [static]

Indicates that the logger logs to console.

Remarks:
This is own attribute of this class.

4.24.5.2 const u8 LOGTOFILE = 2 [static]

Indicates that the logger logs to a textfile.

Remarks:
This is own attribute of this class.

4.24.5.3 LogLevel mConsoleLogLevel [protected]

Logging level for console (from FATAL to NOTICE).

Remarks:
This is own attribute of this class.

4.24.5.4 LogLevel mFileLogLevel [protected]

Logging level for file (from FATAL to NOTICE).

Remarks:
This is own attribute of this class.

4.24.5.5 Pointer< FILE, Mutex > mFp [protected]

File pointer.

Remarks:
This is own attribute of this class.

4.24.5.6 bool mInitialized [protected]

Indicates that the logger is initialized.

Remarks:
This is own attribute of this class.

4.24.5.7 std::string mLogFileName [protected]

Name(p. 134) of the log file.

Remarks:
This is own attribute of this class.

4.24.5.8 u8 mLogMode [protected]

Logging mode (console and/or textfile).

Remarks:
This is own attribute of this class.

4.25 Mutex Class Reference

Inherits **Lock**.

4.25.1 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**

4.25.2 Constructor & Destructor Documentation

4.25.2.1 Mutex () [inline]

Default constructor.

4.25.2.2 ~Mutex () [inline, virtual]

The destructor. This class has virtual destructor.

4.25.3 Member Function Documentation

4.25.3.1 void lock () [inline, virtual]

Lock(p. 124) the mutex.

Implements **Lock** (p. 125).

4.25.3.2 bool trylock () [inline, virtual]

Try locking the mutex.

Returns:
True if the locking was successful.

Implements **Lock** (p. 125).

4.25.3.3 void unlock () [inline, virtual]

Unlock the mutex.

Implements **Lock** (p. 125).

4.25.4 Member Data Documentation

4.25.4.1 pthread_mutex_t mMutex [protected]

Guard mutex.

Remarks:
This is own attribute of this class.

4.26 Name Class Reference

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

4.26.1 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**

4.26.2 Constructor & Destructor Documentation

4.26.2.1 Name ()

Default constructor.

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

Default constructor.

Parameters:
← ***name*** **Name**(p. 134) of **Name**(p. 134):)

4.26.2.3 ~Name () `[virtual]`

The destructor. This class has virtual destructor.

4.26.3 Member Function Documentation

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

Getter of mName. Returns value of mName.

Returns:
The value of mName

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

Setter of mName. Sets value of mName.

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

4.26.4 Member Data Documentation

4.26.4.1 std::string mName [protected]

Name(p. 134).

Remarks:
This is own attribute of this class.

4.27 NetClient Class Reference

Inherits **Client**.

4.27.1 Detailed Description

Network(p. 142) client task.

Methods

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

Public Types

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

Public Member Functions

Constructors & destructor

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

4.27.2 Member Typedef Documentation

4.27.2.1 typedef ParCompMark::Pointer< NetClient, DummyLock > Pointer

Type for pointer to this class.

Reimplemented from **Client** (p. 33).

4.27.3 Constructor & Destructor Documentation

4.27.3.1 NetClient (const std::string & *name*)

Create network class.

Parameters:
← ***name*** **Name**(p. 134) of the network.

4.27.3.2 ~NetClient () [virtual]

The destructor. This class has virtual destructor.

4.27.4 Member Function Documentation

4.27.4.1 void finalize () [virtual]

Initialize the class (broadcast recieve thread).

4.27.4.2 void initialize () [virtual]

Initialize the class (broadcast recieve thread).

4.27.4.3 void task () [protected, virtual]

Thread(p. 226) for recieve broadcast message.

Reimplemented from **Thread** (p. 230).

4.28 NetServer Class Reference

Inherits **Network**.

4.28.1 Detailed Description

Network(p. 142) server class.

Methods

- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual void **sendBroadcastMessage** (const std::string &type, const std::string &message)
- virtual void **buildCluster** ()
- virtual void **task** ()

Public Types

- typedef **ParCompMark::Pointer**< **NetServer**, **DummyLock** > **Pointer**
- typedef **ParCompMark::Pointer**< int, **Mutex** > **IntPointer**

Public Member Functions

Constructors & destructor

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

Getters & setters

- const struct sockaddr_in & **getBroadcastAddress** () const
- const **u32** & **getMaxConnection** () const
- const **u32** & **getHostNumber** () const
- void **setHostNumber** (const **u32** &hostnumber)
- const **NetServer::IntPointer** & **getRecievedNumber** () const
- const **NetServer::IntPointer** & **getStopFrameID** () const

Protected Attributes

Variables

- sockaddr_in **mBroadcastAddress**
- **u32 mMaxConnection**
- **u32 mHostNumber**
- **IntPointer mRecievedNumber**
- **IntPointer mStopFrameID**

4.28.2 Member Typedef Documentation

4.28.2.1 typedef ParCompMark::Pointer< int, Mutex > IntPointer

Type for pointer to an int.

4.28.2.2 typedef ParCompMark::Pointer< NetServer, DummyLock > Pointer

Type for pointer to this class.

Reimplemented from **Network** (p. 143).

4.28.3 Constructor & Destructor Documentation

4.28.3.1 NetServer (const std::string & *name*)

Create network class.

Parameters:
← ***name*** **Name**(p. 134) of the network.

4.28.3.2 ~NetServer () [virtual]

The destructor. This class has virtual destructor.

4.28.4 Member Function Documentation

4.28.4.1 void buildCluster () [virtual]

Build cluster information.

4.28.4.2 void finalize () [virtual]

Initialize the class (broadcast recieve thread).

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

Getter of mBroadcastAddress. Returns value of mBroadcastAddress.

Returns:
The value of mBroadcastAddress

4.28.4.4 const u32 & getHostNumber () const [inline]

Getter of mHostNumber. Returns value of mHostNumber.

Returns:
The value of mHostNumber

4.28.4.5 const u32 & getMaxConnection () const [inline]

Getter of mMaxConnection. Returns value of mMaxConnection.

Returns:
The value of mMaxConnection

4.28.4.6 const NetServer::IntPointer & getRecievedNumber () const [inline]

Getter of mRecievedNumber. Returns value of mRecievedNumber.

Returns:
The value of mRecievedNumber

4.28.4.7 const NetServer::IntPointer & getStopFrameID () const [inline]

Getter of mStopFrameID. Returns value of mStopFrameID.

Returns:
The value of mStopFrameID

4.28.4.8 void initialize () [virtual]

Initialize the class (server thread, broadcast send).

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

Send a broadcast message.

Parameters:
← ***type*** Type of the message.
← ***message*** The message.

4.28.4.10 void setHostNumber (const u32 & *hostnumber*) [inline]

Setter of mHostNumber. Sets value of mHostNumber.

Parameters:
← ***hostnumber*** The value of mHostNumber

4.28.4.11 void task () [protected, virtual]

Thread(p. 226) for accept TCP/IP connection.

Reimplemented from **Thread** (p. 230).

4.28.5 Member Data Documentation

4.28.5.1 struct sockaddr_in mBroadcastAddress [protected]

The broadcast address structure for send later time.

Remarks:
This is own attribute of this class.

4.28.5.2 u32 mHostNumber [protected]

Number if the host in the cluster.

Remarks:
This is own attribute of this class.

4.28.5.3 u32 mMaxConnection [protected]

The broadcast address structure for send later time.

Remarks:
This is own attribute of this class.

4.28.5.4 IntPointer mRecievedNumber [protected]

Number of the recieved message for a broadcast.

Remarks:
This is own attribute of this class.

4.28.5.5 IntPointer mStopFrameID [protected]

Frame id for stop PC.

Remarks:
This is own attribute of this class.

4.29 Network Class Reference

Inherits **Thread**.

Inherited by **Client**, and **NetServer**.

4.29.1 Detailed Description

Class for network routines.

Getters & setters

- const bool & **getInitialized** () const
- const int & **getStreamSocket** () const
- void **setStreamSocket** (const int &streamsocket)
- const int & **getBroadcastSocket** () const
- const std::string & **getOwnIP** () const
- static **Container**< **Network::IfConf**, **DummyLock** >::**Pointer** & **getIfConfs** ()
- static const unsigned short & **getCommunicationPort** ()
- static void **setCommunicationPort** (const unsigned short &communicationport)
- static const unsigned short & **getBroadcastPort** ()
- static void **setBroadcastPort** (const unsigned short &broadcastport)

Public Types

- typedef **Pointer**< **Network**, **DummyLock** > **Pointer**
- typedef **ParCompMark::Network::IfConf IfConf**

Public Member Functions

Constructors & destructor

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

Static Public Member Functions

Class methods

- static void **getIP** ()
- static std::string **getHostName** ()

Protected Attributes

Variables

- bool **mInitialized**
- int **mStreamSocket**
- int **mBroadcastSocket**
- std::string **mOwnIP**

Static Protected Attributes

Class variables

- static **Container**< **Network::IfConf**, **DummyLock** >::**Pointer mIfConfs**
- static unsigned short **mCommunicationPort** = 1234
- static unsigned short **mBroadcastPort** = 1235

Classes

- struct **IfConf**

4.29.2 Member Typedef Documentation

4.29.2.1 typedef struct ParCompMark::Network::IfConf IfConf

Struct for network interfaces IP information.

4.29.2.2 typedef Pointer< Network, DummyLock > Pointer

Type for pointer to this class.

Reimplemented in **Client** (p. 33), **HandleClient** (p. 109), **NetClient** (p. 136), and **NetServer** (p. 139).

4.29.3 Constructor & Destructor Documentation

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

Create network class.

Parameters:
← ***name*** **Name**(p. 134) of the network.

4.29.3.2 ~Network () `[virtual]`

The destructor. This class has virtual destructor.

4.29.4 Member Function Documentation

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

Getter of mBroadcastPort. Returns value of mBroadcastPort.

Returns:
The value of mBroadcastPort

4.29.4.2 const int & getBroadcastSocket () const [inline]

Getter of mBroadcastSocket. Returns value of mBroadcastSocket.

Returns:
The value of mBroadcastSocket

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

Getter of mCommunicationPort. Returns value of mCommunicationPort.

Returns:
The value of mCommunicationPort

4.29.4.4 std::string getHostName () [static]

Get local machine name.

Returns:
name of host

4.29.4.5 Container< Network::IfConf, DummyLock >::Pointer & getIfConfs () [inline, static]

Getter of mIfConfs. Returns value of mIfConfs.

Returns:
The value of mIfConfs

4.29.4.6 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:
The value of mInitialized

4.29.4.7 void getIP () [static]

Get local machine IP address-es.

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

Getter of mOwnIP. Returns value of mOwnIP.

Returns:
The value of mOwnIP

4.29.4.9 const int & getStreamSocket () const [inline]

Getter of mStreamSocket. Returns value of mStreamSocket.

Returns:
The value of mStreamSocket

4.29.4.10 void setBroadcastPort (const unsigned short & *broadcastport*) [inline, static]

Setter of mBroadcastPort. Sets value of mBroadcastPort.

Parameters:
← ***broadcastport*** The value of mBroadcastPort

4.29.4.11 void setCommunicationPort (const unsigned short & *communicationport*) [inline, static]

Setter of mCommunicationPort. Sets value of mCommunicationPort.

Parameters:
← ***communicationport*** The value of mCommunicationPort

4.29.4.12 void setStreamSocket (const int & *streamsocket*) [inline]

Setter of mStreamSocket. Sets value of mStreamSocket.

Parameters:
← ***streamsocket*** The value of mStreamSocket

4.29.5 Member Data Documentation

4.29.5.1 unsigned short mBroadcastPort = 1235 [static, protected]

The broadcast port number.

Remarks:
This is own attribute of this class.

4.29.5.2 int mBroadcastSocket [protected]

The broadcast socket number.

Remarks:
This is own attribute of this class.

4.29.5.3 unsigned short mCommunicationPort = 1234 [static, protected]

The communication port number.

Remarks:
This is own attribute of this class.

4.29.5.4 Container< Network::IfConf, DummyLock >::Pointer mIfConfs [static, protected]

Configurations of network interfaces of a host.

Remarks:
This is own attribute of this class.

4.29.5.5 bool mInitialized [protected]

Indicates initialized the class.

Remarks:
This is own attribute of this class.

4.29.5.6 std::string mOwnIP [protected]

IP address of the master node.

Remarks:
This is own attribute of this class.

4.29.5.7 int mStreamSocket [protected]

The client socket number.

Remarks:
This is own attribute of this class.

4.30 Network::IfConf Struct Reference

4.30.1 Detailed Description

Struct for network interfaces IP information.

Public Types

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

Public Attributes

- std::string **IP**
- std::string **Netmask**
- std::string **BroadcastIP**

4.30.2 Member Typedef Documentation

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

Smart pointer on this struct

4.30.3 Member Data Documentation

4.30.3.1 std::string BroadcastIP

Broacast IP of the network card.

4.30.3.2 std::string IP

IP address of the network card.

4.30.3.3 std::string Netmask

Netmask of the network card.

4.31 Node Class Reference

Inherits **Name**.

4.31.1 Detailed Description

Node(p. 148) class. Entity for composite and/or render a framelet.

Public Types

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

Public Member Functions

Constructors & destructor

- **Node** ()
- **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** ()
- const bool & **getInitialized** () const

Methods

- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual **Process** * **createProcess** (const char *processName)
- virtual **Buffer** * **createBuffer** (const char *name, const PCuint left, const PCuint top, const PCuint width, const PCuint height, const PCuint depthFormat)
- virtual **Buffer::Pointer getBuffer** (const std::string name)
- virtual void **start** ()
- virtual **u32 stop** ()
- virtual void **collectData** ()
- virtual void **setFrameID** (const **u32** &frameID)

Static Public Member Functions

Scripting binding

- static void **squirrelGlue** ()

Protected Attributes

Variables

- **Host** ∗ **mParent**
- PCstring **mSearchPath**
- **Container**< **Process**, **Mutex** >::**Pointer mProcesses**
- **OutputNode::Pointer mOutputDocument**
- **Container**< **Buffer**, **Mutex** >::**Pointer mBuffers**
- bool **mInitialized**

4.31.2 Member Typedef Documentation

4.31.2.1 typedef Pointer< Node, Mutex > Pointer

Type for pointer on this class.

4.31.3 Constructor & Destructor Documentation

4.31.3.1 Node ()

Default constructor for Squirrel compatibility. Calling this constructor always raises an exception.

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

Create node. Normally **Host**(p. 111) calls this constructor.

Parameters:

← ***name*** **Name**(p. 134) of the node.

← ***parent*** Parent host

4.31.3.3 ∼Node () [virtual]

The destructor. This class has virtual destructor.

4.31.4 Member Function Documentation

4.31.4.1 void collectData () [virtual]

Collect data from nodes.

4.31.4.2 Buffer ∗ createBuffer (const char ∗ *name*, const PCuint *left*, const PCuint *top*, const PCuint *width*, const PCuint *height*, const PCuint *depthFormat*) [virtual]

Create buffer on this node.

Parameters:

← ***name*** **Name**(p. 134) of the buffer

← ***left*** X offset of the image (it is 0 for whole images).

← ***top*** Y offset of the image (it is 0 for whole images).

← ***width*** Width of the image.

← ***height*** Height of the image.

← ***depthFormat*** Depth format used by this image.

Returns:
Pointer(p. 177) to the created buffer.

4.31.4.3 Process * createProcess (const char * *processName*) [virtual]

Create process on this node.

Parameters:
← ***processName*** **Name**(p. 134) of the process (C string for squirrel compatibility).

Returns:
Pointer(p. 177) to the created process.

4.31.4.4 void finalize () [virtual]

Finalize the node.

4.31.4.5 Buffer::Pointer getBuffer (const std::string *name*) [virtual]

Get buffer by name.

Parameters:
← ***name*** **Name**(p. 134) of the buffer

Returns:
Found buffer.

4.31.4.6 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:
The value of mInitialized

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

Getter of mOutputDocument. Returns value of mOutputDocument.

Returns:
The value of mOutputDocument

4.31.4.8 Host * getParent () const [inline]

Getter of mParent. Returns value of mParent.

Returns:
The value of mParent

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

Getter of mProcesses. Returns value of mProcesses.

Returns:
The value of mProcesses

4.31.4.10 const PCstring & getSearchPath () const [inline]

Getter of mSearchPath. Returns value of mSearchPath.

Returns:
The value of mSearchPath

4.31.4.11 void initialize () [virtual]

Initialize the node.

4.31.4.12 void setFrameID (const u32 & *frameID*) [virtual]

Set the ending frameID of processes.

Parameters:
← ***frameID*** StopID.

4.31.4.13 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.

4.31.4.14 void start () [virtual]

Start the processes.

4.31.4.15 u32 stop () [virtual]

Stop the processes.

Returns:
frameID

4.31.5 Member Data Documentation

4.31.5.1 Container< Buffer, Mutex >::Pointer mBuffers [protected]

Grapics memory buffers on this node.

Remarks:
This is own attribute of this class.

4.31.5.2 bool mInitialized [protected]

The node is initialized.

Remarks:
This is own attribute of this class.

4.31.5.3 OutputNode::Pointer mOutputDocument [protected]

Node(p. 148) output document.

Remarks:
This is own attribute of this class.

4.31.5.4 Host* mParent [protected]

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

Remarks:
This attribute references an attribute.

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

Processes on this node.

Remarks:
This is own attribute of this class.

4.31.5.6 PCstring mSearchPath [protected]

Search path of PC library.

Remarks:
This is own attribute of this class.

4.32 OpenGLRenderingEngine Class Reference

4.32.1 Detailed Description

Collection of rendering methods implemented using OpenGL API.

Methods

- virtual **Renderer** ∗ **createCustomRenderer** (const char ∗rendererName)
- virtual void **perspective** (const double fovy, const double zNear, const double zFar)
- virtual void **ortho2D** (const double left, const double right, const double bottom, const double top)
- virtual void **viewport** (const double left, const double top, const double width, const double height)
- virtual void **translate** (const double x, const double y, const double z)
- virtual void **rotate** (const double angle, const double x, const double y, const double z)
- virtual void **scale** (const double x, const double y, const double z)
- virtual void **setCameraPosition** (const double eyeX, const double eyeY, const double eyeZ)
- virtual void **setCameraTarget** (const double centerX, const double centerY, const double centerZ)
- virtual void **setCameraUpVector** (const double upX, const double upY, const double upZ)
- virtual void **drawTriangle** ()
- virtual void **drawSphere** ()
- virtual void **drawCylinder** ()
- virtual void **_refreshCamera** ()

Public Types

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

Public Member Functions

Constructors & destructor

- **OpenGLRenderingEngine** ()
- **OpenGLRenderingEngine** (**Process** ∗parent)
- virtual ∼**OpenGLRenderingEngine** ()

Getters & setters

- **Process** ∗ **getParent** () const
- const **OpenGLRenderingEngine::Camera** & **getCamera** () const

Static Public Member Functions

Scripting binding

- static void **squirrelGlue** ()

Class methods

- static **OpenGLRenderingEngine** ∗ **create** (const char ∗stringPtr)
- static void **loadOpenGLExtensions** (const std::list< std::string > &extensions)

Protected Types

- typedef **ParCompMark::OpenGLRenderingEngine::Camera Camera**

Protected Attributes

Variables

- **Process** $*$ **mParent**
- **Camera mCamera**

Classes

- struct **Camera**

4.32.2 Member Typedef Documentation

4.32.2.1 typedef struct ParCompMark::OpenGLRenderingEngine::Camera Camera [protected]

Camera(p. 159) type.

4.32.2.2 typedef Pointer< OpenGLRenderingEngine, Mutex > Pointer

Type for pointer on this class.

4.32.3 Constructor & Destructor Documentation

4.32.3.1 OpenGLRenderingEngine ()

Default constructor for Squirrel compatibility. Calling this constructor always raises an exception.

4.32.3.2 OpenGLRenderingEngine (Process $*$ *parent*)

Create rendering engine for the specified process.

Parameters:
$\leftarrow$ ***parent*** Parent process

4.32.3.3 ~OpenGLRenderingEngine () [virtual]

The destructor. This class has virtual destructor.

4.32.4 Member Function Documentation

4.32.4.1 void _refreshCamera () [protected, virtual]

Refresh camera setting in OpenGL.

4.32.4.2 OpenGLRenderingEngine * create (const char * *stringPtr*) [static]

Create (cast) rendering engine from a pointer passed in a string in '0x000000' format.

Parameters:
← ***stringPtr*** A pointer passed in a string in '0x000000' format.

Returns:
Cast rendering engine object.

4.32.4.3 Renderer * createCustomRenderer (const char * *rendererName*) [virtual]

Create custom renderer from a renderer plugin.

Parameters:
← ***rendererName*** **Name**(p. 134) of the renderer plugin.

Returns:
The created renderer.

4.32.4.4 void drawCylinder () [virtual]

Draw a cylinder.

4.32.4.5 void drawSphere () [virtual]

Draw a sphere.

4.32.4.6 void drawTriangle () [virtual]

Draw a triangle.

4.32.4.7 const OpenGLRenderingEngine::Camera & getCamera () const [inline]

Getter of mCamera. Returns value of mCamera.

Returns:
The value of mCamera

4.32.4.8 Process * getParent () const [inline]

Getter of mParent. Returns value of mParent.

Returns:
The value of mParent

4.32.4.9 void loadOpenGLExtensions (const std::list< std::string > & *extensions*) [static]

Load OpenGL extensions in the specified list. One extensions is loaded only once.

Parameters:

← ***extensions*** List of OpenGL extensions to load.

4.32.4.10 void ortho2D (const double *left*, const double *right*, const double *bottom*, const double *top*) [virtual]

Define a 2D orthographic projection.

Parameters:

← ***left*** Specifies the coordinate for the left vertical clipping plane.

← ***right*** Specifies the coordinate for the right vertical clipping plane.

← ***bottom*** Specifies the coordinate for the bottom horizontal clipping plane.

← ***top*** Specifies the coordinate for the top horizontal clipping plane.

4.32.4.11 void perspective (const double *fovy*, const double *zNear*, const double *zFar*) [virtual]

Set perspective projection.

Parameters:

← ***fovy*** Specifies the field of view angle, in degrees, in the y direction.

← ***zNear*** Specifies the distance from the viewer to the near clipping plane (always positive).

← ***zFar*** Specifies the distance from the viewer to the far clipping plane (always positive).

4.32.4.12 void rotate (const double *angle*, const double *x*, const double *y*, const double *z*) [virtual]

Rotate the modelview matrix.

Parameters:

← ***angle*** Rotation angle in degrees.

← ***x*** X coordinate of the rotation vector.

← ***y*** Y coordinate of the rotation vector.

← ***z*** Z coordinate of the rotation vector.

4.32.4.13 void scale (const double *x*, const double *y*, const double *z*) [virtual]

Scale the modelview matrix.

Parameters:

← ***x*** X coordinate of the scale vector.

← ***y*** Y coordinate of the scale vector.

← ***z*** Z coordinate of the scale vector.

4.32.4.14 void setCameraPosition (const double *eyeX*, const double *eyeY*, const double *eyeZ*) [virtual]

Set camera position.

Parameters:

← ***eyeX*** Specifies the X coordinate of position of the eye point.

← ***eyeY*** Specifies the Y coordinate of position of the eye point.

← ***eyeZ*** Specifies the Z coordinate of position of the eye point.

4.32.4.15 void setCameraTarget (const double *centerX*, const double *centerY*, const double *centerZ*) [virtual]

Set position of camera target.

Parameters:

← ***centerX*** Specifies the X coordinate of position of the reference point.

← ***centerY*** Specifies the Y coordinate of position of the reference point.

← ***centerZ*** Specifies the Z coordinate of position of the reference point.

4.32.4.16 void setCameraUpVector (const double *upX*, const double *upY*, const double *upZ*) [virtual]

Set up vector of the camera.

Parameters:

← ***upX*** Specifies the X coordinate of the direction of the up vector.

← ***upY*** Specifies the Y coordinate of the direction of the up vector.

← ***upZ*** Specifies the Z coordinate of the direction of the up vector.

4.32.4.17 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.

4.32.4.18 void translate (const double *x*, const double *y*, const double *z*) [virtual]

Translate the modelview matrix.

Parameters:

← ***x*** X coordinate of the translation vector.

← ***y*** Y coordinate of the translation vector.

← ***z*** Z coordinate of the translation vector.

4.32.4.19 void viewport (const double *left*, const double *top*, const double *width*, const double *height*) [virtual]

Set viewport with window relative coordinates (0.0 .. 1.0).

Parameters:

← ***left*** Left coordinate of the viewport.

← ***top*** Top coordinate of the viewport.

← ***width*** Width of the viewport.

← ***height*** Height of the viewport.

4.32.5 Member Data Documentation

4.32.5.1 Camera mCamera [protected]

Camera(p. 159) of this rendering engine.

Remarks:

This attribute references an attribute.

4.32.5.2 Process* mParent [protected]

Parent process of the rendering engine.

Remarks:

This attribute references an attribute.

4.33 OpenGLRenderingEngine::Camera Struct Reference

4.33.1 Detailed Description

Camera(p. 159) type.

Public Member Functions

- **Camera** ()

4.33.2 Constructor & Destructor Documentation

4.33.2.1 Camera () [inline]

Set default values

4.33.3 Member Data Documentation

4.33.3.1 Real position[3]

4.33.3.2 Real target[3]

4.33.3.3 Real upVector[3]

4.34 OutputNode Class Reference

Inherits **Name**.

Inherited by **CPU**, **GPU**, and **HostInfo**.

4.34.1 Detailed Description

Node(p. 148) 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** ()

Class methods

- static **OutputNode::Pointer parseFromXML** (std::string &xmlDocument)
- static bool **_testXMLName** (const std::string &name)
- static std::string **_convertSpecialChars** (const std::string &string)

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** ()
- virtual void **refreshData** ()
- virtual void **clean** ()

Protected Attributes

Variables

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

Static Protected Attributes

Class constants

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

4.34.2 Member Typedef Documentation

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

Type definition for map of attributes.

4.34.2.2 typedef AttributeMap::const_iterator AttributeMapIterator

Type definition for iterator on map of attributes.

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

Type definition for list of child nodes.

4.34.2.4 typedef ChildNodeList::iterator ChildNodeListIterator

Type definition for iterator on list of child nodes.

4.34.2.5 typedef Pointer< OutputNode, Mutex > Pointer

Type for pointer on this class.

Reimplemented in **CPU** (p. 63), **GPU** (p. 106), and **HostInfo** (p. 119).

4.34.3 Member Enumeration Documentation

4.34.3.1 enum NodeType

Node(p. 148) 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.

4.34.4 Constructor & Destructor Documentation

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

Creates output node.

Parameters:

← ***name*** **Name**(p. 134) of the node.

← ***type*** Type of the node.

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

Creates text field.

Parameters:

← ***text*** Text data.

4.34.4.3 ~OutputNode () `[virtual]`

The destructor. This class has virtual destructor.

4.34.5 Member Function Documentation

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

Convert special characters to XML entites.

Parameters:

← ***string*** String to convert.

Returns:

Converted string

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

Test name if its a correct xml name.

Parameters:
← ***name*** **Name**(p. 134) to test.

Returns:
True if the specified name is a correct XML name

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

Add child node.

Parameters:
← ***child*** Child node.

4.34.5.4 void clean () [virtual]

Clean outputnode.

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

Create textual child node.

Parameters:
← ***text*** Text data.

Returns:
Created child node

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

Create nontextual child node.

Parameters:
← ***name*** **Name**(p. 134) of the child node.
← ***type*** Type of the child node.

Returns:
Created child node

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

Get an attribute by name.

Parameters:
← ***attribute*** **Name**(p. 134) of the attribute.

Returns:
Attribute value.

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

Getter of TEXTNODENAME. Returns value of TEXTNODENAME.

Returns:
The value of TEXTNODENAME

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

Getter of mText. Returns value of mText.

Returns:
The value of mText

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

Getter of mType. Returns value of mType.

Returns:
The value of mType

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

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

Parameters:
← ***attribute*** **Name**(p. 134) of the attribute.

Returns:
The node has the specified attribute.

4.34.5.12 OutputNode::Pointer parseFromXML (std::string & *xmlDocument*) [static]

Parse XML document and construct its **OutputNode**(p. 160) equivalent.

Parameters:
→ ***xmlDocument*** XML document or fragment to parse.

Returns:
Root node of the parsed document

4.34.5.13 void refreshData () [virtual]

Refresh datas.

Reimplemented in **CPU** (p. 64), **GPU** (p. 108), and **HostInfo** (p. 121).

4.34.5.14 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.

4.34.5.15 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).

4.34.5.16 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. 134) of the attribute.
← ***value*** Value of the attribute.

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

Setter of mText. Sets value of mText.

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

4.34.6 Member Data Documentation

4.34.6.1 AttributeMap mAttributes [protected]

Attributes of the node.

Remarks:
This is own attribute of this class.

4.34.6.2 ChildNodeList mChildren [protected]

Child nodes.

Remarks:
This is own attribute of this class.

4.34.6.3 std::string mText [protected]

Text data.

Remarks:
This is own attribute of this class.

4.34.6.4 NodeType mType [protected]

Type of the node.

Remarks:
This is own attribute of this class.

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

String constant for node name for text data nodes.

Remarks:
This is own attribute of this class.

4.35 Plugin Class Reference

Inherits **DynLoad**, and **Name**.

Inherited by **RendererPlugin**.

4.35.1 Detailed Description

Application(p. 9) plugin.

Methods

- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual void **_initializeSpecific** ()=0
- virtual void **_finalizeSpecific** ()=0
- virtual void **_getNeededLibs** ()
- virtual void **_setPluginHandle** ()
- virtual void **_setLoggerFunction** ()
- virtual void **_onLoad** ()
- virtual void **_onUnload** ()
- virtual const char * **_getErrorMsg** ()
- virtual void **_checkError** ()

Public Types

- typedef **ParCompMark::Pointer**< **Plugin**, **Mutex** > **Pointer**
- enum **PluginType** { **BASIC_PLUGIN**, **RENDERER_PLUGIN** }

Public Member Functions

Constructors & destructor

- **Plugin** (const **PluginType** &type, const std::string &name, const std::string &filename)
- virtual ~**Plugin** ()

Getters & setters

- const **Plugin::PluginType** & **getPluginType** () const
- const bool & **getInitialized** () const
- const std::list< std::string > & **getNeededLibs** () const
- const **s32** & **getLastError** () const

Protected Types

- typedef const char **(* **pcmGetNeededLibsType**)()
- typedef void(* **loggerFunctionType**)(const void *, const char *)
- typedef int(* **pcmSetPluginHandleType**)(void *)
- typedef int(* **pcmSetLoggerFunctionType**)(**loggerFunctionType**)
- typedef int(* **pcmOnLoadType**)()
- typedef int(* **pcmOnUnloadType**)()
- typedef const char *(* **pcmGetErrorMsgType**)(const int)

Static Protected Member Functions

Class methods

- static void **_loggerFunction** (const void *handle, const char *message)

Protected Attributes

Variables

- **PluginType mPluginType**
- bool **mInitialized**
- std::list< std::string > **mNeededLibs**
- **s32 mLastError**

4.35.2 Member Typedef Documentation

4.35.2.1 typedef void(* loggerFunctionType)(const void *, const char *) [protected]

Function pointer type for the argument of

4.35.2.2 typedef const char*(* pcmGetErrorMsgType)(const int) [protected]

Function pointer type for

4.35.2.3 typedef const char**(* pcmGetNeededLibsType)() [protected]

Function pointer type for

4.35.2.4 typedef int(* pcmOnLoadType)() [protected]

Function pointer type for

4.35.2.5 typedef int(* pcmOnUnloadType)() [protected]

Function pointer type for

4.35.2.6 typedef int(* pcmSetLoggerFunctionType)(loggerFunctionType) [protected]

Function pointer type for

4.35.2.7 typedef int(* pcmSetPluginHandleType)(void *) [protected]

Function pointer type for

4.35.2.8 typedef ParCompMark::Pointer< Plugin, Mutex > Pointer

Type for pointer on this class.

Reimplemented from **DynLoad** (p. 68).

Reimplemented in **RendererPlugin** (p. 205).

4.35.3 Member Enumeration Documentation

4.35.3.1 enum PluginType

Definitions of plugin types.

Enumerator:

BASIC_PLUGIN Type for **Plugin**(p. 167) class

RENDERER_PLUGIN Type for **Renderer**(p. 198) class

4.35.4 Constructor & Destructor Documentation

4.35.4.1 Plugin (const PluginType & *type*, const std::string & *name*, const std::string & *filename*)

Create a plugin.

Parameters:

← ***type*** Type of the plugin.

← ***name*** **Name**(p. 134) of the plugin.

← ***filename*** **Plugin**(p. 167) file name.

4.35.4.2 ~Plugin () [virtual]

The destructor. This class has virtual destructor.

4.35.5 Member Function Documentation

4.35.5.1 void _checkError () [inline, protected, virtual]

Check the error code of the last operation.

4.35.5.2 virtual void _finalizeSpecific () [protected, pure virtual]

Specific finalization code.

Implemented in **RendererPlugin** (p. 205).

4.35.5.3 const char * _getErrorMsg () [protected, virtual]

Call pcmGetErrorMsg function of the plugin.

Returns:
Description of the last error.

4.35.5.4 void _getNeededLibs () [protected, virtual]

Get needed libraries from the plugin.

4.35.5.5 virtual void _initializeSpecific () [protected, pure virtual]

Specific initialization code.

Implemented in **RendererPlugin** (p. 206).

4.35.5.6 void _loggerFunction (const void * *handle*, const char * *message*) [static, protected]

Logger(p. 126) function with simple C style interface can be called by the plugin.

Parameters:
← ***handle*** **Plugin**(p. 167) handle
← ***message*** Message to log.

4.35.5.7 void _onLoad () [protected, virtual]

Call pcmOnLoad event handler of the plugin.

4.35.5.8 void _onUnload () [protected, virtual]

Call pcmOnUnload event handler of the plugin.

4.35.5.9 void _setLoggerFunction () [protected, virtual]

Set logger function to the plugin.

4.35.5.10 void _setPluginHandle () [protected, virtual]

Set plugin handle to the plugin.

4.35.5.11 void finalize () [virtual]

Finalize the plugin.

4.35.5.12 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:
The value of mInitialized

4.35.5.13 const s32 & getLastError () const [inline]

Getter of mLastError. Returns value of mLastError.

Returns:
The value of mLastError

4.35.5.14 const std::list< std::string > & getNeededLibs () const [inline]

Getter of mNeededLibs. Returns value of mNeededLibs.

Returns:
The value of mNeededLibs

4.35.5.15 const Plugin::PluginType & getPluginType () const [inline]

Getter of mPluginType. Returns value of mPluginType.

Returns:
The value of mPluginType

4.35.5.16 void initialize () [virtual]

Initialize the plugin.

4.35.6 Member Data Documentation

4.35.6.1 bool mInitialized [protected]

The plugin is initialized.

Remarks:
This is own attribute of this class.

4.35.6.2 s32 mLastError [protected]

Error code of the last error occured by the plugin code. 0 must mean no error.

Remarks:
This is own attribute of this class.

4.35.6.3 std::list< std::string > mNeededLibs [protected]

Needed dynamic libraries.

Remarks:
This is own attribute of this class.

4.35.6.4 PluginType mPluginType [protected]

Plugin(p. 167) type.

Remarks:
This is own attribute of this class.

4.36 PluginManager Class Reference

Inherits **Singleton**< **PluginManager** >.

4.36.1 Detailed Description

Singleton(p. 209) plugin loader and handler class.

Methods

- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual void **loadPlugins** ()
- virtual void **unloadPlugins** ()
- virtual **Plugin::Pointer getPlugin** (const std::string &name)
- virtual **Plugin::Pointer _loadPlugin** (const **Plugin::PluginType** &type, const std::string &filename)
- virtual void **_checkNeededLibs** (**Plugin::Pointer** &plugin)

Public Member Functions

Constructors & destructor

- **PluginManager** (const std::string &pluginDirectory="plugins/", const std::string &iniFile="plugins.ini")
- virtual ~**PluginManager** ()

Getters & setters

- const std::string & **getPluginDirectory** () const
- const std::string & **getPluginIniFile** () const
- const bool & **getInitialized** () const

Protected Attributes

Variables

- **Container**< **Plugin**, **Mutex** >::**Pointer mPlugins**
- std::string **mPluginDirectory**
- std::string **mPluginIniFile**
- bool **mInitialized**

4.36.2 Constructor & Destructor Documentation

4.36.2.1 PluginManager (const std::string & *pluginDirectory* = "plugins/", const std::string & *iniFile* = "plugins.ini")

Creates plugin manager.

Parameters:

← ***pluginDirectory*** **Plugin**(p. 167) directory path, relative to application data directory

← ***iniFile*** **Name**(p. 134) of plugin ini file.

4.36.2.2 ~PluginManager () [virtual]

The destructor. This class has virtual destructor.

4.36.3 Member Function Documentation

4.36.3.1 void _checkNeededLibs (Plugin::Pointer & *plugin*) [protected, virtual]

Check needed libs for the specified plugin. Throw exception when a lib needed by the plugin is not linked to parcompmark.

Parameters:

→ ***plugin*** Type of the plugin.

4.36.3.2 Plugin::Pointer _loadPlugin (const Plugin::PluginType & *type*, const std::string & *filename*) [protected, virtual]

Load a plugin for the given filename relative to mPluginDirectory.

Parameters:

← ***type*** **Pointer**(p. 177) to the plugin.

← ***filename*** Filename relative to mPluginDirectory.

Returns:

Pointer(p. 177) to the plugin.

4.36.3.3 void finalize () [virtual]

Finalize plugin manager.

4.36.3.4 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:

The value of mInitialized

4.36.3.5 Plugin::Pointer getPlugin (const std::string & *name*) [virtual]

Get plugin by name.

Parameters:

← ***name*** **Name**(p. 134) of the plugin.

Returns:

Pointer(p. 177) to the plugin.

4.36.3.6 const std::string & getPluginDirectory () const [inline]

Getter of mPluginDirectory. Returns value of mPluginDirectory.

Returns:
The value of mPluginDirectory

4.36.3.7 const std::string & getPluginIniFile () const [inline]

Getter of mPluginIniFile. Returns value of mPluginIniFile.

Returns:
The value of mPluginIniFile

4.36.3.8 void initialize () [virtual]

Initialize plugin manager.

4.36.3.9 void loadPlugins () [virtual]

Load all plugins in described mPluginIniFile.

4.36.3.10 void unloadPlugins () [virtual]

Unload all loaded plugins.

4.36.4 Member Data Documentation

4.36.4.1 bool mInitialized [protected]

The plugin manager is initialized.

Remarks:
This is own attribute of this class.

4.36.4.2 std::string mPluginDirectory [protected]

Plugin(p. 167) directory path, relative to application data directory (

Remarks:
This is own attribute of this class.

4.36.4.3 std::string mPluginIniFile [protected]

Name(p. 134) of plugin ini file placed in mPluginDirectory.

Remarks:
This is own attribute of this class.

4.36.4.4 Container< Plugin, Mutex >::Pointer mPlugins [protected]

Container(p. 47) of plugins.

Remarks:
This is own attribute of this class.

4.37 Pointer Class Template Reference

4.37.1 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**

4.37.2 Constructor & Destructor Documentation

4.37.2.1 Pointer () [inline]

Create a NULL pointer.

4.37.2.2 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.

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

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

Parameters:

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

4.37.2.4 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.

4.37.2.5 ∼Pointer () [virtual]

The destructor. This class has virtual destructor.

4.37.3 Member Function Documentation

4.37.3.1 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.

4.37.3.2 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.

4.37.3.3 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.

4.37.3.4 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.

4.37.3.5 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.

4.37.3.6 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.

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

Assign another smart pointer with keeping it locked.

Parameters:

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

4.37.3.8 bool getLocked () const [inline, virtual]

Return true, if the object is locked.

Returns:

True if the referenced object is locked.

4.37.3.9 T * getPtr () [inline, virtual]

Return a C style pointer to the referenced object.

Returns:

Pointer(p. 177) to the referenced object.

4.37.3.10 bool isNotNull () const [inline, virtual]

Return true if the pointer does reference something.

Returns:

True if the pointer does reference something.

4.37.3.11 bool isNull () const [inline, virtual]

Return true if the pointer does not reference anything.

Returns:

True if the pointer does not reference anything.

4.37.3.12 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.

4.37.3.13 void lock () [inline, virtual]

Lock(p. 124) the referenced object.

4.37.3.14 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.

4.37.3.15 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.

4.37.3.16 T * operator → () [inline, virtual]

Get referenced object.

Returns:
Referenced object.

4.37.3.17 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.

4.37.3.18 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.

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

Assign a C style pointer.

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

Returns:
Self reference.

4.37.3.20 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.

4.37.3.21 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.

4.37.3.22 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

4.37.3.23 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.

4.37.3.24 bool trylock () [inline, virtual]

Try locking referenced object.

Returns:
True if the locking was successful.

4.37.3.25 void unlock () [inline, virtual]

Unlock the referenced object.

4.37.4 Member Data Documentation

4.37.4.1 Meta* mMeta [protected]

Meta(p. 184) field for the referenced object.

Remarks:
This is own attribute of this class.

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

Initial value:

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

Null pointer constant.

Remarks:
This is own attribute of this class.

4.38 Pointer::Meta Struct Reference

4.38.1 Detailed Description

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

Meta(p. 184) field type for the referenced object.

Public Attributes

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

4.38.2 Member Data Documentation

4.38.2.1 bool dead

Indicates that the object is deleted

4.38.2.2 Lock lock

Lock(p. 124) for the object

4.38.2.3 bool ownMemory

Indicates that the referenced memory should deallocated by the smart pointer

4.38.2.4 T* ptr

Pointer(p. 177) to the referenced object

4.38.2.5 u32 usage

Usage counter

4.39 Process Class Reference

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

4.39.1 Detailed Description

Entity that composite or render something.

Methods

- virtual void **threadInitialize** ()
- virtual void **threadFinalize** ()
- virtual void **start** ()
- virtual **u32 stop** ()
- virtual void **initPC** ()
- virtual **Context** ∗ **getContext** ()
- 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 **setViewportForRendering** ()
- virtual void **displayFrameletIcon** ()
- virtual void **initProcess** ()
- virtual void **runningProcess** ()
- virtual void **setProcessTypeSq** (const int processType)
- virtual int **getProcessTypeSq** ()
- virtual void **setBufferByName** (const char ∗name)
- virtual void **setFinito** (const **u32** &frameID)
- virtual void **task** ()

Public Types

- typedef **Pointer**< **Process**, **Mutex** > **Pointer**
- enum **ProcessType** { **COMPOSITE** = 0, **RENDER** = 1 }

Public Member Functions

Constructors & destructor

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

Getters & setters

- **Process::ProcessType** & **getProcessType** ()
- void **setProcessType** (const **Process::ProcessType** &processtype)

- **Node** ∗ **getParent** () const
- **GLXRenderWindow::Pointer** & **getRenderWindow** ()
- **OpenGLRenderingEngine::Pointer** & **getRenderingEngine** ()
- const std::string & **getConfig** () const
- void **setConfig** (const std::string &config)
- char ∗ **getInitProcCode** ()
- void **setInitProcCode** (const char ∗initproccode)
- char ∗ **getRunningProcCode** ()
- void **setRunningProcCode** (const char ∗runningproccode)
- **Buffer::Pointer** & **getBuffer** ()
- const bool & **getInitialized** () const
- const PCid & **getFrameID** () const
- const PCid & **getStopID** () const
- const **u32** & **getFrameNumber** () const
- const **Real** & **getFrameTime** () const
- const **Real** & **getStartTime** () const
- void **setStartTime** (const **Real** &starttime)
- const bool & **getStop** () const
- void **setStop** (const bool &stop)
- const bool & **getStopAble** () const
- void **setStopAble** (const bool &stopable)
- **OutputNode::Pointer** & **getOutputDocument** ()
- const bool & **getShowFrameletIcon** () const

Static Public Member Functions

Scripting binding

- static void **squirrelGlue** ()

Static Public Attributes

Class constants

- static const std::string **PROCESSINITNUT** = "scripts/framework/process-init.nut"

Protected Attributes

Variables

- **ProcessType mProcessType**
- **Node** ∗ **mParent**
- **GLXRenderWindow::Pointer mRenderWindow**
- **OpenGLRenderingEngine::Pointer mRenderingEngine**
- std::string **mConfig**
- **Context::Pointer mContext**
- **SqVM::Pointer mSqVM**
- char ∗ **mInitProcCode**
- char ∗ **mRunningProcCode**
- **Buffer::Pointer mBuffer**
- bool **mInitialized**
- PCid **mFrameID**
- PCid **mStopID**
- **u32 mFrameNumber**
- **Real mFrameTime**
- **Real mStartTime**
- bool **mStop**
- bool **mStopAble**
- **OutputNode::Pointer mOutputDocument**
- bool **mShowFrameletIcon**

4.39.2 Member Typedef Documentation

4.39.2.1 typedef Pointer< Process, Mutex > Pointer

Type for pointer on this class.

4.39.3 Member Enumeration Documentation

4.39.3.1 enum ProcessType

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

Enumerator:

COMPOSITE **Process**(p. 185) composite and provides frame and requires framelets.

RENDER **Process**(p. 185) provides framelets.

4.39.4 Constructor & Destructor Documentation

4.39.4.1 Process ()

Default constructor for Squirrel compatibility. Calling this constructor always raises an exception.

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

Creates a process with a specified name.

Parameters:

← ***name*** **Name**(p. 134) of the host.

← ***parent*** Parent node

4.39.4.3 ~Process () [virtual]

The destructor. This class has virtual destructor.

4.39.5 Member Function Documentation

4.39.5.1 void actualizeRenderWindow () [virtual]

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

4.39.5.2 void displayFrameletIcon () [virtual]

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

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

Getter of mBuffer. Returns value of mBuffer.

Returns:
The value of mBuffer

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

Getter of mConfig. Returns value of mConfig.

Returns:
The value of mConfig

4.39.5.5 Context * getContext () [virtual]

Get PX context of this process.

Returns:
Pointer(p. 177) to the context.

4.39.5.6 const PCid & getFrameID () const [inline]

Getter of mFrameID. Returns value of mFrameID.

Returns:
The value of mFrameID

4.39.5.7 const u32 & getFrameNumber () const [inline]

Getter of mFrameNumber. Returns value of mFrameNumber.

Returns:
The value of mFrameNumber

4.39.5.8 const Real & getFrameTime () const [inline]

Getter of mFrameTime. Returns value of mFrameTime.

Returns:
The value of mFrameTime

4.39.5.9 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:
The value of mInitialized

4.39.5.10 char * getInitProcCode () [inline]

Getter of mInitProcCode. Returns value of mInitProcCode.

Returns:
The value of mInitProcCode

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

Getter of mOutputDocument. Returns value of mOutputDocument.

Returns:
The value of mOutputDocument

4.39.5.12 Node * getParent () const [inline]

Getter of mParent. Returns value of mParent.

Returns:
The value of mParent

4.39.5.13 Process::ProcessType & getProcessType () [inline]

Getter of mProcessType. Returns value of mProcessType.

Returns:
The value of mProcessType

4.39.5.14 int getProcessTypeSq () [inline, virtual]

Alternate squirrel setter of mProcessType.

Returns:
Process(p. 185) type int value (0 means COMPOSITE, 1 means RENDER).

4.39.5.15 OpenGLRenderingEngine::Pointer & getRenderingEngine () [inline]

Getter of mRenderingEngine. Returns value of mRenderingEngine.

Returns:
The value of mRenderingEngine

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

Getter of mRenderWindow. Returns value of mRenderWindow.

Returns:
The value of mRenderWindow

4.39.5.17 char * getRunningProcCode () [inline]

Getter of mRunningProcCode. Returns value of mRunningProcCode.

Returns:
The value of mRunningProcCode

4.39.5.18 const bool & getShowFrameletIcon () const [inline]

Getter of mShowFrameletIcon. Returns value of mShowFrameletIcon.

Returns:
The value of mShowFrameletIcon

4.39.5.19 const Real & getStartTime () const [inline]

Getter of mStartTime. Returns value of mStartTime.

Returns:
The value of mStartTime

4.39.5.20 const bool & getStop () const [inline]

Getter of mStop. Returns value of mStop.

Returns:
The value of mStop

4.39.5.21 const bool & getStopAble () const [inline]

Getter of mStopAble. Returns value of mStopAble.

Returns:
The value of mStopAble

4.39.5.22 const PCid & getStopID () const [inline]

Getter of mStopID. Returns value of mStopID.

Returns:
The value of mStopID

4.39.5.23 void initPC () [virtual]

Intialize PC functionality.

4.39.5.24 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.

4.39.5.25 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!)

4.39.5.26 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.

4.39.5.27 void setBufferByName (const char * *name*) [virtual]

Set mBuffer by name. mBuffer will be retrieved from the buffer container of the parent node by the name of the buffer.

Parameters:

← ***name*** **Name**(p. 134) of the buffer (C string for squirrel compatibility).

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

Setter of mConfig. Sets value of mConfig.

Parameters:

← ***config*** The value of mConfig

4.39.5.29 void setFinito (const u32 & *frameID*) [virtual]

Set parameters to finite pc.

Parameters:
← ***frameID*** StopID.

4.39.5.30 void setInitProcCode (const char * *initproccode*) [inline]

Setter of mInitProcCode. Sets value of mInitProcCode.

Parameters:
← ***initproccode*** The value of mInitProcCode

4.39.5.31 void setProcessType (const Process::ProcessType & *processtype*) [inline]

Setter of mProcessType. Sets value of mProcessType.

Parameters:
← ***processtype*** The value of mProcessType

4.39.5.32 void setProcessTypeSq (const int *processType*) [inline, virtual]

Alternate squirrel setter of mProcessType.

Parameters:
← ***processType*** **Process**(p. 185) type int value (0 means COMPOSITE, 1 means RENDER).

4.39.5.33 void setRunningProcCode (const char * *runningproccode*) [inline]

Setter of mRunningProcCode. Sets value of mRunningProcCode.

Parameters:
← ***runningproccode*** The value of mRunningProcCode

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

Setter of mStartTime. Sets value of mStartTime.

Parameters:
← ***starttime*** The value of mStartTime

4.39.5.35 void setStop (const bool & *stop*) [inline]

Setter of mStop. Sets value of mStop.

Parameters:
← ***stop*** The value of mStop

4.39.5.36 void setStopAble (const bool & *stopable*) [inline]

Setter of mStopAble. Sets value of mStopAble.

Parameters:
← ***stopable*** The value of mStopAble

4.39.5.37 void setViewportForRendering () [virtual]

Setup viewport for the rendering process based on the context properties.

4.39.5.38 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.

4.39.5.39 void start () [virtual]

Start the process.

4.39.5.40 u32 stop () [virtual]

Stop the process.

Returns:
frameID

4.39.5.41 void task () [protected, virtual]

What we do during rendering/compositing.

Reimplemented from **Thread** (p. 230).

4.39.5.42 void threadFinalize () [virtual]

Some finitiate step as thread.

Reimplemented from **Thread** (p. 230).

4.39.5.43 void threadInitialize () [virtual]

Some init step as thread.

Reimplemented from **Thread** (p. 231).

4.39.6 Member Data Documentation

4.39.6.1 Buffer::Pointer mBuffer [protected]

Graphics memory buffer on which the process operates. The buffer belongs the to parent node.

Remarks:
This is own attribute of this class.

4.39.6.2 std::string mConfig [protected]

Config string.

Remarks:
This is own attribute of this class.

4.39.6.3 Context::Pointer mContext [protected]

Context(p. 50) description.

Remarks:
This is own attribute of this class.

4.39.6.4 PCid mFrameID [protected]

The current frame id. It sets by PC.

Remarks:
This is own attribute of this class.

4.39.6.5 u32 mFrameNumber [protected]

Number of current frame.

Remarks:
This is own attribute of this class.

4.39.6.6 Real mFrameTime [protected]

Time elapsed between the start of the benchmark and the start of the current frame.

Remarks:
This is own attribute of this class.

4.39.6.7 bool mInitialized [protected]

The process is initialized.

Remarks:
This is own attribute of this class.

4.39.6.8 char* mInitProcCode [protected]

Squirrel initialization code (C string for squirrel compatibility).

Remarks:
This is own attribute of this class.

4.39.6.9 OutputNode::Pointer mOutputDocument [protected]

Process(p. 185) output document.

Remarks:
This is own attribute of this class.

4.39.6.10 Node* mParent [protected]

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

Remarks:
This attribute references an attribute.

4.39.6.11 ProcessType mProcessType [protected]

Composite or render process.

Remarks:
This is own attribute of this class.

4.39.6.12 OpenGLRenderingEngine::Pointer mRenderingEngine [protected]

Rendering engine of this process.

Remarks:
This is own attribute of this class.

4.39.6.13 GLXRenderWindow::Pointer mRenderWindow [protected]

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

Remarks:
This is own attribute of this class.

4.39.6.14 char* mRunningProcCode [protected]

Squirrel running code (C string for squirrel compatibility).

Remarks:
This is own attribute of this class.

4.39.6.15 bool mShowFrameletIcon [protected]

Show little framelet icon on rendering windows.

Remarks:
This is own attribute of this class.

4.39.6.16 SqVM::Pointer mSqVM [protected]

Squirrel virtual machine.

Remarks:
This is own attribute of this class.

4.39.6.17 Real mStartTime [protected]

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

Remarks:
This is own attribute of this class.

4.39.6.18 bool mStop [protected]

Indicates that process must be stopped.

Remarks:
This is own attribute of this class.

4.39.6.19 bool mStopAble [protected]

Indicates that process can be stopped during compositing.

Remarks:
This is own attribute of this class.

4.39.6.20 PCid mStopID [protected]

Stop id where composite is ended.

Remarks:
This is own attribute of this class.

4.39.6.21 const std::string PROCESSINITNUT = "scripts/framework/process-init.nut" [static]

Process(p. 185) initializer Squirrel script file (relative to the data directory).

Remarks:
This is own attribute of this class.

4.40 Renderer Class Reference

4.40.1 Detailed Description

Custom renderer object belongs to a renderer plugin.

Public Types

- typedef **ParCompMark::Pointer**< **Renderer**, **Mutex** > **Pointer**

Public Member Functions

Constructors & destructor

- **Renderer** ()
- **Renderer** (**RendererPlugin** *rendererPlugin, void *rendererHandle, **OpenGLRenderingEngine** *parent)
- virtual ~**Renderer** ()

Getters & setters

- void * **getRendererHandle** () const
- **OpenGLRenderingEngine** * **getParent** () const
- const bool & **getInitialized** () const

Methods

- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual void **resize** (const **u32** width, const **u32** height)
- virtual void **render** ()
- virtual void **setMiscParam** (const char *name, const char *value)
- virtual void **setObjectSpaceBoundingBox** (const double x0, const double y0, const double z0, const double x1, const double y1, const double z1)
- virtual void **setObjectId** (const unsigned objectId)
- virtual void **setScreenSpaceFramelet** (const double u0, const double v0, const double u1, const double v1)

Static Public Member Functions

Scripting binding

- static void **squirrelGlue** ()

Protected Attributes

Variables

- void * **mRendererHandle**
- **RendererPlugin** * **mRendererPlugin**
- **OpenGLRenderingEngine** * **mParent**
- bool **mInitialized**

4.40.2 Member Typedef Documentation

4.40.2.1 typedef ParCompMark::Pointer< Renderer, Mutex > Pointer

Type for pointer on this class.

4.40.3 Constructor & Destructor Documentation

4.40.3.1 Renderer ()

Default constructor for Squirrel compatibility. Calling this constructor always raises an exception.

4.40.3.2 Renderer (RendererPlugin * *rendererPlugin*, void * *rendererHandle*, OpenGLRenderingEngine * *parent*)

Create a custom renderer.

Parameters:

← ***rendererPlugin*** Corresponding renderer plugin.

← ***rendererHandle*** **Pointer**(p. 177) to the renderer instance in the plugin code.

← ***parent*** Parent rendering engine of the renderer.

4.40.3.3 ~Renderer () [virtual]

The destructor. This class has virtual destructor.

4.40.4 Member Function Documentation

4.40.4.1 void finalize () [virtual]

Finalize the renderer.

4.40.4.2 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:

The value of mInitialized

4.40.4.3 OpenGLRenderingEngine * getParent () const [inline]

Getter of mParent. Returns value of mParent.

Returns:

The value of mParent

4.40.4.4 void * getRendererHandle () const [inline]

Getter of mRendererHandle. Returns value of mRendererHandle.

Returns:
The value of mRendererHandle

4.40.4.5 void initialize () [virtual]

Initialize the renderer.

4.40.4.6 void render () [inline, virtual]

Call the `pcmOnRender` event handler of the plugin.

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

Call the `pcmOnResize` event handler of the plugin.

Parameters:
← ***width*** New width of the window on which the renderer renders.
← ***height*** New height of the window on which the renderer renders.

4.40.4.8 void setMiscParam (const char * *name*, const char * *value*) [inline, virtual]

Call the `pcmSetMiscParam` function of the plugin.

Parameters:
← ***name*** **Name**(p. 134) of the parameter.
← ***value*** Value of the parameter.

4.40.4.9 void setObjectId (const unsigned *objectId*) [inline, virtual]

Call the `pcmSetObjectId` function of the plugin.

Parameters:
← ***objectId*** Render object with the specified id.

4.40.4.10 void setObjectSpaceBoundingBox (const double *x0*, const double *y0*, const double *z0*, const double *x1*, const double *y1*, const double *z1*) [inline, virtual]

Call the `pcmSetObjectSpaceBoundingBox` function of the plugin.

Parameters:
← ***x0*** Lowest corner of the bounding box.
← ***y0*** Lowest corner of the bounding box.

← ***z0*** Lowest corner of the bounding box.

← ***x1*** Highest corner of the bounding box.

← ***y1*** Highest corner of the bounding box.

← ***z1*** Highest corner of the bounding box.

4.40.4.11 void setScreenSpaceFramelet (const double *u0*, const double *v0*, const double *u1*, const double *v1*) [inline, virtual]

Call the `pcmSetScreenSpaceFramelet` function of the plugin.

Parameters:

← ***u0*** Top-left corner of the framelet.

← ***v0*** Top-left corner of the framelet.

← ***u1*** Bottom-right corner of the framelet.

← ***v1*** Bottom-right corner of the framelet.

4.40.4.12 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.

4.40.5 Member Data Documentation

4.40.5.1 bool mInitialized [protected]

The renderer is initialized.

Remarks:

This is own attribute of this class.

4.40.5.2 OpenGLRenderingEngine* mParent [protected]

Parent rendering engine of the renderer.

Remarks:

This attribute references an attribute.

4.40.5.3 void* mRendererHandle [protected]

Pointer(p. 177) to the renderer instance in the plugin code.

Remarks:

This attribute references an attribute.

4.40.5.4 RendererPlugin* mRendererPlugin [protected]

Corresponding renderer plugin.

Remarks:
This attribute references an attribute.

4.41 RendererPlugin Class Reference

Inherits **Plugin**.

4.41.1 Detailed Description

Custom renderer plugin.

Methods

- virtual **Renderer** ∗ **createRenderer** (**GLXRenderWindow::Pointer** window, **OpenGLRenderingEngine** ∗parent)
- virtual void **destroyRenderer** (**Renderer** ∗renderer)
- virtual void **resize** (void ∗rendererHandle, const **u32** &width, const **u32** &height)
- virtual void **render** (void ∗rendererHandle, const double &time, const unsigned &frame)
- virtual void **setMiscParam** (void ∗rendererHandle, const char ∗&name, const char ∗&value)
- virtual void **setObjectSpaceBoundingBox** (void ∗rendererHandle, const double &x0, const double &y0, const double &z0, const double &x1, const double &y1, const double &z1)
- virtual void **setObjectId** (void ∗rendererHandle, const unsigned &objectId)
- virtual void **setScreenSpaceFramelet** (void ∗rendererHandle, const double &u0, const double &v0, const double &u1, const double &v1)
- virtual void **_initializeSpecific** ()
- virtual void **_finalizeSpecific** ()
- virtual void **_getNeededOpenGLExts** ()
- virtual void **_initFastFunctions** ()

Public Types

- typedef **ParCompMark::Pointer**< **RendererPlugin**, **Mutex** > **Pointer**
- typedef int(∗ **pcmSetMiscParamType**)(void ∗, const char ∗, const char ∗)
- typedef int(∗ **pcmSetObjectSpaceBoundingBoxType**)(void ∗, double, double, double, double, double, double)
- typedef int(∗ **pcmSetObjectIdType**)(void ∗, unsigned)
- typedef int(∗ **pcmSetScreenSpaceFrameletType**)(void ∗, double, double, double, double)
- typedef int(∗ **pcmOnResizeType**)(void ∗, unsigned, unsigned)
- typedef int(∗ **pcmOnRenderType**)(void ∗, double, unsigned)

Public Member Functions

Constructors & destructor

- **RendererPlugin** (const **Plugin::PluginType** &type, const std::string &name, const std::string &filename)
- virtual ∼**RendererPlugin** ()

Getters & setters

- const std::list< std::string > & **getNeededOpenGLExts** () const

Protected Types

- typedef const char **(* **pcmGetNeededOpenGLExtsType**)()
- typedef void *(* **pcmOnCreateRendererType**)(Display *, Window, XVisualInfo *, GLXContext)
- typedef int(* **pcmOnDestroyRendererType**)(void *)

Protected Attributes

Variables

- std::list< std::string > **mNeededOpenGLExts**
- **RendererPlugin::pcmOnResizeType mPcmOnResize**
- **RendererPlugin::pcmOnRenderType mPcmOnRender**

4.41.2 Member Typedef Documentation

4.41.2.1 typedef const char**(* pcmGetNeededOpenGLExtsType)() [protected]

Function pointer type for

4.41.2.2 typedef void*(* pcmOnCreateRendererType)(Display *, Window, XVisualInfo *, GLXContext) [protected]

Function pointer type for

4.41.2.3 typedef int(* pcmOnDestroyRendererType)(void *) [protected]

Function pointer type for

4.41.2.4 typedef int(* pcmOnRenderType)(void *, double, unsigned)

Function pointer type for

4.41.2.5 typedef int(* pcmOnResizeType)(void *, unsigned, unsigned)

Function pointer type for

4.41.2.6 typedef int(* pcmSetMiscParamType)(void *, const char *, const char *)

Function pointer type for

4.41.2.7 typedef int(* pcmSetObjectIdType)(void *, unsigned)

Function pointer type for

4.41.2.8 typedef int(∗ pcmSetObjectSpaceBoundingBoxType)(void ∗, double, double, double, double, double, double)

Function pointer type for

4.41.2.9 typedef int(∗ pcmSetScreenSpaceFrameletType)(void ∗, double, double, double, double)

Function pointer type for

4.41.2.10 typedef ParCompMark::Pointer< RendererPlugin, Mutex > Pointer

Type for pointer on this class.

Reimplemented from **Plugin** (p. 169).

4.41.3 Constructor & Destructor Documentation

4.41.3.1 RendererPlugin (const Plugin::PluginType & *type*, const std::string & *name*, const std::string & *filename*)

Create a renderer plugin.

Parameters:

← ***type*** Type of the plugin.

← ***name*** **Name**(p. 134) of the renderer plugin.

← ***filename*** **Plugin**(p. 167) file name.

4.41.3.2 ∼RendererPlugin () [virtual]

The destructor. This class has virtual destructor.

4.41.4 Member Function Documentation

4.41.4.1 void _finalizeSpecific () [protected, virtual]

Specific finalization code.

Implements **Plugin** (p. 169).

4.41.4.2 void _getNeededOpenGLExts () [protected, virtual]

Get needed OpenGL externsions from the plugin.

4.41.4.3 void _initFastFunctions () [protected, virtual]

Initialize mPcmOnResize and mPcmOnRender function pointers. These function calls should be fast.

4.41.4.4 void _initializeSpecific () [protected, virtual]

Specific initialization code.

Implements **Plugin** (p. 170).

4.41.4.5 Renderer * createRenderer (GLXRenderWindow::Pointer *window*, OpenGLRenderingEngine * *parent*) [virtual]

Create renderer object. Also call OpenGL intialization code for the plugin.

Parameters:

← ***window*** The window on which the renderer will render.

← ***parent*** Parent rendering engine of the renderer.

Returns:

Created renderer.

4.41.4.6 void destroyRenderer (Renderer * *renderer*) [virtual]

Destroy the specified renderer.

Parameters:

← ***renderer*** **Renderer**(p. 198) to destroy.

4.41.4.7 const std::list< std::string > & getNeededOpenGLExts () const [inline]

Getter of mNeededOpenGLExts. Returns value of mNeededOpenGLExts.

Returns:

The value of mNeededOpenGLExts

4.41.4.8 void render (void * *rendererHandle*, const double & *time*, const unsigned & *frame*) [inline, virtual]

Call the `pcmOnRender` event handler of the plugin.

Parameters:

← ***rendererHandle*** **Pointer**(p. 177) to the renderer instance in the plugin code.

← ***time*** Time since start of the rendering.

← ***frame*** Frame number.

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

Call the `pcmOnResize` event handler of the plugin.

Parameters:

← ***rendererHandle*** **Pointer**(p. 177) to the renderer instance in the plugin code.

← ***width*** New width of the window on which the renderer renders.

← ***height*** New height of the window on which the renderer renders.

4.41.4.10 void setMiscParam (void * *rendererHandle*, const char *& *name*, const char *& *value*) [inline, virtual]

Call the `pcmSetMiscParam` function of the plugin.

Parameters:

← ***rendererHandle*** **Pointer**(p. 177) to the renderer instance in the plugin code.

← ***name*** **Name**(p. 134) of the parameter.

← ***value*** Value of the parameter.

4.41.4.11 void setObjectId (void * *rendererHandle*, const unsigned & *objectId*) [inline, virtual]

Call the `pcmSetObjectId` function of the plugin.

Parameters:

← ***rendererHandle*** **Pointer**(p. 177) to the renderer instance in the plugin code.

← ***objectId*** Render object with the specified id.

4.41.4.12 void setObjectSpaceBoundingBox (void * *rendererHandle*, const double & *x0*, const double & *y0*, const double & *z0*, const double & *x1*, const double & *y1*, const double & *z1*) [inline, virtual]

Call the `pcmSetObjectSpaceBoundingBox` function of the plugin.

Parameters:

← ***rendererHandle*** **Pointer**(p. 177) to the renderer instance in the plugin code.

← ***x0*** Lowest corner of the bounding box.

← ***y0*** Lowest corner of the bounding box.

← ***z0*** Lowest corner of the bounding box.

← ***x1*** Highest corner of the bounding box.

← ***y1*** Highest corner of the bounding box.

← ***z1*** Highest corner of the bounding box.

4.41.4.13 void setScreenSpaceFramelet (void * *rendererHandle*, const double & *u0*, const double & *v0*, const double & *u1*, const double & *v1*) [inline, virtual]

Call the `pcmSetScreenSpaceFramelet` function of the plugin.

Parameters:

← ***rendererHandle*** **Pointer**(p. 177) to the renderer instance in the plugin code.

← ***u0*** Top-left corner of the framelet.

← ***v0*** Top-left corner of the framelet.

← ***u1*** Bottom-right corner of the framelet.

← ***v1*** Bottom-right corner of the framelet.

4.41.5 Member Data Documentation

4.41.5.1 std::list< std::string > mNeededOpenGLExts [protected]

Needed OpenGL extensions.

Remarks:
This is own attribute of this class.

4.41.5.2 RendererPlugin::pcmOnRenderType mPcmOnRender [protected]

Function pointer for

Remarks:
This is own attribute of this class.

4.41.5.3 RendererPlugin::pcmOnResizeType mPcmOnResize [protected]

Function pointer for

Remarks:
This is own attribute of this class.

4.42 Singleton Class Template Reference

4.42.1 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**

4.42.2 Constructor & Destructor Documentation

4.42.2.1 Singleton ()

Default constructor.

4.42.2.2 ~Singleton () [virtual]

The destructor. This class has virtual destructor.

4.42.3 Member Function Documentation

4.42.3.1 void createInstance () [inline, static]

Create singleton instance.

4.42.3.2 void destroyInstance () [inline, static]

Destroy singleton instance.

4.42.3.3 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

Reimplemented in **Host** (p. 113).

4.42.4 Member Data Documentation

4.42.4.1 T * mInstance [static, protected]

Class level instance.

Remarks:
This is own attribute of this class.

4.43 SqVM Class Reference

Inherits **Name**.

4.43.1 Detailed Description

Squirrel virtual machine.

Getters & setters

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

Methods

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

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**
- **Container**< **SqVM::Script**, **Mutex** >::**Pointer mScripts**

Static Protected Attributes

Class variables

- static **SqVM::Pointer mCurrentVM**
- static **Mutex mCurrentVMLock**
- static char **mStringBuffer** [32768] = ""

Classes

- struct **Script**

4.43.2 Member Typedef Documentation

4.43.2.1 typedef Pointer< SqVM, Mutex > Pointer

Type for pointer on this class.

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

Struct for script attributes.

4.43.3 Member Enumeration Documentation

4.43.3.1 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)

4.43.4 Constructor & Destructor Documentation

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

Create Squirrel virtual machine.

Parameters:

← ***name*** **Name**(p. 134) of the virtual machine.

4.43.4.2 ~SqVM () [virtual]

The destructor. This class has virtual destructor.

4.43.5 Member Function Documentation

4.43.5.1 void activate () [protected, virtual]

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

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

Compile and execute the script.

Parameters:

→ ***script*** **Script**(p. 219) handle.

4.43.5.3 SqVM::Script::Pointer createScript (const std::string & *scriptName* = "", const bool & *dynamic* = false, const std::string & *mainMethod* = "main", const bool & *hasReturn* = false) [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. 134) of the script.

← ***dynamic*** Dynamic flag.

← ***mainMethod*** **Script**(p. 219) entry method name. Giving **SqVM::NOMAINMETHOD**(p. 218) as mainMethod indicates that no main method.

← ***hasReturn*** The script has a return value

Returns:
Pointer(p. 177) to the created script object.

4.43.5.4 void deactivate () [protected, virtual]

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

4.43.5.5 void finalize () [virtual]

Finalize virtual machine.

4.43.5.6 SqVM::Script::Pointer findOrAddScript (const std::string & *scriptName*, const bool & *dynamic* = false, const std::string & *mainMethod* = "main", const bool & *hasReturn* = false) [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. 134) of the searched script.

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

← ***mainMethod*** **Script**(p. 219) entry method name. Giving **SqVM::NOMAINMETHOD**(p. 218) as mainMethod indicates that no main method.

← ***hasReturn*** The script has a return value

Returns:
Pointer(p. 177) of the found or the newly created script object.

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

Finds a prevously added script.

Parameters:
← ***scriptName*** **Name**(p. 134) of a dynamic script name or filename.

Returns:
Pointer(p. 177) of the found script object.

4.43.5.8 const bool & getError () const [inline]

Getter of mError. Returns value of mError.

Returns:
The value of mError

4.43.5.9 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:
The value of mInitialized

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

Getter of mStringBufferSize. Returns value of mStringBufferSize.

Returns:
The value of mStringBufferSize

4.43.5.11 void initialize () [protected, virtual]

Initialize virtual machine.

4.43.5.12 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

4.43.5.13 std::string runScriptByName (const std::string & *scriptName*, const std::list< std::string > & *parameters* = std::list< std::string >()) [virtual]

Execute a previously stored script.

Parameters:
← ***scriptName*** **Name**(p. 134) of a dynamic script name or filename.
← ***parameters*** List of passed string parameters

Returns:
The return value of the script.

4.43.5.14 std::string runScriptFromFile (const std::string & *filename*, const std::string & *mainMethod* = "main", const std::list< std::string > & *parameters* = std::list< std::string >(), const bool & *hasReturn* = false) [virtual]

Execute script loaded from file. Giving **SqVM::NOMAINMETHOD**(p. 218) as mainMethod indicates that no main method.

Parameters:

← ***filename*** **Script**(p. 219) filename

← ***mainMethod*** **Script**(p. 219) entry method name. Giving **SqVM::NOMAINMETHOD**(p. 218) as mainMethod indicates that no main method.

← ***parameters*** List of passed string parameters

← ***hasReturn*** The script has a return value

Returns:

The return value of the script.

4.43.5.15 std::string runScriptFromString (const std::string & *scriptString*, const std::string & *scriptName* = "", const std::string & *mainMethod* = "main", const std::list< std::string > & *parameters* = std::list< std::string >(), const bool & *hasReturn* = false) [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. 218) as mainMethod indicates that no main method.

Parameters:

← ***scriptString*** **Script**(p. 219) string

← ***scriptName*** **Name**(p. 134) of the script. If it is not empty the VM store this script into the class level store for better performance.

← ***mainMethod*** **Script**(p. 219) entry method name. Giving **SqVM::NOMAINMETHOD**(p. 218) as mainMethod indicates that no main method.

← ***parameters*** List of passed string parameters

← ***hasReturn*** The script has a return value

Returns:

The return value of the script.

4.43.5.16 void setParameters (SqVM::Script::Pointer & *script*, const std::list< std::string > & *parameters*) [protected, virtual]

Set parameters of a script.

Parameters:

→ ***script*** **Script**(p. 219) handle.

← ***parameters*** List of passed string parameters

4.43.6 Member Data Documentation

4.43.6.1 SqVM::Pointer mCurrentVM [static, protected]

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

Remarks:

This is own attribute of this class.

4.43.6.2 Mutex mCurrentVMLock [static, protected]

External mutex for mCurrentVM.

Remarks:
This is own attribute of this class.

4.43.6.3 bool mError [protected]

An error occured on this virtual machine.

Remarks:
This is own attribute of this class.

4.43.6.4 bool mInitialized [protected]

The virtual machine is initialized.

Remarks:
This is own attribute of this class.

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

Script(p. 219) container. Dynamic script can be also stored here when they have a proper name.

Remarks:
This is own attribute of this class.

4.43.6.6 SquirrelVMSys* mSquirrelVMSys [protected]

Squirrel virtual machine.

Remarks:
This attribute references an attribute.

4.43.6.7 char mStringBuffer = "" [static, protected]

Common C style string buffer of printFunction.

Remarks:
This is own attribute of this class.

4.43.6.8 const u32 mStringBufferSize = 32768 [static, protected]

Size of mStringBuffer.

Remarks:
This is own attribute of this class.

4.43.6.9 SqVM::Pointer mThis [protected]

Smart pointer on this object.

Remarks:
This is own attribute of this class.

4.43.6.10 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.

4.44 SqVM::Script Struct Reference

4.44.1 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**
- bool **hasReturn**
- std::string **returnValue**
- const SQChar * **parameters** [10]
- **u32 parameterCount**

4.44.2 Member Typedef Documentation

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

Smart pointer on this struct

4.44.3 Member Data Documentation

4.44.3.1 bool compiled

The loaded script

4.44.3.2 bool dynamic

Indicate that the script is dynamic otherwise loaded from a file

4.44.3.3 bool hasReturn

The script has a return value

4.44.3.4 std::string mainMethod

Name(p. 134) of the entry method

4.44.3.5 std::string name

Name(p. 134) of the script (filename or dynamic script name)

4.44.3.6 u32 parameterCount

Number of parameters

4.44.3.7 const SQChar* parameters[10]

Script(p. 219) parameters

4.44.3.8 std::string returnValue

Return value of the last execution

4.44.3.9 SquirrelObject scriptObject

Squirrel script object

4.44.3.10 std::string scriptString

Script(p. 219) containing the source code. This is only set at dynamic scripts

4.45 StringConverter Class Reference

4.45.1 Detailed Description

Convert different types to std::string.

Class methods

- static std::string **toString** (const **u8** &value)
- static std::string **toString** (const **u16** &value)
- static std::string **toString** (const **u32** &value)
- static std::string **toString** (const **s8** &value)
- static std::string **toString** (const **s16** &value)
- static std::string **toString** (const **s32** &value)
- static std::string **toString** (const **Real** &value, const **s32** &fieldWidth=**StringConverter::DEFAULTFIELDWIDTH**, const **s32** &precision=**StringConverter::DEFAULTPRECISION**)
- static std::string **toString** (const bool value)
- static std::string **toString** (const void *value)
- static **u32 toU32** (const std::string value)
- static **Real toReal** (const std::string value)
- static void * **parsePointer** (const std::string value)
- static void **trim** (std::string &str)
- static std::vector< std::string > **tokenize** (const std::string &str, const char *delimiters)
- static std::string **_fitFieldWidth** (const std::string &str, const **s32** &fieldWidth)

Static Public Attributes

Class constants

- static const **s32 DEFAULTPRECISION** = -1
- static const **s32 DEFAULTFIELDWIDTH** = -1

4.45.2 Member Function Documentation

4.45.2.1 std::string _fitFieldWidth (const std::string & *str*, const s32 & *fieldWidth*) [static, protected]

Append empty spaces to fit to field width.

Parameters:

← ***str*** String to fill to fit to the field width.

← ***fieldWidth*** Field width.

Returns:

Converted string value

4.45.2.2 void * parsePointer (const std::string *value*) [static]

Convert string to void *.

Parameters:
← ***value*** Value to convert.

Returns:
Converted pointer

4.45.2.3 std::vector< std::string > tokenize (const std::string & *str*, const char * *delimiters*) [static]

Tokenize the specified string. Original source: http://www.digitalpeer.com/id/simple

Parameters:
← ***str*** String to trim.
← ***delimiters*** String to trim.

Returns:
Array of tokens.

4.45.2.4 Real toReal (const std::string *value*) [static]

Convert string to Real.

Parameters:
← ***value*** Value to convert.

Returns:
Converted Real value

4.45.2.5 std::string toString (const void * *value*) [static]

Convert void * to string.

Parameters:
← ***value*** Value to convert.

Returns:
Converted string value

4.45.2.6 std::string toString (const bool *value*) [static]

Convert bool to string.

Parameters:
← ***value*** Value to convert.

Returns:
Converted string value

4.45.2.7 std::string toString (const Real & *value*, const s32 & *fieldWidth* = StringConverter::DEFAULTFIELDWIDTH, const s32 & *precision* = StringConverter::DEFAULTPRECISION) [static]

Convert Real to string. If the precision is set, the real value is converted in fixed point notation scientific notation otherwise.

Parameters:

← ***value*** Value to convert.

← ***fieldWidth*** Field width.

← ***precision*** Number of fragment digits.

Returns:

Converted string value

4.45.2.8 std::string toString (const s32 & *value*) [static]

Convert s32 to string.

Parameters:

← ***value*** Value to convert.

Returns:

Converted string value

4.45.2.9 std::string toString (const s16 & *value*) [static]

Convert s16 to string.

Parameters:

← ***value*** Value to convert.

Returns:

Converted string value

4.45.2.10 std::string toString (const s8 & *value*) [static]

Convert s8 to string.

Parameters:

← ***value*** Value to convert.

Returns:

Converted string value

4.45.2.11 std::string toString (const u32 & *value*) [static]

Convert u32 to string.

Parameters:
← ***value*** Value to convert.

Returns:
Converted string value

4.45.2.12 std::string toString (const u16 & *value*) [static]

Convert u16 to string.

Parameters:
← ***value*** Value to convert.

Returns:
Converted string value

4.45.2.13 std::string toString (const u8 & *value*) [static]

Convert u8 to string.

Parameters:
← ***value*** Value to convert.

Returns:
Converted string value

4.45.2.14 u32 toU32 (const std::string *value*) [static]

Convert string to u32.

Parameters:
← ***value*** Value to convert.

Returns:
Converted u32 value

4.45.2.15 void trim (std::string & *str*) [static]

Leave whitespaces from the beginning and the end of the string.

Parameters:
→ ***str*** String to trim.

4.45.3 Member Data Documentation

4.45.3.1 const s32 DEFAULTFIELDWIDTH = -1 [static]

Constant for default field width.

Remarks:
This is own attribute of this class.

4.45.3.2 const s32 DEFAULTPRECISION = -1 [static]

Constant for default precision.

Remarks:
This is own attribute of this class.

4.46 Thread Class Reference

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

4.46.1 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 **threadInitialize** ()
- virtual void **threadFinalize** ()
- virtual void **startThread** ()
- virtual void **joinThread** ()
- virtual void **shutDownThread** ()
- virtual void **stopThread** ()
- virtual void **go** ()
- virtual void **thread** ()
- virtual bool **iteration** ()
- virtual void **task** ()
- virtual void **wait** ()

Class methods

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

Public Types

- typedef **ParCompMark::Pointer**< bool, **Mutex** > **BoolPointer**

Public Member Functions

Constructors & destructor

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

Getters & setters

- const std::string & **getThreadName** () const
- const **Thread::BoolPointer** & **getWait** () 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**
- **BoolPointer mWait**
- bool **mStopRequested**
- bool **mJoinable**
- bool **mWaitThread**
- pthread_t **mThread**
- pthread_cond_t **mCondition**
- pthread_mutex_t **mConditionMutex**
- bool **mRunning**
- **u32 mCurrentFPS**
- **u32 mExpectedFPS**
- **u32 mIterationNumber**

4.46.2 Member Typedef Documentation

4.46.2.1 typedef ParCompMark::Pointer< bool, Mutex > BoolPointer

Type for pointer to an int.

4.46.3 Constructor & Destructor Documentation

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

Thread(p. 226) constructor.

Parameters:
← ***name*** **Name**(p. 134) of the thread.

4.46.3.2 ~Thread () [virtual]

The destructor. This class has virtual destructor.

4.46.4 Member Function Documentation

4.46.4.1 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.

4.46.4.2 const u32 & getCurrentFPS () const [inline]

Getter of mCurrentFPS. Returns value of mCurrentFPS.

Returns:
The value of mCurrentFPS

4.46.4.3 const u32 & getExpectedFPS () const [inline]

Getter of mExpectedFPS. Returns value of mExpectedFPS.

Returns:
The value of mExpectedFPS

4.46.4.4 const u32 & getIterationNumber () const [inline]

Getter of mIterationNumber. Returns value of mIterationNumber.

Returns:
The value of mIterationNumber

4.46.4.5 const bool & getJoinable () const [inline]

Getter of mJoinable. Returns value of mJoinable.

Returns:
The value of mJoinable

4.46.4.6 const bool & getRunning () const [inline]

Getter of mRunning. Returns value of mRunning.

Returns:
The value of mRunning

4.46.4.7 const bool & getStopRequested () const [inline]

Getter of mStopRequested. Returns value of mStopRequested.

Returns:
The value of mStopRequested

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

Getter of mThreadName. Returns value of mThreadName.

Returns:
The value of mThreadName

4.46.4.9 Real getUSTime () [inline, static]

Get system time in us.

Returns:
System time in us.

4.46.4.10 const Thread::BoolPointer & getWait () const [inline]

Getter of mWait. Returns value of mWait.

Returns:
The value of mWait

4.46.4.11 const bool & getWaitThread () const [inline]

Getter of mWaitThread. Returns value of mWaitThread.

Returns:
The value of mWaitThread

4.46.4.12 void go () [virtual]

Go forward from wait.

4.46.4.13 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?

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

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

Returns:
If true, the thread is stopped.

4.46.4.15 void joinThread () [virtual]

Join thread. Wait for its ending.

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

4.46.4.16 void shutDownThread () [virtual]

Immediately stop the thread.

4.46.4.17 void startThread () [virtual]

Start thread.

4.46.4.18 void stopThread () [virtual]

Request thread shut down.

4.46.4.19 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. 110), **NetClient** (p. 137), **NetServer** (p. 140), and **Process** (p. 193).

4.46.4.20 void thread () [protected, virtual]

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

4.46.4.21 void threadFinalize () [virtual]

Some finitiate step as thread.

Reimplemented in **Process** (p. 193).

4.46.4.22 void threadInitialize () [virtual]

Some init step as thread.

Reimplemented in **Process** (p. 193).

4.46.4.23 void wait () [protected, virtual]

Wait for something.

4.46.4.24 void yield () [inline, static]

Yield current thread.

4.46.5 Member Data Documentation

4.46.5.1 pthread_cond_t mCondition [protected]

Condition.

Remarks:
This is own attribute of this class.

4.46.5.2 pthread_mutex_t mConditionMutex [protected]

Condition.

Remarks:
This is own attribute of this class.

4.46.5.3 u32 mCurrentFPS [protected]

The thread is running.

Remarks:
This is own attribute of this class.

4.46.5.4 u32 mExpectedFPS [protected]

The thread is running. Zero means no limit.

Remarks:
This is own attribute of this class.

4.46.5.5 u32 mIterationNumber [protected]

Iteration number of task. Zero means nan.

Remarks:
This is own attribute of this class.

4.46.5.6 bool mJoinable [protected]

The thread is created as joinable or not.

Remarks:
This is own attribute of this class.

4.46.5.7 bool mRunning [protected]

The thread is running.

Remarks:
This is own attribute of this class.

4.46.5.8 bool mStopRequested [protected]

The thread is requested to stop.

Remarks:
This is own attribute of this class.

4.46.5.9 pthread_t mThread [protected]

Thread(p. 226) handle.

Remarks:
This is own attribute of this class.

4.46.5.10 std::string mThreadName [protected]

Name(p. 134) of the thread.

Remarks:
This is own attribute of this class.

4.46.5.11 BoolPointer mWait [protected]

Indicates that thread waits or not.

Remarks:
This is own attribute of this class.

4.46.5.12 bool mWaitThread [protected]

When class destructor call, wait thread or not.

Remarks:
This is own attribute of this class.

4.47 Timer Class Reference

4.47.1 Detailed Description

Collection of time handling methods.

Static Public Member Functions

Class methods

- static **Real getSystemTime** ()
- static std::string **getTimeString** (const bool &microseconds=true, const bool &seconds=true, const bool &minutes=true, const bool &hours=true, const **Real** &correction=0.0)
- static std::string **getDateString** (const bool &days=true, const bool &months=true, const bool &years=true, const **Real** &correction=0.0)
- static std::string **getTimeDateString** (const bool &microseconds=true, const bool &seconds=true, const bool &minutes=true, const bool &hours=true, const bool &days=true, const bool &months=true, const bool &years=true, const **Real** &correction=0.0)
- static void **sleep** (const **Real** &time)

Static Public Attributes

Class constants

- static const **Real EPSILONDELAY** = 0.000001

4.47.2 Member Function Documentation

4.47.2.1 std::string getDateString (const bool & *days* = `true`, const bool & *months* = `true`, const bool & *years* = `true`, const Real & *correction* = `0.0`) `[static]`

Return current date as a string.

Parameters:

← ***days*** Flag indicates days to display in output

← ***months*** Flag indicates months to display in output

← ***years*** Flag indicates years to display in output

← ***correction*** Time correction in seconds

Returns:

Current time in string format

4.47.2.2 Real getSystemTime () `[inline, static]`

Return time in secondes.

Returns:

Time in seconds

4.47.2.3 std::string getTimeDateString (const bool & *microseconds* = true, const bool & *seconds* = true, const bool & *minutes* = true, const bool & *hours* = true, const bool & *days* = true, const bool & *months* = true, const bool & *years* = true, const Real & *correction* = 0.0) [static]

Return current time and date as a string.

Parameters:

← ***microseconds*** Flag indicates microseconds to display in output

← ***seconds*** Flag indicates seconds to display in output

← ***minutes*** Flag indicates minutes to display in output

← ***hours*** Flag indicates hours to display in output

← ***days*** Flag indicates days to display in output

← ***months*** Flag indicates months to display in output

← ***years*** Flag indicates years to display in output

← ***correction*** Time correction in seconds

Returns:

Current time in string format

4.47.2.4 std::string getTimeString (const bool & *microseconds* = true, const bool & *seconds* = true, const bool & *minutes* = true, const bool & *hours* = true, const Real & *correction* = 0.0) [static]

Return current time as a string.

Parameters:

← ***microseconds*** Flag indicates microseconds to display in output

← ***seconds*** Flag indicates seconds to display in output

← ***minutes*** Flag indicates minutes to display in output

← ***hours*** Flag indicates hours to display in output

← ***correction*** Time correction in seconds

Returns:

Current time in string format

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

Sleep process for the specified seconds.

Parameters:

← ***time*** Time to sleep

4.47.3 Member Data Documentation

4.47.3.1 const Real EPSILONDELAY = 0.000001 [static]

Constant for minimal time delay (1us).

Remarks:
This is own attribute of this class.

4.48 XDisplay Class Reference

4.48.1 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** ()
- static const **Mutex** & **getErrorHandlerMutex** ()
- static const bool & **getTolerateErrors** ()
- static void **setTolerateErrors** (const bool &tolerateerrors)

Methods

- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual void **synchronize** (const bool &discard=false)
- 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

Static Protected Member Functions

Class methods

- static int **errorHandler** (Display ∗display, XErrorEvent ∗errorEvent)

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
- static **Mutex mErrorHandlerMutex**
- static bool **mTolerateErrors** = false

Classes

- struct **VisualAttribs**

4.48.2 Member Typedef Documentation

4.48.2.1 typedef Pointer< XDisplay, Mutex > Pointer

Type for pointer on this class.

4.48.3 Constructor & Destructor Documentation

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

Create an X display.

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

4.48.3.2 ∼XDisplay () `[virtual]`

The destructor. This class has virtual destructor.

4.48.4 Member Function Documentation

4.48.4.1 int errorHandler (Display * *display*, XErrorEvent * *errorEvent*) [static, protected]

X error handler.

Parameters:

← ***display*** X display

← ***errorEvent*** Error event

Returns:

Return code of the handler.

4.48.4.2 void finalize () [virtual]

Finalize X display.

4.48.4.3 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

4.48.4.4 inline::Display * getDisplay () const

Getter of mDisplay. Returns value of mDisplay.

Returns:

The value of mDisplay

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

Getter of mDisplayName. Returns value of mDisplayName.

Returns:

The value of mDisplayName

4.48.4.6 const Mutex & getErrorHandlerMutex () [inline, static]

Getter of mErrorHandlerMutex. Returns value of mErrorHandlerMutex.

Returns:
The value of mErrorHandlerMutex

4.48.4.7 const u32 & getHeight () const [inline]

Getter of mHeight. Returns value of mHeight.

Returns:
The value of mHeight

4.48.4.8 const bool & getInitialized () const [inline]

Getter of mInitialized. Returns value of mInitialized.

Returns:
The value of mInitialized

4.48.4.9 const bool & getTolerateErrors () [inline, static]

Getter of mTolerateErrors. Returns value of mTolerateErrors.

Returns:
The value of mTolerateErrors

4.48.4.10 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

4.48.4.11 const u32 & getWidth () const [inline]

Getter of mWidth. Returns value of mWidth.

Returns:
The value of mWidth

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

Getter of mXMTInitialized. Returns value of mXMTInitialized.

Returns:
The value of mXMTInitialized

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

Getter of mXMTSupported. Returns value of mXMTSupported.

Returns:
The value of mXMTSupported

4.48.4.14 void initialize () [virtual]

Initialize X display.

4.48.4.15 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. 237) initialization.)

4.48.4.16 void setTolerateErrors (const bool & *tolerateerrors*) [inline, static]

Setter of mTolerateErrors. Sets value of mTolerateErrors.

Parameters:
← ***tolerateerrors*** The value of mTolerateErrors

4.48.4.17 void synchronize (const bool & *discard* = false) [virtual]

Flush the output buffer and then waits until all requests have been received and processed by the X server.

Parameters:
← ***discard*** Indicates whether **synchronize**()(p. 241) discards all events on the event queue

4.48.5 Member Data Documentation

4.48.5.1 const s32 IGNOREMULTISAMPLE = -1 [static]

Constant for ignoring multisample.

Remarks:
This is own attribute of this class.

4.48.5.2 ::Display* mDisplay [protected]

Wrapped X Display.

Remarks:
This attribute references an attribute.

4.48.5.3 std::string mDisplayName [protected]

X display name.

Remarks:
This is own attribute of this class.

4.48.5.4 Mutex mErrorHandlerMutex [static, protected]

Mutex(p. 132) for synchronization the class level error handling.

Remarks:
This is own attribute of this class.

4.48.5.5 u32 mHeight [protected]

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

Remarks:
This is own attribute of this class.

4.48.5.6 bool mInitialized [protected]

The display is initialized.

Remarks:
This is own attribute of this class.

4.48.5.7 bool mTolerateErrors = false [static, protected]

Skip Xlib errors. Do not use it.

Remarks:
This is own attribute of this class.

4.48.5.8 u32 mWidth [protected]

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

Remarks:
This is own attribute of this class.

4.48.5.9 bool mXMTInitialized = false [static, protected]

X multithreading is initialized.

Remarks:
This is own attribute of this class.

4.48.5.10 bool mXMTSupported = false [static, protected]

X multithreading is supported.

Remarks:
This is own attribute of this class.

4.48.5.11 const s32 UNKNOWNDIMENSION = 0 [static]

Constant for unknown display dimension.

Remarks:
This is own attribute of this class.

4.49 XDisplay::VisualAttribs Struct Reference

4.49.1 Detailed Description

Struct for visual attributes.

4.49.2 Member Data Documentation

4.49.2.1 int accumAlphaSize

4.49.2.2 int accumBlueSize

4.49.2.3 int accumGreenSize

4.49.2.4 int accumRedSize

4.49.2.5 int alphaSize

4.49.2.6 int auxBuffers

4.49.2.7 int bitsPerRGB

4.49.2.8 int blueMask

4.49.2.9 int blueSize

4.49.2.10 int bufferSize

4.49.2.11 int colormapSize

4.49.2.12 int depth

4.49.2.13 int depthSize

4.49.2.14 int doubleBuffer

4.49.2.15 int greenMask

4.49.2.16 int greenSize

4.49.2.17 int id

4.49.2.18 int klass

4.49.2.19 int level

4.49.2.20 int numMultisample

4.49.2.21 int numSamples

4.49.2.22 int redMask

4.49.2.23 int redSize

4.49.2.24 int rgba

4.49.2.25 int stencilSize

4.49.2.26 int stereo

4.49.2.27 int supportsGL

4.49.2.28 int transparentAlphaValue

4.49.2.29 int transparentBlueValue

4.49.2.30 int transparentGreenValue

Index