Tiny Multimedia Framework

1.0

Generated by Doxygen 1.8.6

Sun Aug 31 2014 14:18:32

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Class Documentation

3.1 Buffer< Type > Class Template Reference

```
#include <Buffer.h>
```

Public Member Functions

- Buffer (int size)
- void insert (Type ∗e)
- Type ∗ getNode ()
- int getSize ()
- Type ∗ getNode (int i)
- Type ∗ getNextNode ()
- ∼Buffer ()

3.1.1 Detailed Description

template<class Type>class Buffer< Type >

Buffer is a circular list of data. Buffer is used in output ports.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 template<class Type> Buffer< Type >::Buffer (int *size*) `[inline]`

Buffer constructor

Parameters

size	the size of the buffer

3.1.2.2 template<class Type> Buffer< Type >::∼Buffer ( ) `[inline]`

Buffer destructor

3.1.3 Member Function Documentation

3.1.3.1 template<class Type> Type∗ Buffer< Type >::getNextNode ( ) [inline]

Get the next element of the buffer. Used when the node is a reference and the client of the buffer wants to initialize the node.

Returns

the next element of the buffer

3.1.3.2 template<class Type> Type∗ Buffer< Type >::getNode ( ) [inline]

Get the current node in the buffer

Returns

the current element of the buffer

3.1.3.3 template<class Type> Type∗ Buffer< Type >::getNode (int *i*) [inline]

Get an element of the buffer by index

Parameters

i	the number of the element

Returns

the element number i

3.1.3.4 template<class Type> int Buffer< Type >::getSize ( ) [inline]

Get the size of the buffer

Returns

the size of the buffer

3.1.3.5 template<class Type> void Buffer< Type >::insert (Type ∗ *e*) [inline]

Insert an element into the buffer

Parameters

e	the element to be inserted

The documentation for this class was generated from the following file:

- core/Buffer.h

3.2 Filter Class Reference

```
#include <Filter.h>
```

Public Member Functions

- virtual FilterStatus init ()
- void setProp (const string &key, const string &val)
- string getProp (const string &key)
- void connectFilter (Filter ∗f)
- FilterStatus executeFilter ()
- FilterStatus initFilter (Message ∗msg)
- void increaseLinked ()
- int inputPortNum ()
- int outputPortNum ()
- virtual ∼Filter ()

Protected Member Functions

- Filter (const string &name)
- virtual FilterStatus process ()=0

Protected Attributes

- Message ∗ inMsg
- Message ∗ outMsg
- vector< Port ∗ > inputPorts
- vector< Port ∗ > outputPorts

3.2.1 Detailed Description

Abstraction of a filter in a pipeline. Every concrete filter inherits from filter and can be connected to multiple filters, and receive various data from predecessor filters and send data to accessor filter.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Filter::Filter (const string & *name*) `[protected]`

Filter constructor

Parameters

name	The name of the filter.

3.2.2.2 Filter::∼Filter ( ) `[virtual]`

Destructor of the filter.

3.2.3 Member Function Documentation

3.2.3.1 void Filter::connectFilter (Filter ∗ *f*)

Connect this filter to another filter in the pipeline. It is used by pipeline. User must use Pipeline::connectFilters

Parameters

f	The filter to connect to.

3.2.3.2 FilterStatus Filter::executeFilter ( )

Execute the processing of this filter. The filters are connected by a link list and each filter calls executeFilter of the next filter.

Returns

The new status of the filter.

3.2.3.3 string Filter::getProp (const string & *key*)

Get the value of a filter property.

Parameters

key	The property name.

3.2.3.4 void Filter::increaseLinked ( )

Increase the number of the linked filters.

3.2.3.5 virtual FilterStatus Filter::init ( ) `[inline]`, `[virtual]`

Perform initialization of the filter. To be overridden in subclasses to allow initialization of specific filter values.

3.2.3.6 FilterStatus Filter::initFilter (Message ∗ *msg*)

Execute the init of this filter. The filters are connected by a link list and each filter calls initFilter of the next filter.

Returns

The new status of the filter.

3.2.3.7 int Filter::inputPortNum ( )

Get the number of input ports.

3.2.3.8 int Filter::outputPortNum ( )

Get the number of output port.

3.2.3.9 virtual FilterStatus Filter::process ( ) `[protected]`, `[pure virtual]`

Virtual function, to be implemented in the subclass filters. Read data from input filter, process the data, and write the result to the output port.

3.2.3.10 void Filter::setProp (const string & *key,* const string & *val*)

Set a property of the filter.

Parameters

key	The property name.
val	The property value.

3.2.4 Member Data Documentation

3.2.4.1 Message∗ Filter::inMsg `[protected]`

Input message of the filter

3.2.4.2 vector< Port∗ > Filter::inputPorts `[protected]`

List of the input ports

3.2.4.3 Message∗ Filter::outMsg `[protected]`

Output message of the filter

3.2.4.4 vector< Port∗ > Filter::outputPorts `[protected]`

List of the output ports

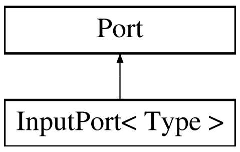
The documentation for this class was generated from the following files:

- core/Filter.h
- core/Filter.cpp

3.3 InputPort< Type > Class Template Reference

```
#include <Port.h>
```

Inheritance diagram for InputPort< Type >:



Public Member Functions

- InputPort (string name, Filter ∗owner)
- void consume (Type ∗bn)
- Type ∗ read ()
- ∼InputPort ()

Additional Inherited Members

3.3.1 Detailed Description

template<class Type>class InputPort< Type >

InputPort class is a subclass of the Port class. It is a class template and the type of the buffer of the port is a template.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 template<class Type > InputPort< Type >::InputPort (string *name*, Filter ∗ *owner*) [inline]

InputPort constructor

Parameters

name	The name of the port
owner	The owner of the port

3.3.2.2 template<class Type > InputPort< Type >::∼InputPort ( ) [inline]

InputPort destructor

3.3.3 Member Function Documentation

3.3.3.1 template<class Type > void InputPort< Type >::consume (Type ∗ *bn*) [inline]

Consume a data coming from The output port calls the consume of input port and owner of this port is executed

Parameters

bn	the data to be consumed

3.3.3.2 template<class Type > Type∗ InputPort< Type >::read ( ) [inline]

Read data from the port

Returns

input buffer of the port

The documentation for this class was generated from the following file:

- core/Port.h

3.4 Message Class Reference

`#include <Message.h>`

Public Member Functions

- void setProp (const string &key, const string &val)
- MessageError getPropInt (const string &key, int &val)
- MessageError getPropString (const string &key, string &val)
- void setPropInt (const string &key, const int &val)

3.4.1 Detailed Description

Message to communicate between the filters.

3.4.2 Member Function Documentation

3.4.2.1 MessageError Message::getPropInt (const string & *key*, int & *val*) `[inline]`

Get the integer message by passing the key

Parameters

key	the key of the message
val	reference to receive the value of the message

Returns

MSG_OK if the message is found and MSG_NOT_FOUND if the message is not found.

3.4.2.2 MessageError Message::getPropString (const string & *key*, string & *val*) `[inline]`

Get the string message by passing the key

Parameters

key	the key of the message
val	reference to receive the value of the message

Returns

MSG_OK if the message is found and MSG_NOT_FOUND if the message is not found.

3.4.2.3 void Message::setProp (const string & *key*, const string & *val*) `[inline]`

Set the string message by key and value

Parameters

key	the key of the message
val	the string value of the message

3.4.2.4 void Message::setPropInt (const string & *key*, const int & *val*) `[inline]`

Set the integer message by key and value

Parameters

key	the key of the message
val	the integer value of the message

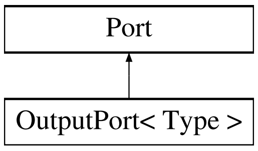
The documentation for this class was generated from the following file:

- core/Message.h

3.5 OutputPort< Type > Class Template Reference

```
#include <Port.h>
```

Inheritance diagram for OutputPort< Type >:



Public Member Functions

- OutputPort (string name, Filter ∗owner)
- void produce (Type ∗bn)
- Buffer< Type > ∗ getBuffer ()
- void process ()
- ∼OutputPort ()

Additional Inherited Members

3.5.1 Detailed Description

template<class Type>class OutputPort< Type >

OutputPort class is a subclass of the Port class. It is a class template and the type of the buffer of the port is a template.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 template<class Type > **OutputPort< Type >::OutputPort (string *name,* Filter ∗ *owner*)** `[inline]`

OutputPort constructor

Parameters

name	The name of the output port
owner	The owner of the port

3.5.2.2 template<class Type > **OutputPort< Type >::∼OutputPort ( )** `[inline]`

OutputPort desctructor

3.5.3 Member Function Documentation

3.5.3.1 template<class Type > **Buffer<Type>∗ OutputPort< Type >::getBuffer ( )** `[inline]`

Get buffer

Returns

the output port buffer

3.5.3.2 template<class Type > void OutputPort< Type >::process ( ) [inline]

Process the port It calls consume function of the next ports and therefore executes the next filters

3.5.3.3 template<class Type > void OutputPort< Type >::produce (Type ∗ *bn*) [inline]

Produce data This function produce data on the output buffer

Parameters

bn	data to be produced

The documentation for this class was generated from the following file:

- core/Port.h

3.6 Pipeline Class Reference

```
#include <Pipeline.h>
```

Public Member Functions

- Pipeline (const string &name)
- void connectFilters (Filter ∗fi, Filter ∗fo)
- PipelineStatus init ()
- PipelineStatus run ()
- ∼Pipeline ()

3.6.1 Detailed Description

A pipeline, consisting of a number of interconnected filters. Filters have a many-to-many relation, with directed pipes. Cycles are not allowed.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 Pipeline::Pipeline (const string & *name*)

Pipeline constructor

Parameters

name	The name of the pipeline.

3.6.2.2 Pipeline::∼Pipeline ( )

Pipeline destructor

3.6.3 Member Function Documentation

3.6.3.1 void Pipeline::connectFilters (Filter ∗ *fi,* Filter ∗ *fo*)

Create a pipe between two filters in the pipeline. These filters should be in the pipeline.

Parameters

fi	The source filter for the pipe.
fo	The target filter for the pipe.

3.6.3.2 PipelineStatus Pipeline::init ( )

Initialize the pipeline.

Returns

the current pipeline status.

3.6.3.3 PipelineStatus Pipeline::run ( )

Run one iteration of the pipeline.

Returns

the current pipeline status.

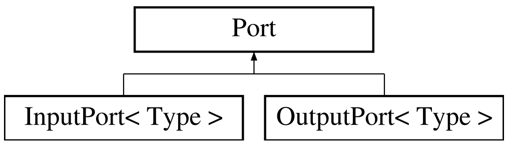
The documentation for this class was generated from the following files:

- core/Pipeline.h
- core/Pipeline.cpp

3.7 Port Class Reference

`#include <Port.h>`

Inheritance diagram for Port:



Public Member Functions

- Port (string name, Filter ∗owner)
- string getName ()
- int getLinked ()
- void increaseLinked ()
- string getType ()
- Filter ∗ getOwner ()
- vector< Port ∗ > & getNextPorts ()
- void addNextPort (Port ∗n)
- virtual ∼Port ()

Protected Attributes

- string type
- Filter $*$ owner
- vector$<$ Port $*$ $>$ nextPorts

3.7.1 Detailed Description

Abstraction of a port in a filter. A port can be either input port of output port.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 Port::Port (string *name,* Filter $*$ *owner*) [inline]

Port constructor

Parameters

name	The name of the filter.
owner	The owner of the filter

3.7.2.2 virtual Port::$\sim$Port ( ) [inline], [virtual]

Port descructor

3.7.3 Member Function Documentation

3.7.3.1 void Port::addNextPort (Port $*$ *n*) [inline]

Add next port to this port

Parameters

n	next port to connect to

3.7.3.2 int Port::getLinked ( ) [inline]

Get the number of the ports connected to this port

Returns

the number of the port.

3.7.3.3 string Port::getName ( ) [inline]

Get the name of the port

Returns

the name of the port.

3.7.3.4 vector<Port*>& Port::getNextPorts ( ) [inline]

Get next ports

Returns

the next ports

3.7.3.5 Filter* Port::getOwner ( ) [inline]

Get the owner of the port

Returns

the owner of the filter

3.7.3.6 string Port::getType ( ) [inline]

Get the type of the port

Returns

the type of the port

3.7.3.7 void Port::increaseLinked ( ) [inline]

Increase the number of the ports linked to the port. (The filter uses this function when it connects two filters)

3.7.4 Member Data Documentation

3.7.4.1 vector<Port*> Port::nextPorts [protected]

A list of the next ports. A subclass filter must add its filters to this list

3.7.4.2 Filter* Port::owner [protected]

The filter which owns this port

3.7.4.3 string Port::type [protected]

The type of the buffer of the port (note: the typeid is used to retrieve the type of the buffer and the type name is not complete. This is used when a filter wants to connect ports and needs to know the type of the ports)

The documentation for this class was generated from the following file:

- core/Port.h

Index