A Description of the Classes and Methods of the Porsonify Audio Software

The Picasso Research Group
University of Illinois at Urbana-Champaign
Department of Computer Science
1304 West Springfield Avenue
Urbana, Illinois 61801

October 1, 1992

Copyright (c) 1992 The University of Illinois Board of Trustees.
All Rights Reserved.

CONFIDENTIAL INFORMATION.
Distribution restricted under license agreement.

This work was funded in part by NASA ICLASS Contract No. NAG–1–613, National Science Foundation grants NSF CCR86–57696, NSF CCR87–06653 and NSF CDA87–22836 (Tapestry), DARPA Contract No. DAVT63–91–C–0029, and by grants from the Digital Equipment Corporation External Research Program and Apple Computer.

Name

ArrayType – Implements the basic string type for network data values.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – DataType, Data, SoundObj.

Description

ArrayType is the primitive Val type for a string.

Available Methods

Public

- binReceive – Receive binary representation of data
- binSend – Send binary representation of data
- char – Return first character of string
- char* – Return string as itself
- double – Return string as double
- duplicateString (inherited) – Return a copy of string
- enumAsString (inherited) – Return a particular enumerate as a string
- float – Return string as float.
- getLowerBound (inherited) – Returns the lower bound, if bounded
- getUpperBound (inherited) – Returns the upper bound, if bounded
- int – Return string as int
- isBounded (inherited) – Must this type be in a specific range?
- operator= – Assignment operator
- print – Return a string representation of string
- printOn – Stream output function
- recString (inherited) – Receive and allocate space for a string
- receive (inherited) – Receive data from fd
- send (inherited) – Write data to fd
- setBounds (inherited) – Set upper and lower bounds
- setValueFromDouble – Set the value of string from double
- setValueFromString – Set data type value from string

Included Files

"DataType.h"

Public Methods

ArrayType(char *string);

The ArrayType constructor.

ArrayType( ) ;

The ArrayType constructor.

~**ArrayType**( );

The ArrayType destructor.

ArrayType& **operator**=(const ArrayType& at) ;

Assignment operator

virtual void **setValueFromString**(const char *string);

Set data type value from string

virtual void **binSend**(int sfd);

Send binary representation of data

virtual void **binReceive**(int sfd);

Receive binary representation of data

virtual char* **print**( );

Return a string representation of string

virtual void **setValueFromDouble**(double d);

Set the value of string from double

virtual operator **int**( ) ;

Return first character of string

virtual operator **char**( ) ;

Return first character of string

`virtual operator` **float**`( ) ;`

Return string as float.

`virtual operator` **double**`( ) ;`

Return string as double

`virtual operator` **char***`( ) ;`

Return string as itself

`void` **printOn**`( ostream &os = cout);`

Stream output function

Name

MultiSonicWidget – Implements the MultiSonicWidget.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – BaseSonicWidget, SCFParsing, SoundObj.

Description

MultiSonicWidget encapsulates a list of Sound Control Files and a Transform Function. It recalculates an integer index from the Transform Function and a scalar data value, and uses this to select one of the list of Sound Control Files. That Sound Control File is executed with the original Data value as an argument.

If the index is out of range, nothing is executed.

Available Methods

Protected

- looksValid (inherited) – Check that a sound control file ends in .scf.

Public

- configure (inherited) – Called when Sonic Widget is configured if any special actions are required.
- duplicateString (inherited) – Return a copy of string
- eatComment (inherited) – Get a comment from the input stream
- execute – Execute the sonic widget with scalar or vector
- getHeader (inherited) – Get the header of a sound control file.
- getId (inherited) – Read an identifier from the input stream
- getName (inherited) – Return the name of the sonic widget.
- getSCF (inherited) – Return one of the constituant sound control files.
- getTok (inherited) – Get a token from the input stream
- getTransformFunction (inherited) – Return the transform function for this sonic widget.
- initialize – Initialize the sonic widget from a widget control file that has been positioned past the sonic widget class identifier.
- myClassName – Return the name of the widget class
- numSCFs (inherited) – Return the number of sound control files managed.
- peekTok (inherited) – Peek at a token without removing it from the stream
- printOn – Stream output function
- setBlocking (inherited) – Turn blocking on or off.

- setTransformFunction (inherited) – Set the transform function for this sonic widget.
- writeParamsResourceFile (inherited) – Write WCL resource file for SCF child, returning filename.
- writeSCF (inherited) – Write out a specific sound control file.
- writeWCF (inherited) – Write out widget control file.

Included Files

"BaseSonicWidget.h"
"SoundControlFile.h"

Public Methods

`virtual void` **execute(** `double d);`

Execute the sonic widget with scalar or vector

`virtual void` **execute(** `double *a, int length);`

Execute the sonic widget with scalar or vector

`virtual Bool_` **initialize(** `istream &is);`

Initialize the sonic widget from a widget control file that has been positioned past the sonic widget class identifier.

`virtual const char*` **myClassName(** `) const ;`

Return the name of the widget class

`virtual void` **printOn(** `ostream &os = cout);`

Stream output function

Name

BaseSonicWidget – The base class for Sonic Widgets.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SCFParsing, SoundObj.

Description

BaseSonicWidget is the base class for all Sonic Widgets. It does nothing except define the virtual execute procedures which all derived Sonic widgets must support.

Available Methods

Protected
- looksValid – Check that a sound control file ends in .scf.

Public
- configure – Called when Sonic Widget is configured if any special actions are required.
- duplicateString (inherited) – Return a copy of string
- eatComment (inherited) – Get a comment from the input stream
- execute – Execute the sonic widget with scalar or vector.
- getHeader – Get the header of a sound control file.
- getId (inherited) – Read an identifier from the input stream
- getName – Return the name of the sonic widget.
- getSCF – Return one of the constituant sound control files.
- getTok (inherited) – Get a token from the input stream
- getTransformFunction – Return the transform function for this sonic widget.
- initialize – Initialize the sonic widget from a widget control file that has been positioned past the sonic widget class identifier.
- myClassName – Return the name of the widget class.
- numSCFs – Return the number of sound control files managed.
- peekTok (inherited) – Peek at a token without removing it from the stream
- printOn – Stream output function
- setBlocking – Turn blocking on or off.
- setTransformFunction – Set the transform function for this sonic widget.
- writeParamsResourceFile – Write WCL resource file for SCF child, returning filename.
- writeSCF – Write out a specific sound control file.
- writeWCF – Write out widget control file.

Included Files

"SCFParsing.h"
"SoundControlFile.h"
"Val.h"

Protected Methods

`Bool_` **looksValid**`( char *soundControlFile );`

Check that a sound control file ends in .scf.

Public Methods

BaseSonicWidget`( char *name = "");`

The BaseSonicWidget constructor.

~BaseSonicWidget`( );`

The BaseSonicWidget destructor.

`virtual void` **configure**`( ) ;`

Called when Sonic Widget is configured if any special actions are required.

`virtual void` **execute**`( double d) = 0 ;`

Execute the sonic widget with scalar or vector.

`virtual void` **execute**`( double *a, int length) = 0 ;`

Execute the sonic widget with scalar or vector.

`virtual Bool_` **getHeader**`( char *soundControlFile, char *hostname, char *portname);`

Get the header of a sound control file.

`virtual char*` **getName**`( ) const ;`

Return the name of the sonic widget.

`virtual SoundControlFile*` **getSCF**`( int which) ;`

Return one of the constituant sound control files.

`virtual const char*` **myClassName**`( ) const ;`

Return the name of the widget class.

```
virtual int numSCFs(    ) const ;
```

Return the number of sound control files managed.

```
virtual void setBlocking(    int ack);
```

Turn blocking on or off.

```
virtual char* writeParamsResourceFile(    char *directory, int which = 0 );
```

Write WCL resource file for SCF child, returning filename.

```
virtual void writeWCF(    ostream &os) const;
```

Write out widget control file.

```
virtual void writeSCF(    ostream &os, int which = 0 ) const;
```

Write out a specific sound control file.

```
virtual DataTransform* getTransformFunction(    ) ;
```

Return the transform function for this sonic widget.

```
virtual void setTransformFunction(    DataTransform *tf) ;
```

Set the transform function for this sonic widget.

```
virtual Bool_ initialize(    istream &is) = 0;
```

Initialize the sonic widget from a widget control file that has been positioned past the sonic widget class identifier.

```
virtual Bool_ initialize(    SoundControlFile *widgetControlFile ...);
```

Initialize the sonic widget from sound control file(s).

```
virtual Bool_ initialize(    SoundControlFile *widgetControlFile);
```

Initialize the sonic widget from sound control file(s).

```
virtual void printOn(    ostream &os = cout);
```

Stream output function

Name

Oscil – Implements a basic oscillator used for SPARC synthesis.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SoundObj.

Description

Oscil is an implementation of a basic oscillator which accepts a wavetable as input or generates its own.

Available Methods

Public
- bsamples – Generate a block of samples.
- duplicateString (inherited) – Return a copy of string
- printOn (inherited) – Stream output function

Included Files

"SoundObj.h"
"WaveTable.h"

Public Methods

Oscil(`WaveTable *wave);`

The Oscil constructor.

Oscil(`int samplerate, int tabsize=512);`

The Oscil constructor.

~Oscil(`) ;`

The Oscil destructor.

`Samples* bsamples( double frequency, double duration);`

Generate a block of samples.

`Samples* bsamples( char note, double duration);`

Generate a block of samples.

Name

BaseSoundDevice – Implements a pseudo-sound device for applications that expect one.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SoundObj.

Description

This isn't really a "base class" for sound devices, since sound devices can support such radically different kinds of methods. It is the pseudo-sound device used to build applications that do not require a "real" sound device.

Available Methods

Public
- duplicateString (inherited) – Return a copy of string
- printOn – Stream output function

Included Files

"SoundObj.h"

Public Methods

BaseSoundDevice( );

The BaseSoundDevice constructor.

~BaseSoundDevice( );

The BaseSoundDevice destructor.

`void` **printOn(** `ostream &os = cout);`

Stream output function

Name

PassThrough – Implements the simplest transformation function.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – DataTransform, SoundObj.

Description

PassThrough is the simplest "transformation", passing the data value through unchanged.

Available Methods

Public

- calculateValue – Set value of Val result from variable
- duplicateString (inherited) – Return a copy of string
- idNumber – Return the id number of this transform
- printConfiguration – Return a string starting with header that describes configuration
- printOn – Stream output function
- readConfiguration – Get params from input stream
- setValue (inherited) – Virtual set value of internal parameters

Included Files

"DataTransform.h"
"Val.h"

Public Methods

`Bool_` **readConfiguration(** `istream &is)` ;

Get params from input stream

`void` **calculateValue(** `void *variable, void *result)` ;

Set value of Val result from variable

`virtual int` **idNumber**`( ) ;`

Return the id number of this transform

`char*` **printConfiguration**`( char *header);`

Return a string starting with header that describes configuration

`void` **printOn**`( ostream &os = cout);`

Stream output function

Name

BoolType – Implements a basic Val boolean type.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – IntType, DataType, Data, SoundObj.

Description

BoolType is the basic boolean type, used by the Val class.

Available Methods

Public

- binReceive (inherited) – Receive binary representation of data
- binSend (inherited) – Send binary representation of data
- char (inherited) – Return IntType as character
- char* (inherited) – Return IntType as character string
- double (inherited) – Return IntType as double
- duplicateString (inherited) – Return a copy of string
- enumAsString (inherited) – Return a particular enumerate as a string
- float (inherited) – Return IntType as float
- getLowerBound (inherited) – Returns the lower bound, if bounded
- getUpperBound (inherited) – Returns the upper bound, if bounded
- int (inherited) – Return IntType as integer
- isBounded (inherited) – Must this type be in a specific range?
- print – Return a string representation of DataType
- printOn – Stream output function
- recString (inherited) – Receive and allocate space for a string
- receive (inherited) – Receive data from fd
- send (inherited) – Write data to fd
- setBounds – Set upper and lower bounds
- setValueFromDouble (inherited) – Set the value of DataType from double
- setValueFromString – Set data type value from string

Included Files

"IntType.h"

Public Methods

`virtual void` **setValueFromString(** `const char *string );`

Set data type value from string

`virtual Bool_` **setBounds(** `const char l, const char u );`

Set upper and lower bounds

`virtual char*` **print(** `);`

Return a string representation of DataType

`void` **printOn(** `ostream &os = cout);`

Stream output function

Name

PtrList – Implements a list of pointers to things.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – Data, SoundObj.

Description

Implements a list of void pointers to things. It is the base class for other typed lists.

Available Methods

Protected
- computeNewSize – Determine new size for list resizing
- growList – Enlarge list, based on index n
- resizeListTo – Resize list to given size

Public
- addElement – Add pointer to end of list
- binReceive (inherited) – Receive a binary representation of this object.
- binSend (inherited) – Send a binary representation of this object.
- deleteElement – Delete element from list, returning it
- duplicateString (inherited) – Return a copy of string
- getElement – Return element from list
- printOn (inherited) – Stream output function
- recString (inherited) – Receive and allocate space for a string
- receive (inherited) – Receive data from fd
- send (inherited) – Write data to fd
- setElement – Assign to a specific element of the list

Included Files

"SoundObj.h"
"Data.h"

Protected Methods

`int` **computeNewSize**`( int requestedIndex);`

Determine new size for list resizing

`void` **growList**`( int n);`

Enlarge list, based on index n

`Bool_` **resizeListTo**`( int n);`

Resize list to given size

Public Methods

PtrList`( int initialSize = 10);`

The PtrList constructor.

~PtrList`( );`

The PtrList destructor.

`virtual int` **addElement**`( void *arg);`

Add pointer to end of list

`virtual int` **setElement**`( int n, void *arg);`

Assign to a specific element of the list

`virtual void*` **getElement**`( int n) const;`

Return element from list

`virtual void*` **deleteElement**`( int n);`

Delete element from list, returning it

Name

BooleanMenu – Implements a boolean transformation.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – DataTransform, SoundObj.

Description

BooleanMenu is a transformation which performs one of six boolean comparisons between a scalar data value and a constant which may be set interactively.

Available Methods

Private
- compute – Does the actual computation
- setOperation – Sets the operation

Public
- calculateValue – Sets value of Val result according to variable
- duplicateString (inherited) – Return a copy of string
- idNumber – Return the id number of this transform
- printConfiguration – Return a string starting with header that describes configuration
- printOn – Stream output function
- readConfiguration – Get params from input stream
- setValue – Set value of internal parameters

Included Files

"DataTransform.h"
"Val.h"

Private Methods

`double compute( double x);`

Does the actual computation

`void` **setOperation(** `int op);`

Sets the operation

Public Methods

`void` **calculateValue(** `void *variable, void *result);`

Sets value of Val result according to variable

`virtual int` **idNumber(** `) ;`

Return the id number of this transform

`void` **setValue(** `char *args, double value);`

Set value of internal parameters

`void` **setValue(** `char *args, char *value);`

Set value of internal parameters

`Bool_` **readConfiguration(** `istream &is);`

Get params from input stream

`char*` **printConfiguration(** `char *header);`

Return a string starting with header that describes configuration

`void` **printOn(** `ostream &os = cout);`

Stream output function

Name

RemoteFileType – Implements the Val type for remote files.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – EnumVal, IntType, DataType, Data, SoundObj.

Description

RemoteFileType is used by Val to represent a remote file. It is basically an index into an enumerated list of files. This is why it is derived from EnumVal.

Available Methods

Public

- binReceive (inherited) – Receive binary representation of data
- binSend (inherited) – Send binary representation of data
- char (inherited) – Return IntType as character
- char* (inherited) – Return IntType as character string
- double (inherited) – Return IntType as double
- duplicateString (inherited) – Return a copy of string
- enumAsString (inherited) – Return a particular enumerate as a string
- float (inherited) – Return IntType as float
- getLowerBound (inherited) – Returns the lower bound, if bounded
- getUpperBound (inherited) – Returns the upper bound, if bounded
- int (inherited) – Return IntType as integer
- isBounded (inherited) – Must this type be in a specific range?
- print (inherited) – Return a string representation of DataType
- printOn – Stream output function
- recString (inherited) – Receive and allocate space for a string
- receive (inherited) – Receive data from fd
- send (inherited) – Write data to fd
- setBounds (inherited) – Set upper and lower bounds
- setValueFromDouble (inherited) – Set the value of EnumVal from double
- setValueFromString (inherited) – Set data type value from string

Included Files

"EnumVal.h"

Public Methods

```
void printOn(   ostream &os = cout);
```

Stream output function

Name

BooleanSonicWidget – Implements a boolean sonic widget.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – BaseSonicWidget, SCFParsing, SoundObj.

Description

BooleanSonicWidget encapsulates a single Sound Control File and a Transform Function. The "execute" procedure calls execute on the Sound Control File with the provided data only if the transform function generates a value that is greater than 0 (true).

Available Methods

Protected
- looksValid (inherited) – Check that a sound control file ends in .scf.

Public
- configure (inherited) – Called when Sonic Widget is configured if any special actions are required.
- duplicateString (inherited) – Return a copy of string
- eatComment (inherited) – Get a comment from the input stream
- execute – Execute the sonic widget with scalar or vector
- getHeader (inherited) – Get the header of a sound control file.
- getId (inherited) – Read an identifier from the input stream
- getName (inherited) – Return the name of the sonic widget.
- getSCF (inherited) – Return one of the constituant sound control files.
- getTok (inherited) – Get a token from the input stream
- getTransformFunction (inherited) – Return the transform function for this sonic widget.
- initialize – Initialize the sonic widget from a widget control file that has been positioned past the sonic widget class identifier.
- myClassName – Return the name of the widget class
- numSCFs (inherited) – Return the number of sound control files managed.
- peekTok (inherited) – Peek at a token without removing it from the stream
- printOn – Stream output function
- setBlocking (inherited) – Turn blocking on or off.
- setTransformFunction (inherited) – Set the transform function for this sonic widget.
- writeParamsResourceFile (inherited) – Write WCL resource file for SCF child, returning

filename.
- writeSCF (inherited) – Write out a specific sound control file.
- writeWCF (inherited) – Write out widget control file.

Included Files

"BaseSonicWidget.h"
"SoundControlFile.h"

Public Methods

`virtual void` **execute**`( double d);`

Execute the sonic widget with scalar or vector

`virtual void` **execute**`( double *a, int length);`

Execute the sonic widget with scalar or vector

`virtual Bool_` **initialize**`( istream &is);`

Initialize the sonic widget from a widget control file that has been positioned past the sonic widget class identifier.

`virtual const char*` **myClassName**`( ) const ;`

Return the name of the widget class

`virtual void` **printOn**`( ostream &os = cout);`

Stream output function

Name

SCFParsing – Implements sound control file parsing primitives.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SoundObj.

Description

SCFParsing implements some sound control file parsing primitives.

Available Methods

Public

- duplicateString (inherited) – Return a copy of string
- eatComment – Get a comment from the input stream
- getId – Read an identifier from the input stream
- getTok – Get a token from the input stream
- peekTok – Peek at a token without removing it from the stream
- printOn – Stream output function

Included Files

<ctype.h>
"SoundObj.h"

Public Methods

SCFParsing( ) ;

The SCFParsing constructor.

~SCFParsing( ) ;

The SCFParsing destructor.

`Bool_` **eatComment(** `istream &is);`

Get a comment from the input stream

`Tok` **getTok(** `istream &is, char *ret);`

Get a token from the input stream

`Tok` **getTok(** `istream &is, int &ret);`

Get a token from the input stream

`Tok` **peekTok(** `istream &is);`

Peek at a token without removing it from the stream

`Tok` **getId(** `istream &is, char *ret);`

Read an identifier from the input stream

`void` **printOn(** `ostream &os = cout);`

Stream output function

Name

Buffer – Implements a buffer to be used in network traffic.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SoundObj.

Description

Buffer implements a buffer to be used by Socket.

Available Methods

Public
- duplicateString (inherited) – Return a copy of string
- getData – Return the data in the buffer
- getLength – Return the length of the buffer
- printOn (inherited) – Stream output function
- setData – Initialize the data in the buffer

Included Files

"SoundObj.h"

Public Methods

Buffer(`int len = DefaultLength);`

The Buffer constructor

~Buffer(`);`

The Buffer destructor

`int` **getLength(** `) ;`

Return the length of the buffer

```
void* getData(   ) ;
```

Return the data in the buffer

```
void setData(   void *data, int l);
```

Initialize the data in the buffer

Name

SPARCSoundDevice – Implements the base sound device for the SPARC audio chip.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SoundObj.

Description

SPARCSoundDevice implements the base sound device for the SPARC audio chip.

Available Methods

Public

- duplicateString (inherited) – Return a copy of string
- playNote – Play a note
- playNumber – Say a number
- playSampleFile – Play a sample file
- printOn – Stream output function
- reinitialize – Reinitialize the audio device
- selectInstrument – Select the instrument
- setOutputPort – Select the output port
- setRecordVolume – Set the recording volume
- setVolume – Set the main volume

Included Files

"SoundObj.h"
"ValList.h"
"SparcAudioHardware.h"

Public Methods

SPARCSoundDevice( );

The SPARCSoundDevice constructor

~SPARCSoundDevice( );

The SPARCSoundDevice destructor

`void` **setVolume**(`ValList &`);

Set the main volume

`void` **setRecordVolume**(`ValList &`);

Set the recording volume

`void` **reinitialize**( );

Reinitialize the audio device

`void` **selectInstrument**(`ValList &`);

Select the instrument

`void` **setOutputPort**(`ValList &`);

Select the output port

`void` **playNote**(`ValList &`);

Play a note

`void` **playNumber**(`ValList &`);

Say a number

`void` **playSampleFile**(`ValList &`);

Play a sample file

`void` **printOn**(`ostream &os = cout`);

Stream output function

Name

CharType – Implements the char type used by Val.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – DataType, Data, SoundObj.

Description

CharType is the basic type used for a character by the Val class.

Available Methods

Public

- binReceive – Receive binary representation of data
- binSend – Send binary representation of data
- char – Return CharType as char
- char* – Return pointer to char
- double (inherited) – Return DataType as double
- duplicateString (inherited) – Return a copy of string
- enumAsString (inherited) – Return a particular enumerate as a string
- float – Please describe this method.
- getLowerBound (inherited) – Returns the lower bound, if bounded
- getUpperBound (inherited) – Returns the upper bound, if bounded
- int – Return CharType as integer
- isBounded (inherited) – Must this type be in a specific range?
- print – Return a string representation of DataType
- printOn – Stream output function
- recString (inherited) – Receive and allocate space for a string
- receive (inherited) – Receive data from fd
- send (inherited) – Write data to fd
- setBounds (inherited) – Set upper and lower bounds
- setValueFromDouble – Set the value of DataType from double
- setValueFromString – Set data type value from string

Included Files

"DataType.h"

Public Methods

CharType(`char c) : c(c) ;`

The CharType constructor.

CharType(`) ;`

The CharType constructor.

~CharType(`) ;`

The CharType destructor.

`virtual void` **setValueFromString(** `const char *string );`

Set data type value from string

`virtual void` **binSend(** `int sfd);`

Send binary representation of data

`virtual void` **binReceive(** `int sfd);`

Receive binary representation of data

`virtual char*` **print(** `);`

Return a string representation of DataType

`virtual void` **setValueFromDouble(** `double d);`

Set the value of DataType from double

`virtual operator` **int(** `) ;`

Return CharType as integer

`virtual operator` **char(** `) ;`

Return CharType as char

`virtual operator` **float(** `) ;`

`virtual operator` **char*(** `) ;`

Return pointer to char

`void` **printOn(** `ostream &os = cout);`

Stream output function

Name

SWMultiParams – The X Window System interface for sonic widget configuration parameters.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SmallWCLWidget, SoundObj.

Description

SWMultiParams implements an X interface to configure a sonic widget. The sonic widget should be at most two levels deep.

Available Methods

Private

- centerWidget – Center the widget with respect to a reference widget
- registerCallbacks – Register all the X callbacks
- resetAllToggles – Reset toggles on transform function menu
- writeResourceFiles – Write the WCL resource files to a temporary directory.

Public

- PutLineResourceCB – Put a line in the resource database
- cancelledCallback – Called when TF menu is cancelled
- configureTransformCallback – Configure the TF menu
- create – Create the WCL widget
- dismissedCallback – Called when window is dismissed
- doneRunningCallback – Callback called when finished running
- duplicateString (inherited) – Return a copy of string
- fixupSize – Make sure the popup has the correct size
- generalSWParamsCallback – The basic callback – all callbacks are routed through here
- getCallbackArgs – Get the callback args from a resource in the widget
- getTransformFunction – Return the selected transform function
- initialize (inherited) – Initialize the widget
- printOn – Stream output function
- promptTransformFunctionMenu – Pop up the transform function menu
- registerMemberCallback – Register a member function callback
- revertTransformFunctionCallback – Called to revert to old TF
- run – Run the WCL widget
- selectTransformFunctionCallback – Select a new TF

- updateDefaultCallback – Update default from any widget
- updateDefaultFromListCallback – Update default from list widget
- updateDefaultFromSliderCallback – Update default from scale widget
- updateDefaultFromTextCallback – Update default from text widget
- updateDefaultFromToggleCallback – Update default from toggle button
- updateRevertLabel – Update the revert label in the TF menu
- updateTransformDefaultCallback – update TF defaults

Included Files

<X11/Wc/WcCreate.h>
<X11/StringDefs.h>
"BaseSonicWidget.h"
<X11/Xmp/Table.h>
"SmallWCLWidget.h"
<X11/IntrinsicP.h>
"TransformFunctions.h"
<X11/CoreP.h>
<Xm/Xm.h>

Private Methods

`void` **centerWidget(** `Widget reference, Widget center);`

Center the widget with respect to a reference widget

`void` **resetAllToggles(** `);`

Reset toggles on transform function menu

`void` **registerCallbacks(** `);`

Register all the X callbacks

`void` **writeResourceFiles(** `);`

Write the WCL resource files to a temporary directory.

Public Methods

SWMultiParams(`Widget parent, const char *name, const char *AppName, BaseSonicWidget *SW);`

The SWMultiParams constructor.

~SWMultiParams(`);`

The SWMultiParams destructor.

`void` **create**`(  );`

Create the WCL widget

`void` **fixupSize**`(  Widget it, int i);`

Make sure the popup has the correct size

`Bool_` **getCallbackArgs**`(  char *arglist);`

Get the callback args from a resource in the widget

`void` **promptTransformFunctionMenu**`(  );`

Pop up the transform function menu

`DataTransform*` **getTransformFunction**`(  ) ;`

Return the selected transform function

`void` **run**`(  );`

Run the WCL widget

`static void` **PutLineResourceCB**`(  Widget w, char *line, void *unused);`

Put a line in the resource database

`void` **updateRevertLabel**`(  DataTransform *transform);`

Update the revert label in the TF menu

`void` **doneRunningCallback**`(  Widget w, SWMultiParams *obj, XtPointer);`

Callback called when finished running

`void` **revertTransformFunctionCallback**`(  Widget w, SWMultiParams *obj, XtPointer);`

Called to revert to old TF

`void` **dismissedCallback**`(  Widget w, SWMultiParams *obj, XtPointer);`

Called when window is dismissed

`void` **cancelledCallback**`(  Widget w, SWMultiParams *obj, XtPointer);`

Called when TF menu is cancelled

`void` **configureTransformCallback**`(  Widget w, SWMultiParams *obj, XtPointer);`

Configure the TF menu

`void` **selectTransformFunctionCallback(** `Widget w, SWMultiParams *obj, XmToggleButtonCallbackStruct *sta`

Select a new TF

`void` **updateTransformDefaultCallback(** `Widget w, SWMultiParams *obj, XtPointer);`

update TF defaults

`void` **updateDefaultFromSliderCallback(** `Widget w, SWMultiParams *obj, XtPointer);`

Update default from scale widget

`void` **updateDefaultFromListCallback(** `Widget w, SWMultiParams *obj, XtPointer);`

Update default from list widget

`void` **updateDefaultFromTextCallback(** `Widget w, SWMultiParams *obj, XtPointer);`

Update default from text widget

`void` **updateDefaultFromToggleCallback(** `Widget w, SWMultiParams *obj, XtPointer);`

Update default from toggle button

`void` **updateDefaultCallback(** `Widget w, SWMultiParams *, XtPointer);`

Update default from any widget

`static void` **generalSWParamsCallback(** `Widget w, void *rec, XtPointer);`

The basic callback – all callbacks are routed through here

`void` **registerMemberCallback(** `char *name, void *proc, SWMultiParams *data);`

Register a member function callback

`void` **printOn(** `ostream &os = cout);`

Stream output function

Name

Data – Implements the basic type for all data transmitted across a network.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SoundObj.

Description

Data is the base class for all data structures that are shipped over the network.

Available Methods

Public

- binReceive – Receive a binary representation of this object.
- binSend – Send a binary representation of this object.
- duplicateString (inherited) – Return a copy of string
- printOn – Stream output function
- recString – Receive and allocate space for a string
- receive – Receive data from fd
- send – Write data to fd

Included Files

"SoundObj.h"

Public Methods

Data( ) ;

The Data constructor.

Data(int fd) : fd(fd) ;

The Data constructor.

~Data( ) ;

The Data destructor.

```
virtual int send(   char *string);
```
Write data to fd

```
virtual int send(   const char &c);
```
Write data to fd

```
virtual int send(   const int &i);
```
Write data to fd

```
virtual int send(   const float &f);
```
Write data to fd

```
virtual int send(   const double &d);
```
Write data to fd

```
virtual int receive(   char &c);
```
Receive data from fd

```
virtual int receive(   int &i);
```
Receive data from fd

```
virtual int receive(   float &f);
```
Receive data from fd

```
virtual int receive(   double &d);
```
Receive data from fd

```
char* recString(   );
```
Receive and allocate space for a string

```
virtual void binReceive(   int fd);
```
Receive a binary representation of this object.

```
virtual void binSend(   int fd);
```
Send a binary representation of this object.

```
void printOn(   ostream &os = cout);
```
Stream output function

Name

Samples – Implements various operations on a block of samples.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SoundObj.

Description

The Samples type supports various operations on blocks of samples.

Available Methods

Public

- concat – Concatenate two blocks of samples.
- duplicateString (inherited) – Return a copy of string
- get – Extract a sample.
- operator+ – Mix two blocks of samples.
- printOn (inherited) – Stream output function
- put – Add a sample.
- size – Return the size of the block of samples.

Included Files

"SoundObj.h"

Public Methods

Samples(`int numsamples` **) ;**

The Samples constructor.

~Samples(**) ;**

The Samples destructor.

put(`int i, double sample` **) ;**

Add a sample.

`double get( int i) ;`

Extract a sample.

`size( ) ;`

Return the size of the block of samples.

`Samples concat( Samples& samp1, Samples& samp2);`

Concatenate two blocks of samples.

`Samples operator+( Samples& samp);`

Mix two blocks of samples.

Name

DataRange – Implements a data range transformation function.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – DataTransform, SoundObj.

Description

DataRange allows you to map a data value into a range, returning the index of that range.

Available Methods

Public
- calculateValue – Sets value of Val result according to variable
- duplicateString (inherited) – Return a copy of string
- idNumber – Return the id number of this transform
- printConfiguration – Return a string starting with header that describes configuration
- printOn – Stream output function
- readConfiguration – Get params from input stream
- setValue – Set value of internal parameters

Included Files

"DataTransform.h"
"Mapfile.h"
"Val.h"

Public Methods

DataRange( );

The DataRange constructor

`void` **calculateValue(** `void *variable, void *result);`

Sets value of Val result according to variable

`virtual int` **idNumber**`( ) ;`

Return the id number of this transform

`Bool_` **readConfiguration**`( istream &is);`

Get params from input stream

`void` **setValue**`( char *ignored, char *filename);`

Set value of internal parameters

`char*` **printConfiguration**`( char *header);`

Return a string starting with header that describes configuration

`void` **printOn**`( ostream &os = cout);`

Stream output function

Name

SimpleSonicWidget – The most basic type of sonic widget.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – BaseSonicWidget, SCFParsing, SoundObj.

Description

SimpleSonicWidget is the most primitive type of Sonic Widget. A SimpleSonicWidget encapsulates a single Sound Control File, which specifies: (1) What host and device(port name) the file was created for. (2) The messages to be sent, including variable, fixed, and configurable parameters. The "execute" procedure performs all possible mappings to variable parameters and sends the messages specified in the file to the sound device, so that they are played.

This is the simplest kind of behavior.

Available Methods

Protected

- looksValid (inherited) – Check that a sound control file ends in .scf.

Public

- configure (inherited) – Called when Sonic Widget is configured if any special actions are required.
- duplicateString (inherited) – Return a copy of string
- eatComment (inherited) – Get a comment from the input stream
- execute – Execute the sonic widget with scalar or vector
- getHeader (inherited) – Get the header of a sound control file.
- getId (inherited) – Read an identifier from the input stream
- getName (inherited) – Return the name of the sonic widget.
- getSCF (inherited) – Return one of the constituant sound control files.
- getTok (inherited) – Get a token from the input stream
- getTransformFunction (inherited) – Return the transform function for this sonic widget.
- initialize – Initialize the sonic widget from a widget control file that has been positioned past the sonic widget class identifier.
- myClassName – Return the name of the widget class
- numSCFs (inherited) – Return the number of sound control files managed.
- peekTok (inherited) – Peek at a token without removing it from the stream
- printOn – Stream output function

- setBlocking (inherited) – Turn blocking on or off.
- setTransformFunction (inherited) – Set the transform function for this sonic widget.
- writeParamsResourceFile (inherited) – Write WCL resource file for SCF child, returning filename.
- writeSCF (inherited) – Write out a specific sound control file.
- writeWCF (inherited) – Write out widget control file.

Included Files

"BaseSonicWidget.h"
"SoundControlFile.h"

Public Methods

`virtual void` **execute**`( double d);`

Execute the sonic widget with scalar or vector

`virtual void` **execute**`( double *a, int length);`

Execute the sonic widget with scalar or vector

`virtual Bool_` **initialize**`( istream &is);`

Initialize the sonic widget from a widget control file that has been positioned past the sonic widget class identifier.

`virtual const char*` **myClassName**`( ) const ;`

Return the name of the widget class

`virtual void` **printOn**`( ostream &os = cout);`

Stream output function

Name

DataTransform – The base class for data transformations.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SoundObj.

Description

Base class for all data transformations.

Available Methods

Public

- calculateValue – Virtual set value of Val result
- duplicateString (inherited) – Return a copy of string
- idNumber – Return the id number of this transform
- printConfiguration – Return a string starting with header that describes configuration
- printOn – Stream output function
- readConfiguration – Get params from input stream
- setValue – Virtual set value of internal parameters

Included Files

"SoundObj.h"
"SCFParsing.h"

Public Methods

DataTransform( );

The DataTransform constructor.

~DataTransform( );

The DataTransform destructor.

`virtual void` **calculateValue**`( void *variable, void *result) ;`

Virtual set value of Val result

`virtual int` **idNumber**`( ) ;`

Return the id number of this transform

`virtual void` **setValue**`( char *args, double value) ;`

Virtual set value of internal parameters

`virtual void` **setValue**`( char *args, char *value) ;`

Virtual set value of internal parameters

`virtual Bool_` **readConfiguration**`( istream &is) ;`

Get params from input stream

`virtual char*` **printConfiguration**`( char *header);`

Return a string starting with header that describes configuration

`virtual void` **printOn**`( ostream &os = cout);`

Stream output function

Name

SmallWCLWidget – Implements a basic WCL widget.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SoundObj.

Description

SmallWCLWidget is the base class for WCL widgets.

Available Methods

Public
- create – Create the widget
- duplicateString (inherited) – Return a copy of string
- initialize – Initialize the widget
- printOn (inherited) – Stream output function

Included Files

<string.h>
"SoundObj.h"
<Xm/Xm.h>
<X11/Wc/WcCreate.h>
<X11/StringDefs.h>
<ctype.h>
<X11/Xmp/Table.h>
<X11/IntrinsicP.h>
<X11/CoreP.h>

Public Methods

SmallWCLWidget(`Widget parent, const char *name, const char *AppName);`

The SmallWCLWidget constructor.

~SmallWCLWidget( );

The SmallWCLWidget destructor.

`virtual void` **initialize( );**

Initialize the widget

`virtual void` **create( );**

Create the widget

Name

DataType – Implements the default (unknown) data type.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – Data, SoundObj.

Description

DataType is the default (unknown) data type, which can potentially be part of class Val.

Available Methods

Public

- binReceive – Receive binary representation of data
- binSend – Send binary representation of data
- char – Return DataType as char
- char* – Return DataType as character string
- double – Return DataType as double
- duplicateString (inherited) – Return a copy of string
- enumAsString – Return a particular enumerate as a string
- float – Return DataType as float
- getLowerBound – Returns the lower bound, if bounded
- getUpperBound – Returns the upper bound, if bounded
- int – Return DataType as int
- isBounded – Must this type be in a specific range?
- print – Return a string representation of DataType
- printOn – Stream output function
- recString (inherited) – Receive and allocate space for a string
- receive (inherited) – Receive data from fd
- send (inherited) – Write data to fd
- setBounds – Set upper and lower bounds
- setValueFromDouble – Set the value of DataType from double
- setValueFromString – Set data type value from string

Included Files

"Data.h"

Public Methods

DataType( ) ;

The DataType constructor.

~**DataType**( ) ;

The DataType destructor.

`virtual char*` **enumAsString**(`const int i)` ;

Return a particular enumerate as a string

`virtual void` **setValueFromString**(`const char *string )` ;

Set data type value from string

`virtual Bool_` **setBounds**(`const char l, const char u );`

Set upper and lower bounds

`virtual Bool_` **isBounded**( ) ;

Must this type be in a specific range?

`virtual char` **getLowerBound**( ) ;

Returns the lower bound, if bounded

`virtual char` **getUpperBound**( ) ;

Returns the upper bound, if bounded

`virtual void` **binSend**(`int sfd)` ;

Send binary representation of data

`virtual void` **binReceive**(`int sfd)` ;

Receive binary representation of data

`virtual char*` **print**( ) ;

Return a string representation of DataType

`virtual void` **setValueFromDouble**(`double d)` ;

Set the value of DataType from double

`virtual operator` **int**( ) ;

Return DataType as int

`virtual operator` **`char`**`( ) ;`

Return DataType as char

`virtual operator` **`float`**`( ) ;`

Return DataType as float

`virtual operator` **`double`**`( ) ;`

Return DataType as double

`virtual operator` **`char*`**`( ) ;`

Return DataType as character string

`virtual void` **`printOn`**`( ostream &os = cout);`

Stream output function

Name

Socket – Implements a UNIX socket.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – Data, SoundObj.

Description

Socket implements a UNIX socket.

Available Methods

Public

- binReceive (inherited) – Receive a binary representation of this object.
- binSend (inherited) – Send a binary representation of this object.
- duplicateString (inherited) – Return a copy of string
- getFd – Return the file descriptor for the socket
- getFp – Return the file descriptor as a file pointer
- getOption – Get socket options
- printOn (inherited) – Stream output function
- recString (inherited) – Receive and allocate space for a string
- receive – Receive a buffer
- send – Send a buffer
- setOption – Set socket options

Included Files

"SoundObj.h"
<stdio.h>
"Buffer.h"
"Data.h"

Public Methods

Socket( ) ;

The Socket constructor

~Socket( ) ;

The Socket destructor

int getFd( ) ;

Return the file descriptor for the socket

FILE* getFp(char *mode) ;

Return the file descriptor as a file pointer

virtual int receive(Buffer &);

Receive a buffer

virtual int receive(Buffer &, int n);

Receive a buffer

void receive(Data &v) ;

Receive a buffer

virtual int send(Buffer &, int n);

Send a buffer

void send(Data &v) ;

Send a buffer

virtual void getOption(int level, int name, void *val, int *len);

Get socket options

virtual void setOption(int level, int name, void *val, int len);

Set socket options

Name

DemoInterface – Implements the X Window System interface for the demonstration program.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SoundObj.

Description

DemoInterface is the class that represents the demo program interface.

Available Methods

Protected
- registerCallbacks – Register callback functions
- runLoop – Loop over data file, running sonic widget

Public
- create – Create the DemoInterface
- duplicateString (inherited) – Return a copy of string
- generalXSWIMemberCallback – General member callback
- mapToDurationCallback – Map data to Duration
- mapToPitchCallback – Map data to Pitch
- mapToTimbreCallback – Map data to Timbre
- mapToVolumeCallback – Map data to volume
- printOn – Stream output function
- registerMemberCallback – Register a member function as a callback
- run – Run the interface
- runCallback – Called when running
- stopCallback – Called when stopping
- triggerAlarmsCallback – Trigger alarms

Included Files

"SmallWCLWidget.h"
"FileReader.h"
"SonicWidgetCreator.h"
<Xm/Xm.h>
<ctype.h>

Protected Methods

`void` **registerCallbacks**`( );`

Register callback functions

`void` **runLoop**`( );`

Loop over data file, running sonic widget

Public Methods

DemoInterface`( int argc, char **argv);`

The DemoInterface constructor.

~DemoInterface`( );`

The DemoInterface destructor.

`virtual void` **create**`( );`

Create the DemoInterface

`virtual void` **runCallback**`( Widget w, void *, void *);`

Called when running

`virtual void` **stopCallback**`( Widget w, void *, void *);`

Called when stopping

`void` **mapToVolumeCallback**`( Widget w, void *ignored, void *unused);`

Map data to volume

`void` **mapToPitchCallback**`( Widget w, void *ignored, void *unused);`

Map data to Pitch

`void` **mapToTimbreCallback**`( Widget w, void *ignored, void *unused);`

Map data to Timbre

`void` **mapToDurationCallback**`( Widget w, void *ignored, void *unused);`

Map data to Duration

`void` **triggerAlarmsCallback**`( Widget w, void *ignored, void *unused);`

Trigger alarms

`virtual void` **registerMemberCallback(** `char *name, void *proc, DemoInterface *data);`

Register a member function as a callback

`static void` **generalXSWIMemberCallback(** `Widget w, void *rec, void *callData);`

General member callback

`void` **run(** `);`

Run the interface

`virtual void` **printOn(** `ostream &os = cout);`

Stream output function

Name

SonicWidgetCreator – Allows creation of a sonic widget from a text file.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SCFParsing, SoundObj.

Description

SonicWidgetCreator allows you to create a sonic widget from a widget control file.

Available Methods

Public

- createWidget – Create a new sonic widget.
- duplicateString (inherited) – Return a copy of string
- eatComment (inherited) – Get a comment from the input stream
- getId (inherited) – Read an identifier from the input stream
- getTok (inherited) – Get a token from the input stream
- looksValid – Check that widget controlFile ends with .wcf
- myClassName – Return the name of the widget class
- peekTok (inherited) – Peek at a token without removing it from the stream
- printOn – Stream output function

Included Files

"BaseSonicWidget.h"
"SimpleSonicWidget.h"
"BooleanSonicWidget.h"
"MultiSonicWidget.h"
"LoopingSonicWidget.h"
"InitializingSonicWidget.h"

Public Methods

SonicWidgetCreator( ) ;

The SonicWidgetCreator constructor.

~SonicWidgetCreator( ) ;

The SonicWidgetCreator destructor.

`BaseSonicWidget*` **createWidget(** `char *widgetControlFile);`

Create a new sonic widget.

`virtual const char*` **myClassName(** `) const ;`

Return the name of the widget class

`virtual Bool_` **looksValid(** `char *widgetControlFile);`

Check that widget controlFile ends with .wcf

`void` **printOn(** `ostream &os = cout);`

Stream output function

Name

DoubleType – Implements the basic data type for double precision floating point.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – DataType, Data, SoundObj.

Description

DoubleType is the double precision floating point type used by Val.

Available Methods

Public

- binReceive – Receive binary representation of data
- binSend – Send binary representation of data
- char – Return DoubleType as char
- char* – Return DoubleType as character string
- double (inherited) – Return DataType as double
- duplicateString (inherited) – Return a copy of string
- enumAsString (inherited) – Return a particular enumerate as a string
- float – Return DoubleType as float
- getLowerBound (inherited) – Returns the lower bound, if bounded
- getUpperBound (inherited) – Returns the upper bound, if bounded
- int – Return DoubleType as int
- isBounded (inherited) – Must this type be in a specific range?
- print – Return a string representation of DataType
- printOn – Stream output function
- recString (inherited) – Receive and allocate space for a string
- receive (inherited) – Receive data from fd
- send (inherited) – Write data to fd
- setBounds (inherited) – Set upper and lower bounds
- setValueFromDouble – Set the value of DataType from double
- setValueFromString – Set data type value from string

Included Files

"DataType.h"

Public Methods

DoubleType(`double dd): dd(dd) ;`

The DoubleType constructor.

DoubleType(`) ;`

The DoubleType constructor.

~DoubleType(`) ;`

The DoubleType destructor.

`virtual void` **setValueFromString(** `const char *string );`

Set data type value from string

`virtual void` **binSend(** `int sfd);`

Send binary representation of data

`virtual void` **binReceive(** `int sfd);`

Receive binary representation of data

`virtual char*` **print(** `);`

Return a string representation of DataType

`virtual void` **setValueFromDouble(** `double d) ;`

Set the value of DataType from double

`virtual operator` **int(** `) ;`

Return DoubleType as int

`virtual operator` **char(** `) ;`

Return DoubleType as char

`virtual operator` **float(** `) ;`

Return DoubleType as float

`virtual operator` **char*(** `) ;`

Return DoubleType as character string

`void` **printOn(** `ostream &os = cout);`

Stream output function

Name

SoundControlFile – Implements a sound control file.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SCFParsing, SoundObj.

Description

SoundControlFile implements a sound control file, a building block for sonic widgets.

Available Methods

Private

- configureTransformFunction – Configure transform functions from input stream
- initSoundDevice – Initialize sound device
- readMessages – Read messages from input stream
- registerVariableVal – Make note of variable Vals

Public

- dumpParamsResourceFile – Write out a resource file
- duplicateString (inherited) – Return a copy of string
- eatComment (inherited) – Get a comment from the input stream
- execute – Execute the SCF
- getDefault – Retrieve a default from a Val in the SCF
- getId (inherited) – Read an identifier from the input stream
- getTok (inherited) – Get a token from the input stream
- getTransformFunction – Get a TF from a Val in the SCF
- initialize – Initialize an SCF from an input stream or from a file
- peekTok (inherited) – Peek at a token without removing it from the stream
- printOn – Stream output function
- readSoundDevice – Read the sound device from an input stream
- setBlocking – Determine whether we do blocking or not
- setDefault – Set a default in the SCF
- setTransformFunction – Set the TF for a Val in the SCF
- writeSCF – Write the sound control file back to an output stream

Included Files

"WclParams.h"
"SCFParsing.h"
"SoundObj.h"
"ValList.h"
"MessageList.h"
"SoundDevice.h"
"TransformFunctions.h"

Private Methods

`Bool_` **initSoundDevice**`( istream &is, const char *hostname, const char *portname);`

Initialize sound device

`Bool_` **readMessages**`( istream &is);`

Read messages from input stream

`Bool_` **configureTransformFunction**`( Val &v, istream &is);`

Configure transform functions from input stream

`Bool_` **registerVariableVal**`( Val *v, int arrayIndex);`

Make note of variable Vals

Public Methods

SoundControlFile`( ) ;`

The SoundControlFile constructor.

~SoundControlFile`( );`

The SoundControlFile destructor.

`void` **setBlocking**`( Bool_ ack) ;`

Determine whether we do blocking or not

`Bool_` **initialize**`( istream &from, const char *hostname = "", const char *portname = "");`

Initialize an SCF from an input stream or from a file

`Bool_` **initialize**`( char *scf, const char *hostname = "", const char *portname = "");`

Initialize an SCF from an input stream or from a file

`Bool_` **readSoundDevice**`( istream &is, char *hostname, char *portname);`

Read the sound device from an input stream

`void` **writeSCF**`( ostream &os) const;`

Write the sound control file back to an output stream

`void` **execute**`( double d);`

Execute the SCF

`void` **execute**`( double *a, int length);`

Execute the SCF

`Bool_` **dumpParamsResourceFile**`( char *filename, char *widgetName);`

Write out a resource file

`void` **setDefault**`( int msgnum, int valnum, char *value);`

Set a default in the SCF

`void` **setDefault**`( int msgnum, int valnum, double d);`

Set a default in the SCF

`Val&` **getDefault**`( int msgnum, int valnum);`

Retrieve a default from a Val in the SCF

`DataTransform*` **getTransformFunction**`( int msgnum, int valnum);`

Get a TF from a Val in the SCF

`void` **setTransformFunction**`( int msgnum, int valnum, DataTransform *tf);`

Set the TF for a Val in the SCF

`void` **printOn**`( ostream &os = cout);`

Stream output function

Name

EnumType – Implements the default enumerated type.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – DataType, Data, SoundObj.

Description

EnumType is the default enumerated type for the Val class.

Available Methods

Public

- binReceive – Receive binary representation of data
- binSend – Send binary representation of data
- char – Return EnumType as char
- char* – Return EnumType as character string
- double – Return EnumType as double
- duplicateString (inherited) – Return a copy of string
- enumAsString – Return a particular enumarate as a string
- float – Return EnumType as float
- getLowerBound (inherited) – Returns the lower bound, if bounded
- getUpperBound (inherited) – Returns the upper bound, if bounded
- initFromFile – Initialize EnumType from file
- int – Return EnumType as int
- isBounded (inherited) – Must this type be in a specific range?
- operator= – Assignment operator
- print – Return a string representation of DataType
- printOn – Stream output function
- recString (inherited) – Receive and allocate space for a string
- receive (inherited) – Receive data from fd
- send (inherited) – Write data to fd
- setBounds (inherited) – Set upper and lower bounds
- setValueFromDouble (inherited) – Set the value of DataType from double
- setValueFromString (inherited) – Set data type value from string

Included Files

"DataType.h"
"PtrList.h"

Public Methods

EnumType(`char *filename)` ;

The EnumType constructor.

EnumType( ) ;

The EnumType constructor.

~**EnumType(**) ;

The EnumType destructor.

`EnumType&` **operator=(** `const EnumType& et )` ;

Assignment operator

`virtual void` **initFromFile(** `char *filename )`;

Initialize EnumType from file

`virtual char*` **enumAsString(** `const int i)` ;

Return a particular enumarate as a string

`virtual void` **binSend(** `int sfd)`;

Send binary representation of data

`virtual void` **binReceive(** `int sfd)`;

Receive binary representation of data

`virtual char*` **print(**);

Return a string representation of DataType

`void` **printOn(** `ostream &os = cout)`;

Stream output function

`virtual operator` **int(**) ;

Return EnumType as int

`virtual operator` **`char`**`( ) ;`

Return EnumType as char

`virtual operator` **`float`**`( ) ;`

Return EnumType as float

`virtual operator` **`double`**`( ) ;`

Return EnumType as double

`virtual operator` **`char*`**`( ) ;`

Return EnumType as character string

Name

SoundDevice – Implements a network sound device.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – Data, SoundObj.

Description

SoundDevice implements a network sound device.

Friends

Class UserInterface.
Class WclInterface.

Available Methods

Public

- binReceive (inherited) – Receive a binary representation of this object.
- binSend (inherited) – Send a binary representation of this object.
- connected – Indicate whether we are connected or not
- defaults – Return the device defaults
- duplicateString (inherited) – Return a copy of string
- executeMessage – Execute a sonic message, or a message list
- getFd – Return the fd for the socket connection
- messages – Return the messages
- name – Return the device name
- openConnection – Open a connection to the host/port
- printOn – Stream output function
- recString (inherited) – Receive and allocate space for a string
- receive – Receive data
- send (inherited) – Write data to fd
- setAcknowledge – Set flag to determine whether to block or not
- sysparams – Return the device parameters

Included Files

"SoundObj.h"
"TCPSocket.h"
"SoundServer.h"
"MessageList.h"
<fcntl.h>

Public Methods

SoundDevice(`char *RemoteHost, char *PortName, Bool_ ack = False_);`

The SoundDevice constructor.

SoundDevice(`Bool_ ack = False_) ;`

The SoundDevice constructor.

~SoundDevice(`);`

The SoundDevice destructor.

`void` **setAcknowledge(** `Bool_ ack) ;`

Set flag to determine whether to block or not

`void` **receive(** `Data &v) ;`

Receive data

`Bool_` **openConnection(** `char *RemoteHost, char *PortName);`

Open a connection to the host/port

`Bool_` **connected(** `) ;`

Indicate whether we are connected or not

`ValList*` **defaults(** `) ;`

Return the device defaults

`ValList*` **sysparams(** `) ;`

Return the device parameters

`MessageList*` **messages(** `) ;`

Return the messages

`char* name( ) ;`

Return the device name

`int getFd( ) ;`

Return the fd for the socket connection

`void executeMessage( MessageList &msglist);`

Execute a sonic message, or a message list

`void executeMessage( Message &msg);`

Execute a sonic message, or a message list

`void executeMessage( char *stringMessage);`

Execute a sonic message, or a message list

`void printOn( ostream &os = cout);`

Stream output function

Name

EnumVal – Implements an enumerate.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – IntType, DataType, Data, SoundObj.

Description

EnumVal is an enumerate.

Available Methods

Public

- binReceive (inherited) – Receive binary representation of data
- binSend (inherited) – Send binary representation of data
- char (inherited) – Return IntType as character
- char* (inherited) – Return IntType as character string
- double (inherited) – Return IntType as double
- duplicateString (inherited) – Return a copy of string
- enumAsString (inherited) – Return a particular enumerate as a string
- float (inherited) – Return IntType as float
- getLowerBound (inherited) – Returns the lower bound, if bounded
- getUpperBound (inherited) – Returns the upper bound, if bounded
- int (inherited) – Return IntType as integer
- isBounded (inherited) – Must this type be in a specific range?
- print (inherited) – Return a string representation of DataType
- printOn – Stream output function
- recString (inherited) – Receive and allocate space for a string
- receive (inherited) – Receive data from fd
- send (inherited) – Write data to fd
- setBounds (inherited) – Set upper and lower bounds
- setValueFromDouble – Set the value of EnumVal from double
- setValueFromString – Set data type value from string

Included Files

"IntType.h"

Public Methods

`virtual void` **setValueFromString(** `const char *string );`

Set data type value from string

`virtual void` **setValueFromDouble(** `double d);`

Set the value of EnumVal from double

`void` **printOn(** `ostream &os = cout);`

Stream output function

Name

SoundObj – Base class for all sound objects.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Description

SoundObj is the base class for all Sound Objects.

Available Methods

Public
- duplicateString – Return a copy of string
- printOn – Stream output function

Included Files

<stream.h>
"SoundDefines.h"

Public Methods

SoundObj( ) ;

The SoundObj constructor

virtual ~**SoundObj**( ) ;

The SoundObj destructor

char* **duplicateString**(const char *s) ;

Return a copy of string

virtual void **printOn**(ostream &os = cout) ;

Stream output function

Name

FileReader – A data file reader.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SoundObj.

Description

FileReader reads data values from a file, one per line. It may be used as a base class for more sophisticated file readers.

Available Methods

Public

- duplicateString (inherited) – Return a copy of string
- newFile – Read a new file
- nextValue – Return the next value from a file
- printOn – Stream output function
- rewindFile – Rewind the file to the beginning

Included Files

"SoundObj.h"

Public Methods

FileReader( );

The FileReader constructor.

~FileReader( );

The FileReader destructor.

`virtual Bool_` **newFile(** `const char *filename);`

Read a new file

`virtual Bool_` **nextValue**`( double &d);`

Return the next value from a file

`virtual Bool_` **nextValue**`( double *darray, int &length, const int maxsize);`

Return the next value from a file

`virtual void` **rewindFile**`( );`

Rewind the file to the beginning

`void` **printOn**`( ostream &os = cout);`

Stream output function

Name

SparcAudioHardware – Implements the interface to the SPARC audio chip.

Authors

Tom Birkett (birkett@cs.uiuc.edu)
Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SoundObj.

Description

A SparcAudioHardware defines the interface to the SPARC audio chip.

Available Methods

Public

- duplicateString (inherited) – Return a copy of string
- pausePlay – Pause play
- playFraction – "Say" the fractional part of some number
- playNote – At the next opportunity, play the requested note for the requested duration.
- playNumber – "Say" the given number
- playSamples – Play samples from sparc sample file
- printOn (inherited) – Stream output function
- reconfigDevice – Reconfigure device to match file header
- recordSamples – Record samples
- reset – Reset parameters to their initial values
- resumePlay – Resume play
- selectInstrument – Select one of the available instruments
- setOutputPort – Set the play port
- setRecordVolume – Set the record volume (attenuation)
- setVolume – Set the volume (attenuation) of the hardware.
- stopPlayingSamples – Flush queues and stop

Included Files

<sys/stat.h>
<sys/poll.h>
<sys/errno.h>

<sun/audioio.h>
<multimedia/libaudio.h>
"SoundObj.h"
<multimedia/audio_device.h>
"Oscil.h"
<fcntl.h>
<unistd.h>
<sys/types.h>

Public Methods

SparcAudioHardware( );

~SparcAudioHardware( );

The SparcAudioHardware destructor.

`Bool_` **setVolume(** `unsigned volumeRequested);`

Set the volume (attenuation) of the hardware.

`Bool_` **setRecordVolume(** `unsigned volumeRequested);`

Set the record volume (attenuation)

`Bool_` **reconfigDevice( );**

Reconfigure device to match file header

`Bool_` **selectInstrument(** `int number);`

Select one of the available instruments

`Bool_` **playSamples(** `char *filename, double duration = -1.);`

Play samples from sparc sample file

`Bool_` **playSamples(** `char *abuf, int buflen, int samplerate = 8192, double duration = -1);`

Play samples from sparc sample file

`Bool_` **stopPlayingSamples( );**

Flush queues and stop

`Bool_` **setOutputPort(** `unsigned port);`

Set the play port

`Bool_` **recordSamples(** `char *filename, double duration);`

Record samples

`Bool_` **recordSamples(** `char *buf, int numsamples);`

Record samples

`Bool_` **pausePlay(** `);`

Pause play

`Bool_` **resumePlay(** `);`

Resume play

`Bool_` **reset(** `);`

Reset parameters to their initial values

`Bool_` **playNote(** `char note, double duration );`

At the next opportunity, play the requested note for the requested duration.

`Bool_` **playNumber(** `int num, int voice);`

"Say" the given number

`Bool_` **playNumber(** `double num, int voice);`

"Say" the given number

`Bool_` **playFraction(** `char *fraction, int voice );`

"Say" the fractional part of some number

Name

FloatType – Implements the default floating point number type.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – DataType, Data, SoundObj.

Description

FloatType is used by Val to implement a floating point number.

Available Methods

Public

- binReceive – Receive binary representation of data
- binSend – Send binary representation of data
- char – Return FloatType as char
- char* – Return FloatType as a character string
- double (inherited) – Return DataType as double
- duplicateString (inherited) – Return a copy of string
- enumAsString (inherited) – Return a particular enumerate as a string
- float – Return FloatType as a float
- getLowerBound (inherited) – Returns the lower bound, if bounded
- getUpperBound (inherited) – Returns the upper bound, if bounded
- int – Return FloatType as int
- isBounded (inherited) – Must this type be in a specific range?
- print – Return a string representation of DataType
- printOn – Stream output function
- recString (inherited) – Receive and allocate space for a string
- receive (inherited) – Receive data from fd
- send (inherited) – Write data to fd
- setBounds (inherited) – Set upper and lower bounds
- setValueFromDouble – Set the value of DataType from double
- setValueFromString – Set data type value from string

Included Files

"DataType.h"

Public Methods

FloatType(float f): f(f) ;

The FloatType constructor.

FloatType( ) ;

The FloatType constructor.

~**FloatType**( ) ;

The FloatType destructor.

virtual void **setValueFromString**(const char *string);

Set data type value from string

virtual void **binSend**(int sfd);

Send binary representation of data

virtual void **binReceive**(int sfd);

Receive binary representation of data

virtual char* **print**( );

Return a string representation of DataType

virtual void **setValueFromDouble**(double d) ;

Set the value of DataType from double

virtual operator **int**( ) ;

Return FloatType as int

virtual operator **char**( ) ;

Return FloatType as char

virtual operator **float**( ) ;

Return FloatType as a float

virtual operator **char***( ) ;

Return FloatType as a character string

void **printOn**(ostream &os = cout);

Stream output function

Name

Step – Implements the step transformation function.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – DataTransform, SoundObj.

Description

The Step transform function is a combination of Linear and DataRange; a file specifying ranges is used with a user–configured step and offset to calculate the value from the input data.

Available Methods

Public

- calculateValue – Sets value of Val result according to variable
- duplicateString (inherited) – Return a copy of string
- idNumber – Return the id number of this transform
- printConfiguration – Return a string starting with header that describes configuration
- printOn – Stream output function
- readConfiguration – Get params from input stream
- setValue – Set value of internal parameters

Included Files

"DataTransform.h"
"Mapfile.h"
"Val.h"

Public Methods

Step( );

The Step constructor

void calculateValue(void *variable, void *result);

Sets value of Val result according to variable

`virtual int` **idNumber**`( ) ;`

Return the id number of this transform

`void` **setValue**`( char *args, double v);`

Set value of internal parameters

`void` **setValue**`( char *args, char *v);`

Set value of internal parameters

`Bool_` **readConfiguration**`( istream &is);`

Get params from input stream

`char*` **printConfiguration**`( char *header);`

Return a string starting with header that describes configuration

`void` **printOn**`( ostream &os = cout);`

Stream output function

Name

IndexList – Implements a list of indexes into defaults tables.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – PtrList, Data, SoundObj.

Description

IndexList is basically a list of integers (indexes into defaults tables).

Available Methods

Protected
- computeNewSize (inherited) – Determine new size for list resizing
- growList (inherited) – Enlarge list, based on index n
- resizeListTo (inherited) – Resize list to given size

Public
- addArg – Add element to list
- addElement (inherited) – Add pointer to end of list
- binReceive – Read binary representation of list from sfd
- binSend – Write binary representation of list to sfd
- clear – Delete all allocated integers from list
- deleteArg – Delete element from list
- deleteElement (inherited) – Delete element from list, returning it
- duplicateString (inherited) – Return a copy of string
- getArg – Select and return an element from list
- getElement (inherited) – Return element from list
- getIndex – Select and return an element from list
- operator= – Index list assignment
- printOn – Stream output function
- recString (inherited) – Receive and allocate space for a string
- receive (inherited) – Receive data from fd
- send (inherited) – Write data to fd
- setElement (inherited) – Assign to a specific element of the list

Included Files

<stdarg.h>
"PtrList.h"
"ValList.h"

Public Methods

IndexList(`int initialSize ...);`

The IndexList constructor.

IndexList(`int initialSize = 10) : (initialSize) ;`

The IndexList constructor.

~IndexList(`);`

Delete all allocated integers from list

`void` **clear(** `);`

Delete all allocated integers from list

`void` **binSend(** `int sfd);`

Write binary representation of list to sfd

`void` **binReceive(** `int sfd);`

Read binary representation of list from sfd

`int` **deleteArg(** `int n) ;`

Delete element from list

`int` **addArg(** `int* arg) ;`

Add element to list

`int` **getArg(** `int arg) const ;`

Select and return an element from list

`Val&` **getArg(** `ValList &table, int idx) ;`

Select and return an element from list

`int` **getIndex(** `int n) ;`

Select and return an element from list

`IndexList& operator=( const IndexList& il );`

Index list assignment

`void printOn( ostream &os = cout);`

Stream output function

Name

TCPSocket – Implements a TCP socket.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – Socket, Data, SoundObj.

Description

TCPSocket implements a TCP socket.

Available Methods

Public

- accept – Accept a connection
- binReceive (inherited) – Receive a binary representation of this object.
- binSend (inherited) – Send a binary representation of this object.
- bindToAnonymousPort – Bind to an anonymous port
- bindToNamedPort – Bind to a port name
- bindToPort – Bind to a port number
- connectToPort – Connect to port at host
- connectToService – Connect to service at host
- duplicateString (inherited) – Return a copy of string
- getFd (inherited) – Return the file descriptor for the socket
- getFp (inherited) – Return the file descriptor as a file pointer
- getOption (inherited) – Get socket options
- getPort – Return the port number
- listen – Configure network listener
- printOn – Stream output function
- recString (inherited) – Receive and allocate space for a string
- receive (inherited) – Receive a buffer
- send (inherited) – Send a buffer
- setOption (inherited) – Set socket options

Included Files

<sys/time.h>
<signal.h>

<errno.h>
<stdio.h>
<sys/types.h>
<netinet/in.h>
<sys/socket.h>
<sys/types.h>
<sys/resource.h>
<netdb.h>
"Socket.h"

Public Methods

TCPSocket(`int fd);`

The TCPSocket constructor

TCPSocket(`);`

The TCPSocket constructor

~TCPSocket(`);`

`u_short` **getPort(** `) ;`

Return the port number

`int` **bindToPort(** `u_short p);`

Bind to a port number

`int` **bindToNamedPort(** `char *service);`

Bind to a port name

`u_short` **bindToAnonymousPort(** `);`

Bind to an anonymous port

`void` **listen(** `int backlog);`

Configure network listener

`TCPSocket*` **accept(** `);`

Accept a connection

`int` **connectToPort(** `char *hostname, u_short port);`

Connect to port at host

`int` **connectToService(** `char *hostname, char *service);`

Connect to service at host

`void` **printOn(** `ostream &os);`

Stream output function

Name

InitializingSonicWidget – Implements the initializing sonic widget.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – BaseSonicWidget, SCFParsing, SoundObj.

Description

InitializingSonicWidget encapsulates two Sound Control Files. The first SCF is initialization information for the second, which does all the "work" when the widget receives data.

Available Methods

Protected
- looksValid (inherited) – Check that a sound control file ends in .scf.

Public
- configure – Called when Sonic Widget is configured if any special actions are required
- duplicateString (inherited) – Return a copy of string
- eatComment (inherited) – Get a comment from the input stream
- execute – Execute the sonic widget with scalar or vector
- getHeader (inherited) – Get the header of a sound control file.
- getId (inherited) – Read an identifier from the input stream
- getName (inherited) – Return the name of the sonic widget.
- getSCF (inherited) – Return one of the constituant sound control files.
- getTok (inherited) – Get a token from the input stream
- getTransformFunction (inherited) – Return the transform function for this sonic widget.
- initialize – Initialize the sonic widget from a widget control file that has been positioned past the sonic widget class identifier.
- myClassName – Return the name of the widget class
- numSCFs (inherited) – Return the number of sound control files managed.
- peekTok (inherited) – Peek at a token without removing it from the stream
- printOn – Stream output function
- setBlocking (inherited) – Turn blocking on or off.
- setTransformFunction (inherited) – Set the transform function for this sonic widget.
- writeParamsResourceFile – Write WCL resource file for SCF child, returning filename
- writeSCF (inherited) – Write out a specific sound control file.
- writeWCF (inherited) – Write out widget control file.

Included Files

"BaseSonicWidget.h"
"SoundControlFile.h"

Public Methods

`virtual void` **configure**`( ) ;`

Called when Sonic Widget is configured if any special actions are required

`virtual void` **execute**`( double d);`

Execute the sonic widget with scalar or vector

`virtual void` **execute**`( double *a, int length);`

Execute the sonic widget with scalar or vector

`virtual Bool_` **initialize**`( istream &is);`

Initialize the sonic widget from a widget control file that has been positioned past the sonic widget class identifier.

`virtual char*` **writeParamsResourceFile**`( char *directory, int which = 0 ) ;`

Write WCL resource file for SCF child, returning filename

`virtual const char*` **myClassName**`( ) const ;`

Return the name of the widget class

`virtual void` **printOn**`( ostream &os = cout);`

Stream output function

Name

TG33SoundDevice – Implements the interface to the Yamaha TG33 tone generator.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – MIDISoundDevice, SoundObj.

Description

TG33SoundDevice implements the interface to the Yamaha TG33 tone generator.

Available Methods

Public

- allNotesOff (inherited) – Turn all notes off
- closeDevice (inherited) – Close the MIDI device
- duplicateString (inherited) – Return a copy of string
- noteOff (inherited) – Execute a note off command
- noteOn (inherited) – Execute a note on command
- pause (inherited) – Cause device daemon to sleep before executing again
- playNote (inherited) – Play a note
- playNoteOnChannel – Play note on channel
- printOn (inherited) – Stream output function
- reinitialize (inherited) – Reinitialize the MIDI device
- resetAllControllers – Reinitialize device
- selectChannel (inherited) – Choose a channel
- selectInstrument – Select an instrument
- selectVoiceBank – Choose one of two voice banks
- setBalance – Set stereo balance
- setChannelPressure (inherited) – Set Channel pressure
- setKeyPressure (inherited) – Set key pressure
- setModulationWheel – Set modulation wheel
- setPitchBend (inherited) – Set pitch bend
- setPitchBendGranularity – Adjust pitch bend range.
- setPortamento (inherited) – Set Portamento
- setSustain (inherited) – Set sustain
- setVectorControlXaxis – Adjust vector in X-axis
- setVectorControlYaxis – Adjust vector in Y-axis

- setVibrato (inherited) – Set vibrato
- setVolume (inherited) – Set global volume

Included Files

"MIDISoundDevice.h"

Public Methods

TG33SoundDevice( );

The TG33SoundDevice constructor.

~TG33SoundDevice( );

The TG33SoundDevice destructor.

`void` **setPitchBendGranularity(** `ValList &);`

Adjust pitch bend range.

`void` **setModulationWheel(** `ValList &);`

Set modulation wheel

`void` **setVectorControlXaxis(** `ValList &);`

Adjust vector in X-axis

`void` **setVectorControlYaxis(** `ValList &);`

Adjust vector in Y-axis

`void` **playNoteOnChannel(** `ValList &v);`

Play note on channel

`void` **setBalance(** `ValList &v);`

Set stereo balance

`void` **selectInstrument(** `ValList &v);`

Select an instrument

`void` **selectVoiceBank(** `int vb);`

Choose one of two voice banks

`void` **selectVoiceBank(** `ValList &v);`

Choose one of two voice banks

`void` **resetAllControllers(** `);`

Reinitialize device

Name

IntType – Implements the basic integer Val type.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – DataType, Data, SoundObj.

Description

IntType is the basic type used by Val to represent integers.

Available Methods

Public

- binReceive – Receive binary representation of data
- binSend – Send binary representation of data
- char – Return IntType as character
- char* – Return IntType as character string
- double – Return IntType as double
- duplicateString (inherited) – Return a copy of string
- enumAsString (inherited) – Return a particular enumerate as a string
- float – Return IntType as float
- getLowerBound (inherited) – Returns the lower bound, if bounded
- getUpperBound (inherited) – Returns the upper bound, if bounded
- int – Return IntType as integer
- isBounded (inherited) – Must this type be in a specific range?
- print – Return a string representation of DataType
- printOn – Stream output function
- recString (inherited) – Receive and allocate space for a string
- receive (inherited) – Receive data from fd
- send (inherited) – Write data to fd
- setBounds (inherited) – Set upper and lower bounds
- setValueFromDouble – Set the value of DataType from double
- setValueFromString – Set data type value from string

Included Files

"DataType.h"

Public Methods

IntType(int i) : i(i) ;

The IntType constructor.

IntType( ) ;

The IntType constructor.

~**IntType**( ) ;

The IntType destructor.

virtual void **setValueFromString**(const char *string);

Set data type value from string

virtual void **binSend**(int sfd);

Send binary representation of data

virtual void **binReceive**(int sfd);

Receive binary representation of data

virtual char* **print**( );

Return a string representation of DataType

virtual void **setValueFromDouble**(double d);

Set the value of DataType from double

virtual operator **int**( ) ;

Return IntType as integer

virtual operator **char**( ) ;

Return IntType as character

virtual operator **float**( ) ;

Return IntType as float

virtual operator **double**( ) ;

Return IntType as double

virtual operator **char***( ) ;

Return IntType as character string

`void` **printOn(** `ostream &os = cout);`

Stream output function

Name

TransformFunctions – Manages transformation functions.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SCFParsing, SoundObj.

Description

TransformFunctions performs some management of the various transform functions.

Available Methods

Public

- SelectFunction – Translate a TF number to a TF instance
- duplicateString (inherited) – Return a copy of string
- eatComment (inherited) – Get a comment from the input stream
- functionList – Return the list of available transform functions
- getId (inherited) – Read an identifier from the input stream
- getTok (inherited) – Get a token from the input stream
- peekTok (inherited) – Peek at a token without removing it from the stream
- printOn – Stream output function
- readConfiguration – Read a TF configuration from a file

Included Files

"Step.h"
"BooleanMenu.h"
"SCFParsing.h"
"SoundObj.h"
"PtrList.h"
"DataTransform.h"
"PassThrough.h"
"Linear.h"
"DataRange.h"

Public Methods

TransformFunctions( );

The TransformFunctions constructor

~TransformFunctions( );

The TransformFunctions destructor

`DataTransform*` **readConfiguration(** `istream &is);`

Read a TF configuration from a file

`PtrList&` **functionList( ) ;**

Return the list of available transform functions

`DataTransform*` **SelectFunction(** `int func);`

Translate a TF number to a TF instance

`void` **printOn(** `ostream &os = cout);`

Stream output function

Name

Linear – Implements a linear transformation function.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – DataTransform, SoundObj.

Description

Linear performs the tranformation "y = a*x + b" where x is a scalar data value, a and b are constants set interactively, and y is the result.

Available Methods

Private
- compute – Used to do the actual computation.

Public
- calculateValue – Sets value of Val result according to variable
- duplicateString (inherited) – Return a copy of string
- idNumber – Return the id number of this transform
- printConfiguration – Return a string starting with header that describes configuration
- printOn – Stream output function
- readConfiguration – Get params from input stream
- setValue – Set value of internal parameters

Included Files

"DataTransform.h"
"Val.h"

Private Methods

```
double compute(   double x);
```

Used to do the actual computation.

Public Methods

`void` **calculateValue**`( void *variable, void *result);`

Sets value of Val result according to variable

`void` **setValue**`( char *args, double value);`

Set value of internal parameters

`void` **setValue**`( char *args, char *value);`

Set value of internal parameters

`virtual int` **idNumber**`( ) ;`

Return the id number of this transform

`Bool_` **readConfiguration**`( istream &is);`

Get params from input stream

`char*` **printConfiguration**`( char *header);`

Return a string starting with header that describes configuration

`void` **printOn**`( ostream &os = cout);`

Stream output function

Name

Val – Implements a parameter for a sonic message.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – Data, SoundObj.

Description

Val is the basic type for all values that describe sonic message parameters.

Available Methods

Public

- EnumAsString – Return the name of an enumerated type index
- GetTransformFunction – Return the transform function
- InitEnumFromFile – Initialize Val as an an enumerated type from a file
- InitFromString – Initialize Val from string, as in Default Parameters file
- SetTransformFunction – Set transform function for variable Vals
- SetValueFromString – Set the actual value of Val from an an ascii string
- binReceive (inherited) – Receive a binary representation of this object.
- binSend (inherited) – Send a binary representation of this object.
- duplicateString (inherited) – Return a copy of string
- index – Return index of Val
- kind – Return the kind of Val (configurable, fixed, variable)
- printOn (inherited) – Stream output function
- printVal – Return a char string with the value of Val
- recString (inherited) – Receive and allocate space for a string
- receive (inherited) – Receive data from fd
- retname – Return the name of the Val
- send (inherited) – Write data to fd
- stringToType – Determine type from its name
- type – Return the type of the Val

Included Files

"DoubleType.h"
"Data.h"

"BoolType.h"
"DataTransform.h"
"ArrayType.h"
"PtrList.h"
"SoundObj.h"
"RemoteFileType.h"
"DataType.h"
<ctype.h>
"CharType.h"
<sys/types.h>
"IntType.h"
<sys/stat.h>
"EnumVal.h"
<fcntl.h>
"EnumType.h"
<strings.h>
"FloatType.h"

Public Methods

Val( ) ;

The Val constructor

Val(`char c)` ;

The Val constructor

Val(`int i)` ;

The Val constructor

Val(`float f)` ;

The Val constructor

Val(`double d)` ;

The Val constructor

Val(`char* a)` ;

The Val constructor

~**Val(**) ;

The Val destructor

`ValType` **type**`( ) ;`
Return the type of the Val

`int` **kind**`( int i) ;`
Return the kind of Val (configurable, fixed, variable)

`int` **kind**`( ) ;`
Return the kind of Val (configurable, fixed, variable)

`int` **index**`( int i) ;`
Return index of Val

`int` **index**`( ) ;`
Return index of Val

`void` **InitFromString**`( char *str);`
Initialize Val from string, as in Default Parameters file

`void` **InitEnumFromFile**`( char *filename) ;`
Initialize Val as an an enumerated type from a file

`void` **SetValueFromString**`( char *s);`
Set the actual value of Val from an an ascii string

`void` **SetTransformFunction**`( DataTransform* dtf);`
Set transform function for variable Vals

`DataTransform*` **GetTransformFunction**`( ) const ;`
Return the transform function

`int` **stringToType**`( char *s);`
Determine type from its name

`char*` **retname**`( ) ;`
Return the name of the Val

`char*` **printVal**`( );`
Return a char string with the value of Val

`char*` **EnumAsString**`( const int i) const;`
Return the name of an enumerated type index

Name

LoopingSonicWidget – Implements the looping sonic widget.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – BooleanSonicWidget, BaseSonicWidget, SCFParsing, SoundObj.

Description

SonicLoopingWidget encapsulates a single Sound Control File and a transform function.

The "execute" procedure calls execute on the Sound Control File for the transformed data number of times.

It is similar to the BooleanSonicWidget because both encapsulate a single SCF and a single transform function.

Available Methods

Protected

- looksValid (inherited) – Check that a sound control file ends in .scf.

Public

- configure (inherited) – Called when Sonic Widget is configured if any special actions are required.
- duplicateString (inherited) – Return a copy of string
- eatComment (inherited) – Get a comment from the input stream
- execute – Execute the sonic widget with scalar or vector
- getHeader (inherited) – Get the header of a sound control file.
- getId (inherited) – Read an identifier from the input stream
- getName (inherited) – Return the name of the sonic widget.
- getSCF (inherited) – Return one of the constituant sound control files.
- getTok (inherited) – Get a token from the input stream
- getTransformFunction (inherited) – Return the transform function for this sonic widget.
- initialize (inherited) – Initialize the sonic widget from a widget control file that has been positioned past the sonic widget class identifier.
- myClassName – Return the name of the widget class
- numSCFs (inherited) – Return the number of sound control files managed.
- peekTok (inherited) – Peek at a token without removing it from the stream
- printOn – Stream output function

- setBlocking (inherited) – Turn blocking on or off.
- setTransformFunction (inherited) – Set the transform function for this sonic widget.
- writeParamsResourceFile (inherited) – Write WCL resource file for SCF child, returning filename.
- writeSCF (inherited) – Write out a specific sound control file.
- writeWCF (inherited) – Write out widget control file.

Included Files

"BooleanSonicWidget.h"

Public Methods

`virtual void` **`execute(`** `double d);`

Execute the sonic widget with scalar or vector

`virtual void` **`execute(`** `double *a, int length);`

Execute the sonic widget with scalar or vector

`virtual const char*` **`myClassName(`** `) const ;`

Return the name of the widget class

`virtual void` **`printOn(`** `ostream &os = cout);`

Stream output function

Name

ValList – Implements a list of Vals.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – PtrList, Data, SoundObj.

Description

ValList implements a list of Val objects.

Available Methods

Protected
- computeNewSize (inherited) – Determine new size for list resizing
- growList (inherited) – Enlarge list, based on index n
- resizeListTo (inherited) – Resize list to given size

Public
- addArg – Add an element to the list
- addElement (inherited) – Add pointer to end of list
- binReceive – Read a binary representation of ValList from sfd
- binReceiveValues – Read a binary representation of Val values in ValList from sfd
- binSend – Write a binary representation of ValList to sfd
- binSendValues – Write a binary representation of Val values in ValList to sfd
- clear – Deletes all Vals in a valList
- deleteArg – Delete an element of the list
- deleteElement (inherited) – Delete element from list, returning it
- duplicateString (inherited) – Return a copy of string
- getArg – Return an element of the list
- getElement (inherited) – Return element from list
- operator= – Val list assignment
- printOn – Stream output function
- recString (inherited) – Receive and allocate space for a string
- receive (inherited) – Receive data from fd
- send (inherited) – Write data to fd
- setArg – Assign to a specific element of the list
- setElement (inherited) – Assign to a specific element of the list

Included Files

"Val.h"
"PtrList.h"

Public Methods

ValList(`FILE *fp);`
The ValList constructor

ValList(`int initialSize = 10) : (initialSize) ;`
The ValList constructor

~ValList(`);`

`void` **printOn(** `ostream &os = cout);`
Stream output function

`void` **binSend(** `int sfd);`
Write a binary representation of ValList to sfd

`void` **binSendValues(** `int sfd);`
Write a binary representation of Val values in ValList to sfd

`void` **binSendValues(** `int sfd, double dd);`
Write a binary representation of Val values in ValList to sfd

`void` **binReceive(** `int sfd);`
Read a binary representation of ValList from sfd

`void` **binReceiveValues(** `int sfd);`
Read a binary representation of Val values in ValList from sfd

`Val&` **deleteArg(** `int n) ;`
Delete an element of the list

`Val&` **getArg(** `int n) const ;`
Return an element of the list

`int` **addArg(** `Val* arg) ;`
Add an element to the list

`int` **addArg(** `Val& arg)` ;

Add an element to the list

`int` **addArg(** `ValList& lookuptab, int idx)` ;

Add an element to the list

`void` **clear(**);

Deletes all Vals in a valList

`int` **setArg(** `int n, Val* arg)` ;

Assign to a specific element of the list

`ValList&` **operator=(** `ValList& vl)`;

Val list assignment

Name

MIDISoundDevice – The base interface class for MIDI instruments.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SoundObj.

Description

MIDISoundDevice is the base class for any MIDI sound devices. It supports the basic MIDI commands. This implementation supports a serial to MIDI interface; commands are sent by writing bytes to the serial port. Therefore, timing is inexact.

Available Methods

Public

- allNotesOff – Turn all notes off
- closeDevice – Close the MIDI device
- duplicateString (inherited) – Return a copy of string
- noteOff – Execute a note off command
- noteOn – Execute a note on command
- pause – Cause device daemon to sleep before executing again
- playNote – Play a note
- printOn – Stream output function
- reinitialize – Reinitialize the MIDI device
- selectChannel – Choose a channel
- selectInstrument – Choose an instrument
- setBalance – Set Balance
- setChannelPressure – Set Channel pressure
- setKeyPressure – Set key pressure
- setPitchBend – Set pitch bend
- setPortamento – Set Portamento
- setSustain – Set sustain
- setVibrato – Set vibrato
- setVolume – Set global volume

Included Files

"SoundObj.h"
"ValList.h"

Public Methods

MIDISoundDevice( );

The MIDISoundDevice constructor

~MIDISoundDevice( );

The MIDISoundDevice destructor

`virtual void` **reinitialize( );**

Reinitialize the MIDI device

`virtual void` **pause(** `ValList &);`

Cause device daemon to sleep before executing again

`virtual void` **playNote(** `ValList &);`

Play a note

`virtual void` **noteOn(** `ValList &);`

Execute a note on command

`virtual void` **noteOff(** `ValList &);`

Execute a note off command

`virtual void` **closeDevice( );**

Close the MIDI device

`virtual void` **selectInstrument(** `ValList &);`

Choose an instrument

`virtual void` **selectChannel(** `ValList &);`

Choose a channel

`virtual void` **setPitchBend(** `ValList &);`

Set pitch bend

`virtual void` **setKeyPressure**`( ValList &);`

Set key pressure

`virtual void` **setChannelPressure**`( ValList &);`

Set Channel pressure

`virtual void` **setPortamento**`( ValList &);`

Set Portamento

`virtual void` **setSustain**`( ValList &);`

Set sustain

`virtual void` **setVibrato**`( ValList &);`

Set vibrato

`virtual void` **setVolume**`( ValList &);`

Set global volume

`virtual void` **setBalance**`( ValList &);`

Set Balance

`virtual void` **allNotesOff**`( );`

Turn all notes off

`virtual void` **printOn**`( ostream &os = cout);`

Stream output function

Name

WaveTable – Stores a wavetable for sound generation functions.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SoundObj.

Description

WaveTable stores a wavetable which may be passed to sound generating functions.

Available Methods

Public

- duplicateString (inherited) – Return a copy of string
- get – Retrieve a value from the wavetable.
- printOn (inherited) – Stream output function
- put – Insert a value into the wavetable.
- samplerate – Return the samplerate of the wavetable.
- size – Return the size of the wavetable.

Included Files

<math.h>
"SoundObj.h"
"Samples.h"

Public Methods

WaveTable(`double samplerate, int tabsize=512);`

The WaveTable constructor.

WaveTable(`double samplerate, char *samplefile, int tabsize=512);`

The WaveTable constructor.

WaveTable(`double samplerate, Samples *mySamples);`

The WaveTable constructor.

~WaveTable(`) ;`

The WaveTable destructor.

`int` **size(** `) ;`

Return the size of the wavetable.

`double` **samplerate(** `) ;`

Return the samplerate of the wavetable.

put(`int i, double value) ;`

Insert a value into the wavetable.

`double` **get(** `int i) ;`

Retrieve a value from the wavetable.

Name

Mapfile – Provides a general mapping of data ranges to parameters.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SoundObj.

Description

Mapfile provides a general way to map data ranges to one or more varying parameters and neatly retrieve that information.

Available Methods

Public

- duplicateString (inherited) – Return a copy of string
- mapInt – Maps a double to an integer
- mapString – Maps a double to a string
- printOn (inherited) – Stream output function

Included Files

"SoundObj.h"

Public Methods

Mapfile(`char *mapfilename);`

The Mapfile constructor

~**Mapfile(** `);`

The Mapfile destructor

`Bool_` **mapString(** `double d, char* s);`

Maps a double to a string

`Bool_` **mapInt(** `double d, int ret);`

Maps a double to an integer

Name

WclInterface – Implements the X interface for the sdinterface application.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SoundObj.

Description

WclInterface is the X interface for the sdinterface application.

Available Methods

Private
- doButton – Do the resources for a button
- doEnumList – Do the resources for a scrolled list
- doSlider – Do the resources for a slider
- doText – Do the resources for a text widget
- dumpDefaultResources – Write out resources for defaults
- dumpMessageListResources – Write out resources for msg list
- dumpWclResources – Write out WCL resource file
- genName – Make up a temporary name
- onSignal – Clean up temp files on terminating signals

Public
- HighlightMessageArgs – Highlight the defaults that are message
- SendMessage – Send numbered message
- duplicateString (inherited) – Return a copy of string
- getDefault – Return numbered default
- getMessage – Return numbered message
- printOn – Stream output function
- run – Run the X interface

Included Files

"SoundObj.h"
"SoundDevice.h"
<Xm/Xm.h>
<ctype.h>

Private Methods

`void` **dumpWclResources(** `char *appClass, ostream &os);`

Write out WCL resource file

`char*` **dumpMessageListResources(** `ostream &os);`

Write out resources for msg list

`char*` **dumpDefaultResources(** `ostream &os);`

Write out resources for defaults

`char*` **genName(** `char *s);`

Make up a temporary name

`char*` **doButton(** `Val& v, ostream &os);`

Do the resources for a button

`char*` **doText(** `Val& v, ostream &os);`

Do the resources for a text widget

`char*` **doEnumList(** `Val& v, ostream &os);`

Do the resources for a scrolled list

`char*` **doSlider(** `Val& v, ostream &os);`

Do the resources for a slider

`static void` **onSignal(** `);`

Clean up temp files on terminating signals

Public Methods

WclInterface(`char *appName, SoundDevice &sd, int argc, char **argv);`

The WclInterface constructor.

~WclInterface(`);`

The WclInterface destructor.

`void` **run(** `);`

Run the X interface

`Val& getDefault( int defaultnum);`

Return numbered default

`Message& getMessage( int messagenum);`

Return numbered message

`void SendMessage( int msgnum = -1);`

Send numbered message

`void HighlightMessageArgs( );`

Highlight the defaults that are message

`void printOn( ostream &os = cout);`

Stream output function

Name

Message – Implements sonic messages.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – Data, SoundObj.

Description

A Message is the basic unit of communication between a local process and a remote sound device. It is a single command to the sound device.

Available Methods

Public

- addArgToValList – Add a Val to message ValList
- binReceive – Read a binary representation of Message from sfd
- binReceiveIndexList – Receive a binary representation of IndexList describing message arguments
- binSend – Write a binary representation of Message to sfd
- binSendIndexList – Send a binary representation of IndexList describing message arguments
- duplicateString (inherited) – Return a copy of string
- execute – Execute a message
- getIndexList – Return the IndexList describing message arguments
- getName – Return the message name
- getNumber – Retrieve the message number
- getValList – Return the ValList describing message arguments
- printOn – Stream output function
- recString (inherited) – Receive and allocate space for a string
- receive (inherited) – Receive data from fd
- send (inherited) – Write data to fd
- setNumber – Set the message number

Included Files

"ValList.h"
"IndexList.h"
"DeviceSpecificDefs.h"

Public Methods

Message(`char *msgname, IndexList &args, PMF mf);`

The Message constructor.

Message( ) ;

The Message constructor.

~**Message(**);

The Message destructor.

`IndexList&` **getIndexList(**) ;

Return the IndexList describing message arguments

`ValList&` **getValList(**) ;

Return the ValList describing message arguments

`char*` **getName(**) ;

Return the message name

`void` **binReceive(** `int sfd)` ;

Read a binary representation of Message from sfd

`void` **binSend(** `int sfd)` ;

Write a binary representation of Message to sfd

`void` **binReceiveIndexList(** `int sfd);`

Receive a binary representation of IndexList describing message arguments

`void` **binSendIndexList(** `int sfd);`

Send a binary representation of IndexList describing message arguments

`void` **addArgToValList(** `Val &v)` ;

Add a Val to message ValList

`void` **execute(**);

Execute a message

`void` **setNumber(** `int num)` ;

Set the message number

```
int getNumber(    ) ;
```

Retrieve the message number

```
void printOn(    ostream &os = cout);
```

Stream output function

Name

WclParams – Generates X resource files for WCL for automatic sonic widget configuration.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SoundObj.

Description

WclParams generates X resource files for WCL that describe the configuration options for Sonic Widgets.

Available Methods

Protected
- *genName – Please describe this method.
- doButton – Write out WCL code for a button
- doLabelWidget – Write out WCL code for a label widget
- doPushButtonWidget – Write out code for a push button
- doText – Write out WCL code for a text widget
- dumpMessage – Write out WCL code for a sonic message
- outputPtrList – Write out a list of strings

Public
- doEnumList – Write out WCL code to configure an enumerated list
- doSlider – Write out WCL code to generate a scale widget
- doVariableVal – Write out WCL code for a variable Val
- dumpMessageList – Write out WCL code for a sonic message list
- dumpVal – Write out WCL code for a Val
- dumpWclResourceFile – Write out a WCL resource file for a target message list
- duplicateString (inherited) – Return a copy of string
- printOn – Stream output function

Included Files

"SoundObj.h"
"SoundDevice.h"
"TransformFunctions.h"

Protected Methods

`char` ***genName(**);

`void` **outputPtrList(** `ostream &os, PtrList &pl);`

Write out a list of strings

`char*` **doLabelWidget(** `ostream &os, char *text);`

Write out WCL code for a label widget

`char*` **doPushButtonWidget(** `ostream &os, char *label);`

Write out code for a push button

`char*` **dumpMessage(** `ostream &os, Message &msg);`

Write out WCL code for a sonic message

`char*` **doButton(** `ostream &os, Val& v);`

Write out WCL code for a button

`char*` **doText(** `ostream &os, Val& v);`

Write out WCL code for a text widget

Public Methods

WclParams(`char *widgetName, ValList* Defaults, ValList* SystemParameters);`

The WclParams constructor.

~WclParams( );

The WclParams destructor.

`Bool_` **dumpWclResourceFile(** `ostream &os, MessageList& targetMessageList);`

Write out a WCL resource file for a target message list

`void` **dumpMessageList(** `ostream &os, MessageList& targetMessageList);`

Write out WCL code for a sonic message list

`char*` **dumpVal(** `ostream &os, Val& v);`

Write out WCL code for a Val

`char* doVariableVal( ostream &os, Val& v);`

Write out WCL code for a variable Val

`char* doEnumList( ostream &os, Val& v);`

Write out WCL code to configure an enumerated list

`char* doEnumList( ostream &os, PtrList &enumlist, char *enumname, int selected);`

Write out WCL code to generate a scale widget

`char* doSlider( ostream &os, Val& v);`

Write out WCL code to generate a scale widget

`void printOn( ostream &os = cout);`

Stream output function

Name

MessageList – Implements a list of messages.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – PtrList, Data, SoundObj.

Description

MessageList implements a list of messages.

Available Methods

Protected
- computeNewSize (inherited) – Determine new size for list resizing
- growList (inherited) – Enlarge list, based on index n
- resizeListTo (inherited) – Resize list to given size

Public
- addArg – Add a message to the list
- addElement (inherited) – Add pointer to end of list
- binReceive – Read a binary representation of Message from sfd
- binSend – Write a binary representation of Message to sfd
- clear – Delete all the messages in the list
- deleteElement (inherited) – Delete element from list, returning it
- duplicateString (inherited) – Return a copy of string
- getArg – Return a message from the list
- getElement (inherited) – Return element from list
- printOn – Stream output function
- recString (inherited) – Receive and allocate space for a string
- receive (inherited) – Receive data from fd
- send (inherited) – Write data to fd
- setElement (inherited) – Assign to a specific element of the list

Included Files

"PtrList.h"
"Message.h"

Public Methods

MessageList(`int initialSize = 10) : (initialSize) ;`

The MessageList constructor.

`int` **addArg(** `Message* msg) ;`

Add a message to the list

`Message&` **getArg(** `int n) const ;`

Return a message from the list

`void` **binSend(** `int sfd);`

Write a binary representation of Message to sfd

`void` **binReceive(** `int sfd);`

Read a binary representation of Message from sfd

`void` **clear(** `);`

Delete all the messages in the list

`void` **printOn(** `ostream &os = cout);`

Stream output function

Name

XSonicWidgetInterface – The base class for X interfaces to manipulate sonic widgets.

Authors

Tara Madhyastha (tara@cs.uiuc.edu)

Heritage

Inherits from – SoundObj.

Description

XSonicWidgetInterface is the base class for X interfaces to manipulate sound control files. Primarily useful for callback registering routines. There is some MAJOR hackery involved in doing callbacks in C++. Beware!

Available Methods

Protected
- registerCallbacks – Register callback functions

Public
- configureCallback – Called to configure the Sonic Widget
- create – Create the XSonicWidgetInterface
- duplicateString (inherited) – Return a copy of string
- generalXSWIMemberCallback – General member callback
- printOn – Stream output function
- quitCallback – Called when quitting
- reRunCallback – Called to rewind the filereader
- registerMemberCallback – Register a member function as a callback
- run – Run the interface
- runCallback – Called when running
- selectNewDataFileCallback – Choose a new data file
- stepCallback – Called to step through one data value
- stopCallback – Called when stopping
- writeWCFCallback – Write an SCF

Included Files

<ctype.h>
"SWMultiParams.h"

"FileReader.h"
<Xm/Xm.h>

Protected Methods

`void` **registerCallbacks**`( );`

Register callback functions

Public Methods

XSonicWidgetInterface`( BaseSonicWidget *SW, FileReader *filereader, int argc, char **argv);`

The XSonicWidgetInterface constructor.

~**XSonicWidgetInterface**`( );`

The XSonicWidgetInterface destructor.

`virtual void` **create**`( );`

Create the XSonicWidgetInterface

`virtual void` **selectNewDataFileCallback**`( Widget w, void *, void *);`

Choose a new data file

`virtual void` **writeWCFCallback**`( Widget w, void *, void *);`

Write an SCF

`virtual void` **quitCallback**`( Widget w, void *, void *);`

Called when quitting

`virtual void` **runCallback**`( Widget w, void *, void *);`

Called when running

`virtual void` **stopCallback**`( Widget w, void *, void *);`

Called when stopping

`virtual void` **reRunCallback**`( Widget w, void *, void *);`

Called to rewind the filereader

`virtual void` **stepCallback**`( Widget w, void *, void *);`

Called to step through one data value

`virtual void` **configureCallback**`( Widget w, void *, void *);`

Called to configure the Sonic Widget

`virtual void` **registerMemberCallback**`( char *name, void *proc, XSonicWidgetInterface *data);`

Register a member function as a callback

`static void` **generalXSWIMemberCallback**`( Widget w, void *rec, void *callData);`

General member callback

`void` **run**`( );`

Run the interface

`virtual void` **printOn**`( ostream &os = cout);`

Stream output function

Pablo Class Hierarchy

SoundObj 71
- BaseSoundDevice 11
- Buffer 26
- Data 36
  - DataType 48
    - ArrayType 1
    - CharType 30
    - DoubleType 58
    - EnumType 63
    - FloatType 77
    - IntType 92
      - BoolType 14
      - EnumVal 69
        - RemoteFileType 20
  - Message 117
  - PtrList 16
    - IndexList 81
    - MessageList 123
    - ValList 104
  - Socket 51
    - TCPSocket 84
  - SoundDevice 66
  - Val 99
- DataTransform 44
  - BooleanMenu 18
  - DataRange 40
  - Linear 97
  - PassThrough 12
  - Step 79
- DemoInterface 53
- FileReader 72
- MIDISoundDevice 107
  - TG33SoundDevice 89
- Mapfile 112
- Oscil 9
- SCFParsing 24
  - BaseSonicWidget 6
    - BooleanSonicWidget 22
      - LoopingSonicWidget 102
    - InitializingSonicWidget 87
    - MultiSonicWidget 4
    - SimpleSonicWidget 42

SonicWidgetCreator 56
SoundControlFile 60
TransformFunctions 95
SPARCSoundDevice 28
Samples 38
SmallWCLWidget 46
SWMultiParams 32
SparcAudioHardware 74
WaveTable 110
WclInterface 114
WclParams 120
XSonicWidgetInterface 125

Index to Classes and their Methods

ArrayType, 1
- ArrayType, 2
- ~ArrayType, 2
- binReceive, 2
- binSend, 2
- char, 2
- char*, 3
- double, 3
- float, 2
- int, 2
- operator=, 2
- print, 2
- printOn, 3
- setValueFromDouble, 2
- setValueFromString, 2

BaseSonicWidget, 6
- &operator<<, 8
- BaseSonicWidget, 7
- ~BaseSonicWidget, 7
- configure, 7
- execute, 7
- getHeader, 7
- getName, 7
- getSCF, 7
- getTransformFunction, 8
- initialize, 8
- looksValid, 7
- myClassName, 7
- numSCFs, 7
- printOn, 8
- setBlocking, 8
- setTransformFunction, 8
- writeParamsResourceFile, 8
- writeSCF, 8
- writeWCF, 8

BaseSoundDevice, 11
- BaseSoundDevice, 11
- ~BaseSoundDevice, 11
- printOn, 11

BooleanMenu, 18
- calculateValue, 19
- compute, 18
- idNumber, 19
- printConfiguration, 19
- printOn, 19
- readConfiguration, 19
- setOperation, 18
- setValue, 19

BooleanSonicWidget, 22
- execute, 23
- initialize, 23
- myClassName, 23
- printOn, 23

BoolType, 14
- print, 15
- printOn, 15
- setBounds, 15
- setValueFromString, 15

Buffer, 26
- Buffer, 26
- ~Buffer, 26
- getData, 26
- getLength, 26
- setData, 27

CharType, 30
- binReceive, 31
- binSend, 31
- char, 31
- char*, 31
- CharType, 31
- ~CharType, 31
- float, 31
- int, 31
- print, 31

printOn, 31
setValueFromDouble, 31
setValueFromString, 31

Data, 36
binReceive, 37
binSend, 37
Data, 36
~Data, 36
printOn, 37
recString, 37
send, 36, 37

DataRange, 40
calculateValue, 40
DataRange, 40
idNumber, 40
printConfiguration, 41
printOn, 41
readConfiguration, 41
setValue, 41

DataTransform, 44
calculateValue, 44
DataTransform, 44
~DataTransform, 44
idNumber, 45
printConfiguration, 45
printOn, 45
readConfiguration, 45
setValue, 45

DataType, 48
binReceive, 49
binSend, 49
char, 49
char*, 50
DataType, 49
~DataType, 49
double, 50
enumAsString, 49
float, 50
getLowerBound, 49
getUpperBound, 49
int, 49
isBounded, 49
print, 49
printOn, 50
setBounds, 49
setValueFromDouble, 49
setValueFromString, 49

DemoInterface, 53
create, 54
DemoInterface, 54
~DemoInterface, 54
generalXSWIMemberCallback, 55
mapToDurationCallback, 54
mapToPitchCallback, 54
mapToTimbreCallback, 54
mapToVolumeCallback, 54
printOn, 55
registerCallbacks, 54
registerMemberCallback, 54
run, 55
runCallback, 54
runLoop, 54
stopCallback, 54
triggerAlarmsCallback, 54

DoubleType, 58
binReceive, 59
binSend, 59
char, 59
char*, 59
DoubleType, 59
~DoubleType, 59
float, 59
int, 59
print, 59
printOn, 59
setValueFromDouble, 59
setValueFromString, 59

EnumType, 63
binReceive, 64
binSend, 64

char, 64
char*, 65
double, 65
enumAsString, 64
EnumType, 64
~EnumType, 64
float, 65
initFromFile, 64
int, 64
operator=, 64
print, 64
printOn, 64

EnumVal, 69
printOn, 70
setValueFromDouble, 70
setValueFromString, 70

FileReader, 72
FileReader, 72
~FileReader, 72
newFile, 72
nextValue, 72, 73
printOn, 73
rewindFile, 73

FloatType, 77
binReceive, 78
binSend, 78
char, 78
char*, 78
float, 78
FloatType, 78
~FloatType, 78
int, 78
print, 78
printOn, 78
setValueFromDouble, 78
setValueFromString, 78

IndexList, 81
addArg, 82
binReceive, 82
binSend, 82
clear, 82
deleteArg, 82
getArg, 82
getIndex, 82
IndexList, 82
~IndexList, 82
operator=, 82
printOn, 83

InitializingSonicWidget, 87
configure, 88
execute, 88
initialize, 88
myClassName, 88
printOn, 88
writeParamsResourceFile, 88

IntType, 92
binReceive, 93
binSend, 93
char, 93
char*, 93
double, 93
float, 93
int, 93
IntType, 93
~IntType, 93
print, 93
printOn, 93
setValueFromDouble, 93
setValueFromString, 93

Linear, 97
calculateValue, 98
compute, 97
idNumber, 98
printConfiguration, 98
printOn, 98
readConfiguration, 98
setValue, 98

LoopingSonicWidget, 102
execute, 103

myClassName, 103
printOn, 103

Mapfile, 112
Mapfile, 112
~Mapfile, 112
mapInt, 112
mapString, 112

Message, 117
addArgToValList, 118
binReceive, 118
binReceiveIndexList, 118
binSend, 118
binSendIndexList, 118
execute, 118
getIndexList, 118
getName, 118
getNumber, 118
getValList, 118
Message, 118
~Message, 118
printOn, 119
setNumber, 118

MessageList, 123
addArg, 124
binReceive, 124
binSend, 124
clear, 124
getArg, 124
MessageList, 124
printOn, 124

MIDISoundDevice, 107
allNotesOff, 109
closeDevice, 108
MIDISoundDevice, 108
~MIDISoundDevice, 108
noteOff, 108
noteOn, 108
pause, 108
playNote, 108
printOn, 109
reinitialize, 108
selectChannel, 108
selectInstrument, 108
setBalance, 109
setChannelPressure, 109
setKeyPressure, 108
setPitchBend, 108
setPortamento, 109
setSustain, 109
setVibrato, 109
setVolume, 109

MultiSonicWidget, 4
execute, 5
initialize, 5
myClassName, 5
printOn, 5

Oscil, 9
bsamples, 9, 10
Oscil, 9
~Oscil, 9

PassThrough, 12
calculateValue, 12
idNumber, 12
printConfiguration, 13
printOn, 13
readConfiguration, 12

PtrList, 16
addElement, 17
computeNewSize, 17
deleteElement, 17
getElement, 17
growList, 17
numberArgs, 17
PtrList, 17
~PtrList, 17
resizeListTo, 17
setElement, 17

RemoteFileType, 20

printOn, 21

Samples, 38
concat, 39
get, 38
operator+, 39
put, 38
Samples, 38
~Samples, 38
size, 39

SCFParsing, 24
eatComment, 24
getId, 25
getTok, 24, 25
peekTok, 25
printOn, 25
SCFParsing, 24
~SCFParsing, 24

SimpleSonicWidget, 42
execute, 43
initialize, 43
myClassName, 43
printOn, 43

SmallWCLWidget, 46
create, 47
initialize, 47
run, 47
SmallWCLWidget, 46
~SmallWCLWidget, 46

Socket, 51
getFd, 52
getFp, 52
getOption, 52
receive, 52
send, 52
setOption, 52
Socket, 51
~Socket, 51

SonicWidgetCreator, 56
createWidget, 57
looksValid, 57
myClassName, 57
printOn, 57
SonicWidgetCreator, 56
~SonicWidgetCreator, 56

SoundControlFile, 60
configureTransformFunction, 61
dumpParamsResourceFile, 62
execute, 62
getDefault, 62
getTransformFunction, 62
initialize, 61
initSoundDevice, 61
printOn, 62
readMessages, 61
readSoundDevice, 61
registerVariableVal, 61
setBlocking, 61
setDefault, 62
setTransformFunction, 62
SoundControlFile, 61
~SoundControlFile, 61
writeSCF, 62

SoundDevice, 66
connected, 67
defaults, 67
executeMessage, 68
getFd, 68
messages, 67
name, 67
openConnection, 67
printOn, 68
receive, 67
setAcknowledge, 67
SoundDevice, 67
~SoundDevice, 67
sysparams, 67

SoundObj, 71
duplicateString, 71
printOn, 71

SoundObj, 71
~SoundObj, 71

SparcAudioHardware, 74
pausePlay, 76
playFraction, 76
playNote, 76
playNumber, 76
playSamples, 75
reconfigDevice, 75
recordSamples, 75, 76
reset, 76
resumePlay, 76
selectInstrument, 75
setOutputPort, 75
setRecordVolume, 75
setVolume, 75
SparcAudioHardware, 75
~SparcAudioHardware, 75
stopPlayingSamples, 75

SPARCSoundDevice, 28
playNote, 29
playNumber, 29
playSampleFile, 29
printOn, 29
reinitialize, 29
selectInstrument, 29
setOutputPort, 29
setRecordVolume, 29
setVolume, 29
SPARCSoundDevice, 28
~SPARCSoundDevice, 28

Step, 79
calculateValue, 79
idNumber, 79
printConfiguration, 80
printOn, 80
readConfiguration, 80
setValue, 80
Step, 79

SWMultiParams, 32
cancelledCallback, 34
centerWidget, 33
configureTransformCallback, 34
create, 33
dismissedCallback, 34
doneRunningCallback, 34
fixupSize, 34
generalSWParamsCallback, 35
getCallbackArgs, 34
getTransformFunction, 34
printOn, 35
promptTransformFunctionMenu, 34
PutLineResourceCB, 34
registerCallbacks, 33
registerMemberCallback, 35
resetAllToggles, 33
revertTransformFunctionCallback, 34
run, 34
selectTransformFunctionCallback, 34
SWMultiParams, 33
~SWMultiParams, 33
updateDefaultCallback, 35
updateDefaultFromListCallback, 35
updateDefaultFromSliderCallback, 35
updateDefaultFromTextCallback, 35
updateDefaultFromToggleCallback, 35
updateRevertLabel, 34
updateTransformDefaultCallback, 35
writeResourceFiles, 33

TCPSocket, 84
accept, 85
bindToAnonymousPort, 85
bindToNamedPort, 85
bindToPort, 85
connectToPort, 85
connectToService, 85
getPort, 85
listen, 85
printOn, 86
TCPSocket, 85
~TCPSocket, 85

TG33SoundDevice, 89

playNoteOnChannel, 90
reinitialize, 91
resetAllControllers, 91
selectInstrument, 90
selectVoiceBank, 90
setBalance, 90
setModulationWheel, 90
setPitchBendGranularity, 90
setVectorControlXaxis, 90
setVectorControlYaxis, 90
TG33SoundDevice, 90
~TG33SoundDevice, 90

TransformFunctions, 95
functionList, 96
printOn, 96
readConfiguration, 96
SelectFunction, 96
TransformFunctions, 96
~TransformFunctions, 96

Val, 99
EnumAsString, 101
GetTransformFunction, 101
index, 101
InitEnumFromFile, 101
InitFromString, 101
kind, 101
name, 101
printVal, 101
retname, 101
SetTransformFunction, 101
SetValueFromString, 101
stringToType, 101
type, 100
Val, 100
~Val, 100

ValList, 104
addArg, 105, 106
binReceive, 105
binReceiveValues, 105
binSend, 105
binSendValues, 105
clear, 106
deleteArg, 105
getArg, 105
operator=, 106
printOn, 105
setArg, 106
ValList, 105
~ValList, 105

WaveTable, 110
get, 111
put, 111
samplerate, 111
size, 111
WaveTable, 110
~WaveTable, 111

WclInterface, 114
doButton, 115
doEnumList, 115
doSlider, 115
doText, 115
dumpDefaultResources, 115
dumpMessageListResources, 115
dumpWclResources, 115
genName, 115
getDefault, 115
getMessage, 116
HighlightMessageArgs, 116
onSignal, 115
printOn, 116
run, 115
SendMessage, 116
WclInterface, 115
~WclInterface, 115

WclParams, 120
*genName, 121
doButton, 121
doEnumList, 122
doLabelWidget, 121
doPushButtonWidget, 121

doSlider, 122
doText, 121
doVariableVal, 121
dumpMessage, 121
dumpMessageList, 121
dumpVal, 121
dumpWclResourceFile, 121
outputPtrList, 121
printOn, 122
WclParams, 121
~WclParams, 121

XSonicWidgetInterface, 125
configureCallback, 126
create, 126
generalXSWIMemberCallback, 127
printOn, 127
quitCallback, 126
registerCallbacks, 126
registerMemberCallback, 127
reRunCallback, 126
run, 127
runCallback, 126
selectNewDataFileCallback, 126
stepCallback, 126
stopCallback, 126
writeWCFCallback, 126
XSonicWidgetInterface, 126
~XSonicWidgetInterface, 126

Index to the Methods

*genName
    WclParams, 121
&operator<<
    BaseSonicWidget, 8

accept
    TCPSocket, 85
addArg
    IndexList, 82
    MessageList, 124
    ValList, 105, 106
addArgToValList
    Message, 118
addElement
    PtrList, 17
allNotesOff
    MIDISoundDevice, 109
ArrayType
    ArrayType, 2
~ArrayType
    ArrayType, 2

BaseSonicWidget
    BaseSonicWidget, 7
~BaseSonicWidget
    BaseSonicWidget, 7
BaseSoundDevice
    BaseSoundDevice, 11
~BaseSoundDevice
    BaseSoundDevice, 11
bindToAnonymousPort
    TCPSocket, 85
bindToNamedPort
    TCPSocket, 85
bindToPort
    TCPSocket, 85
binReceive
    ArrayType, 2
    CharType, 31
    Data, 37
    DataType, 49
    DoubleType, 59
    EnumType, 64
    FloatType, 78
    IndexList, 82
    IntType, 93
    Message, 118
    MessageList, 124
    ValList, 105
binReceiveIndexList
    Message, 118
binReceiveValues
    ValList, 105
binSend
    ArrayType, 2
    CharType, 31
    Data, 37
    DataType, 49
    DoubleType, 59
    EnumType, 64
    FloatType, 78
    IndexList, 82
    IntType, 93
    Message, 118
    MessageList, 124
    ValList, 105
binSendIndexList
    Message, 118
binSendValues
    ValList, 105
bsamples
    Oscil, 9, 10
Buffer
    Buffer, 26
~Buffer
    Buffer, 26

calculateValue
    BooleanMenu, 19
    DataRange, 40
    DataTransform, 44
    Linear, 98
    PassThrough, 12
    Step, 79
cancelledCallback

SWMultiParams, 34
centerWidget
SWMultiParams, 33
char
ArrayType, 2
CharType, 31
DataType, 49
DoubleType, 59
EnumType, 64
FloatType, 78
IntType, 93
char*
ArrayType, 3
CharType, 31
DataType, 50
DoubleType, 59
EnumType, 65
FloatType, 78
IntType, 93
CharType
CharType, 31
~CharType
CharType, 31
clear
IndexList, 82
MessageList, 124
ValList, 106
closeDevice
MIDISoundDevice, 108
compute
BooleanMenu, 18
Linear, 97
computeNewSize
PtrList, 17
concat
Samples, 39
configure
BaseSonicWidget, 7
InitializingSonicWidget, 88
configureCallback
XSonicWidgetInterface, 126
configureTransformCallback
SWMultiParams, 34
configureTransformFunction
SoundControlFile, 61
connected
SoundDevice, 67
connectToPort
TCPSocket, 85
connectToService
TCPSocket, 85
create
DemoInterface, 54
SmallWCLWidget, 47
SWMultiParams, 33
XSonicWidgetInterface, 126
createWidget
SonicWidgetCreator, 57

Data
Data, 36
~Data
Data, 36
DataRange
DataRange, 40
DataTransform
DataTransform, 44
~DataTransform
DataTransform, 44
DataType
DataType, 49
~DataType
DataType, 49
defaults
SoundDevice, 67
deleteArg
IndexList, 82
ValList, 105
deleteElement
PtrList, 17
DemoInterface
DemoInterface, 54
~DemoInterface
DemoInterface, 54
dismissedCallback
SWMultiParams, 34
doButton
WclInterface, 115

WclParams, 121
doEnumList
WclInterface, 115
WclParams, 122
doLabelWidget
WclParams, 121
doneRunningCallback
SWMultiParams, 34
doPushButtonWidget
WclParams, 121
doSlider
WclInterface, 115
WclParams, 122
doText
WclInterface, 115
WclParams, 121
double
ArrayType, 3
DataType, 50
EnumType, 65
IntType, 93
DoubleType
DoubleType, 59
~DoubleType
DoubleType, 59
doVariableVal
WclParams, 121
dumpDefaultResources
WclInterface, 115
dumpMessage
WclParams, 121
dumpMessageList
WclParams, 121
dumpMessageListResources
WclInterface, 115
dumpParamsResourceFile
SoundControlFile, 62
dumpVal
WclParams, 121
dumpWclResourceFile
WclParams, 121
dumpWclResources
WclInterface, 115
duplicateString
SoundObj, 71

eatComment
SCFParsing, 24
EnumAsString
Val, 101
enumAsString
DataType, 49
EnumType, 64
EnumType
EnumType, 64
~EnumType
EnumType, 64
execute
BaseSonicWidget, 7
BooleanSonicWidget, 23
InitializingSonicWidget, 88
LoopingSonicWidget, 103
Message, 118
MultiSonicWidget, 5
SimpleSonicWidget, 43
SoundControlFile, 62
executeMessage
SoundDevice, 68

FileReader
FileReader, 72
~FileReader
FileReader, 72
fixupSize
SWMultiParams, 34
float
ArrayType, 2
CharType, 31
DataType, 50
DoubleType, 59
EnumType, 65
FloatType, 78
IntType, 93
FloatType
FloatType, 78
~FloatType
FloatType, 78
functionList

TransformFunctions, 96

generalSWParamsCallback
SWMultiParams, 35
generalXSWIMemberCallback
DemoInterface, 55
XSonicWidgetInterface, 127
genName
WclInterface, 115
get
Samples, 38
WaveTable, 111
getArg
IndexList, 82
MessageList, 124
ValList, 105
getCallbackArgs
SWMultiParams, 34
getData
Buffer, 26
getDefault
SoundControlFile, 62
WclInterface, 115
getElement
PtrList, 17
getFd
Socket, 52
SoundDevice, 68
getFp
Socket, 52
getHeader
BaseSonicWidget, 7
getId
SCFParsing, 25
getIndex
IndexList, 82
getIndexList
Message, 118
getLength
Buffer, 26
getLowerBound
DataType, 49
getMessage
WclInterface, 116
getName
BaseSonicWidget, 7
Message, 118
getNumber
Message, 118
getOption
Socket, 52
getPort
TCPSocket, 85
getSCF
BaseSonicWidget, 7
getTok
SCFParsing, 24, 25
GetTransformFunction
Val, 101
getTransformFunction
BaseSonicWidget, 8
SoundControlFile, 62
SWMultiParams, 34
getUpperBound
DataType, 49
getValList
Message, 118
growList
PtrList, 17

HighlightMessageArgs
WclInterface, 116

idNumber
BooleanMenu, 19
DataRange, 40
DataTransform, 45
Linear, 98
PassThrough, 12
Step, 79
index
Val, 101
IndexList
IndexList, 82
~IndexList
IndexList, 82
InitEnumFromFile
Val, 101

initFromFile
EnumType, 64
InitFromString
Val, 101
initialize
BaseSonicWidget, 8
BooleanSonicWidget, 23
InitializingSonicWidget, 88
MultiSonicWidget, 5
SimpleSonicWidget, 43
SmallWCLWidget, 47
SoundControlFile, 61
initSoundDevice
SoundControlFile, 61
int
ArrayType, 2
CharType, 31
DataType, 49
DoubleType, 59
EnumType, 64
FloatType, 78
IntType, 93
IntType
IntType, 93
~IntType
IntType, 93
isBounded
DataType, 49

kind
Val, 101

listen
TCPSocket, 85
looksValid
BaseSonicWidget, 7
SonicWidgetCreator, 57

Mapfile
Mapfile, 112
~Mapfile
Mapfile, 112
mapInt
Mapfile, 112
mapString
Mapfile, 112
mapToDurationCallback
DemoInterface, 54
mapToPitchCallback
DemoInterface, 54
mapToTimbreCallback
DemoInterface, 54
mapToVolumeCallback
DemoInterface, 54
Message
Message, 118
~Message
Message, 118
MessageList
MessageList, 124
messages
SoundDevice, 67
MIDISoundDevice
MIDISoundDevice, 108
~MIDISoundDevice
MIDISoundDevice, 108
myClassName
BaseSonicWidget, 7
BooleanSonicWidget, 23
InitializingSonicWidget, 88
LoopingSonicWidget, 103
MultiSonicWidget, 5
SimpleSonicWidget, 43
SonicWidgetCreator, 57

name
SoundDevice, 67
Val, 101
newFile
FileReader, 72
nextValue
FileReader, 72, 73
noteOff
MIDISoundDevice, 108
noteOn
MIDISoundDevice, 108
numberArgs
PtrList, 17

numSCFs
BaseSonicWidget, 7

onSignal
WclInterface, 115
openConnection
SoundDevice, 67
operator+
Samples, 39
operator=
ArrayType, 2
EnumType, 64
IndexList, 82
ValList, 106
Oscil
Oscil, 9
~Oscil
Oscil, 9
outputPtrList
WclParams, 121

pause
MIDISoundDevice, 108
pausePlay
SparcAudioHardware, 76
peekTok
SCFParsing, 25
playFraction
SparcAudioHardware, 76
playNote
MIDISoundDevice, 108
SparcAudioHardware, 76
SPARCSoundDevice, 29
playNoteOnChannel
TG33SoundDevice, 90
playNumber
SparcAudioHardware, 76
SPARCSoundDevice, 29
playSampleFile
SPARCSoundDevice, 29
playSamples
SparcAudioHardware, 75
print
ArrayType, 2
BoolType, 15
CharType, 31
DataType, 49
DoubleType, 59
EnumType, 64
FloatType, 78
IntType, 93
printConfiguration
BooleanMenu, 19
DataRange, 41
DataTransform, 45
Linear, 98
PassThrough, 13
Step, 80
printOn
ArrayType, 3
BaseSonicWidget, 8
BaseSoundDevice, 11
BooleanMenu, 19
BooleanSonicWidget, 23
BoolType, 15
CharType, 31
Data, 37
DataRange, 41
DataTransform, 45
DataType, 50
DemoInterface, 55
DoubleType, 59
EnumType, 64
EnumVal, 70
FileReader, 73
FloatType, 78
IndexList, 83
InitializingSonicWidget, 88
IntType, 93
Linear, 98
LoopingSonicWidget, 103
Message, 119
MessageList, 124
MIDISoundDevice, 109
MultiSonicWidget, 5
PassThrough, 13
RemoteFileType, 21
SCFParsing, 25

SimpleSonicWidget, 43
SonicWidgetCreator, 57
SoundControlFile, 62
SoundDevice, 68
SoundObj, 71
SPARCSoundDevice, 29
Step, 80
SWMultiParams, 35
TCPSocket, 86
TransformFunctions, 96
ValList, 105
WclInterface, 116
WclParams, 122
XSonicWidgetInterface, 127
printVal
Val, 101
promptTransformFunctionMenu
SWMultiParams, 34
PtrList
PtrList, 17
~PtrList
PtrList, 17
put
Samples, 38
WaveTable, 111
PutLineResourceCB
SWMultiParams, 34

quitCallback
XSonicWidgetInterface, 126

readConfiguration
BooleanMenu, 19
DataRange, 41
DataTransform, 45
Linear, 98
PassThrough, 12
Step, 80
TransformFunctions, 96
readMessages
SoundControlFile, 61
readSoundDevice
SoundControlFile, 61
receive
Data, 37
Socket, 52
SoundDevice, 67
reconfigDevice
SparcAudioHardware, 75
recordSamples
SparcAudioHardware, 75, 76
recString
Data, 37
registerCallbacks
DemoInterface, 54
SWMultiParams, 33
XSonicWidgetInterface, 126
registerMemberCallback
DemoInterface, 54
SWMultiParams, 35
XSonicWidgetInterface, 127
registerVariableVal
SoundControlFile, 61
reinitialize
MIDISoundDevice, 108
SPARCSoundDevice, 29
TG33SoundDevice, 91
reRunCallback
XSonicWidgetInterface, 126
reset
SparcAudioHardware, 76
resetAllControllers
TG33SoundDevice, 91
resetAllToggles
SWMultiParams, 33
resizeListTo
PtrList, 17
resumePlay
SparcAudioHardware, 76
retname
Val, 101
revertTransformFunctionCallback
SWMultiParams, 34
rewindFile
FileReader, 73
run
DemoInterface, 55
SmallWCLWidget, 47

SWMultiParams, 34
WclInterface, 115
XSonicWidgetInterface, 127
runCallback
DemoInterface, 54
XSonicWidgetInterface, 126
runLoop
DemoInterface, 54

samplerate
WaveTable, 111
Samples
Samples, 38
~Samples
Samples, 38
SCFParsing
SCFParsing, 24
~SCFParsing
SCFParsing, 24
selectChannel
MIDISoundDevice, 108
SelectFunction
TransformFunctions, 96
selectInstrument
MIDISoundDevice, 108
SparcAudioHardware, 75
SPARCSoundDevice, 29
TG33SoundDevice, 90
selectNewDataFileCallback
XSonicWidgetInterface, 126
selectTransformFunctionCallback
SWMultiParams, 34
selectVoiceBank
TG33SoundDevice, 90
send
Data, 36, 37
Socket, 52
SendMessage
WclInterface, 116
setAcknowledge
SoundDevice, 67
setArg
ValList, 106
setBalance
MIDISoundDevice, 109
TG33SoundDevice, 90
setBlocking
BaseSonicWidget, 8
SoundControlFile, 61
setBounds
BoolType, 15
DataType, 49
setChannelPressure
MIDISoundDevice, 109
setData
Buffer, 27
setDefault
SoundControlFile, 62
setElement
PtrList, 17
setKeyPressure
MIDISoundDevice, 108
setModulationWheel
TG33SoundDevice, 90
setNumber
Message, 118
setOperation
BooleanMenu, 18
setOption
Socket, 52
setOutputPort
SparcAudioHardware, 75
SPARCSoundDevice, 29
setPitchBend
MIDISoundDevice, 108
setPitchBendGranularity
TG33SoundDevice, 90
setPortamento
MIDISoundDevice, 109
setRecordVolume
SparcAudioHardware, 75
SPARCSoundDevice, 29
setSustain
MIDISoundDevice, 109
SetTransformFunction
Val, 101
setTransformFunction
BaseSonicWidget, 8

SoundControlFile, 62
setValue
BooleanMenu, 19
DataRange, 41
DataTransform, 45
Linear, 98
Step, 80
setValueFromDouble
ArrayType, 2
CharType, 31
DataType, 49
DoubleType, 59
EnumVal, 70
FloatType, 78
IntType, 93
SetValueFromString
Val, 101
setValueFromString
ArrayType, 2
BoolType, 15
CharType, 31
DataType, 49
DoubleType, 59
EnumVal, 70
FloatType, 78
IntType, 93
setVectorControlXaxis
TG33SoundDevice, 90
setVectorControlYaxis
TG33SoundDevice, 90
setVibrato
MIDISoundDevice, 109
setVolume
MIDISoundDevice, 109
SparcAudioHardware, 75
SPARCSoundDevice, 29
size
Samples, 39
WaveTable, 111
SmallWCLWidget
SmallWCLWidget, 46
~SmallWCLWidget
SmallWCLWidget, 46
Socket
Socket, 51
~Socket
Socket, 51
SonicWidgetCreator
SonicWidgetCreator, 56
~SonicWidgetCreator
SonicWidgetCreator, 56
SoundControlFile
SoundControlFile, 61
~SoundControlFile
SoundControlFile, 61
SoundDevice
SoundDevice, 67
~SoundDevice
SoundDevice, 67
SoundObj
SoundObj, 71
~SoundObj
SoundObj, 71
SparcAudioHardware
SparcAudioHardware, 75
~SparcAudioHardware
SparcAudioHardware, 75
SPARCSoundDevice
SPARCSoundDevice, 28
~SPARCSoundDevice
SPARCSoundDevice, 28
Step
Step, 79
stepCallback
XSonicWidgetInterface, 126
stopCallback
DemoInterface, 54
XSonicWidgetInterface, 126
stopPlayingSamples
SparcAudioHardware, 75
stringToType
Val, 101
SWMultiParams
SWMultiParams, 33
~SWMultiParams
SWMultiParams, 33
sysparams
SoundDevice, 67

TCPSocket
    TCPSocket, 85
~TCPSocket
    TCPSocket, 85
TG33SoundDevice
    TG33SoundDevice, 90
~TG33SoundDevice
    TG33SoundDevice, 90
TransformFunctions
    TransformFunctions, 96
~TransformFunctions
    TransformFunctions, 96
triggerAlarmsCallback
    DemoInterface, 54
type
    Val, 100

updateDefaultCallback
    SWMultiParams, 35
updateDefaultFromListCallback
    SWMultiParams, 35
updateDefaultFromSliderCallback
    SWMultiParams, 35
updateDefaultFromTextCallback
    SWMultiParams, 35
updateDefaultFromToggleCallback
    SWMultiParams, 35
updateRevertLabel
    SWMultiParams, 34
updateTransformDefaultCallback
    SWMultiParams, 35

Val
    Val, 100
~Val
    Val, 100
ValList
    ValList, 105
~ValList
    ValList, 105

WaveTable
    WaveTable, 110
~WaveTable
    WaveTable, 111
WclInterface
    WclInterface, 115
~WclInterface
    WclInterface, 115
WclParams
    WclParams, 121
~WclParams
    WclParams, 121
writeParamsResourceFile
    BaseSonicWidget, 8
    InitializingSonicWidget, 88
writeResourceFiles
    SWMultiParams, 33
writeSCF
    BaseSonicWidget, 8
    SoundControlFile, 62
writeWCF
    BaseSonicWidget, 8
writeWCFCallback
    XSonicWidgetInterface, 126

XSonicWidgetInterface
    XSonicWidgetInterface, 126
~XSonicWidgetInterface
    XSonicWidgetInterface, 126